Detecting Semantic Similarity: Biases, Evaluation and Models

by

Nicole Peinelt

A thesis submitted to the University of Warwick in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

January 2021

Contents

Abstract

Semantic Similarity Detection refers to a collection of binary text pair classification tasks which aim to indicate certain semantic relations between two short texts. Modelling the semantic similarity between texts is a fundamental task in natural language processing with many applications. This is a challenging problem because the same meaning can be conveyed through a variety of expressions which requires a deep understanding of language beyond mere surface structure and heuristics. This thesis addresses Semantic Similarity Detection in a generic framework and places importance on its application for Community Question Answering (CQA). CQA aims at harnessing user-generated texts from question-answering websites and online forums to answer complex new questions. This involves two crucial Semantic Similarity Detection tasks: question paraphrase detection and answer identification. CQA content is characterised by informal language and domain-specific vocabulary, making it a demanding research area with practical application.

This thesis sets out to examine existing dataset biases and propose new methods for evaluating and modelling semantic similarity between text pairs. We shed light on lexical overlap bias in existing datasets and introduce alternative evaluation metrics which take direct word overlap between text pairs into account. Our metrics highlight model performance on difficult dataset instances, resulting in a more rigorous evaluation setup. We follow by investigating whether alternative information sources can be leveraged for more resilient and effective Semantic Similarity Detection models. Our first approach incorporates topic models with successful neural architectures. We experiment with a topic-enriched CNN-LSTM model, which subsequently leads to the development of a framework for combining topic models with a pre-trained Transformer model. Our second approach incorporates linguistically-enriched word embeddings with pre-trained Transformers by introducing a versatile and lightweight method for injecting dependency-based and counter-fitted embeddings into BERT. Finally, we summarise our main findings and discuss directions for future work.

Acknowledgements

First and foremost, I would like to express my profound gratitude to my supervisor Prof. Maria Liakata who patiently guided me in my transition from Chinese Studies to Computer Science, tutoring me even before the official start of my doctoral studies. Thank you for teaching me how to do research and supporting me throughout my PhD. I appreciate that you took the time for weekly progress meetings (and even daily Skype calls in the run-up to the submission of my first paper), as well as all the late-night paper revisions.

I am deeply grateful to my co-authors Dr. Dong Nguyen and Dr. Marek Rei for their insightful comments and suggestions. It was invaluable to know that I could always turn to you for kind words and advice.

My sincere gratitude goes to Dr. Adam Tsakalidis for providing feedback on early paper drafts and the other members of the Alan Turing Natural Language Processing group for sharing their experience and insights with me. In addition, I would like to thank The Alan Turing Institute for awarding me with a full doctoral studentship and Microsoft Azure for providing the computing resources for all experiments in this thesis.

I would like to offer my special thanks to Prof. Andreas Vlachos for the unique opportunity to join the Natural Language and Information Processing Research Group (NLIP) at the University of Cambridge as a visiting student. This was the most inspiring and productive environment I could have hoped for during my final year.

I am thankful to Prof. Yulan He, Prof. Preslav Nakov, Prof. Graham Cormode and Prof. Jane Sinclair for their valuable suggestions and feedback regarding my doctoral research.

Finally, I would like to thank my family. Your unwavering support and belief in me gave me the strength to keep going through Brexit and a global pandemic. Mum, thank you for teaching me to value education and for always being there for me. Paul, thank you for surrendering your workspace to me during the first Covid-19 lockdown, so I could concentrate on writing up my final conference paper. Finally, I am immensely grateful to my husband for inspiring me to pursue a PhD, encouraging me during setbacks and supporting me all the way to submission.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work presented (including data generated and data analysis) was carried out primarily by the author. Parts of this thesis have been published as full papers by the author:

- Nicole Peinelt, Maria Liakata, and Dong Nguyen. Aiming beyond the Obvious: Identifying Non-Obvious Cases in Semantic Similarity Datasets. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2792–2798, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1268.

- Nicole Peinelt, Dong Nguyen, and Maria Liakata. Better Early than Late: Fusing Topics with Word Embeddings for Neural Question Paraphrase Identification. *arXiv:2007.11314 [cs]*, July 2020a.

- Nicole Peinelt, Dong Nguyen, and Maria Liakata. tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7047–7055, Seattle, USA, 2020b. Association for Computational Linguistics.

Other parts of this thesis are currently under review:

- Nicole Peinelt, Marek Rei, and Maria Liakata. GiBERT: Introducing Linguistic Knowledge into BERT through a Lightweight Gated Injection Method. *arXiv:2010.12532 [cs]*, October 2020c.

The author has also published the following works, albeit not directly linked to this thesis:

- Nicole Peinelt, Maria Liakata, and Shu-Kai Hsieh. ClassifierGuesser: A Context-based Classifier Prediction System for Chinese Language Learners. *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 41–44, 2017.

- Chieh-Yang Huang, Nicole Peinelt, and Lun-Wei Ku. Automatically Suggesting Example Sentences of Near-Synonyms for Language Learners. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 302–306, Osaka, Japan, 2016.

Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag-of-Words
CNN	Convolutional Neural Network
CBOW	Continuous Bag-of-Words
CQA	Community Question Answering
DC	Dice Coefficient
ELMo	Embeddings from Language Models
FFN	Feed-Forward Neural Network
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
GloVe	Global Vectors for Word Representation
IR	Information Retrieval
JSD	Jensen-Shannon Divergence
LSTM	Long Short-Term Memory
LR	Linear/Logistic Regression
MAP	Mean Average Precision
MSRP	Microsoft Research Paraphrase Corpus
NER	Named Entity Recognition
NN	Neural Network
NLG	Natural Language Generation
NLI	Natural Language Inference

NLP	Natural Language Processing
NLU	Natural Language Understanding
NSP	Next Sentence Prediction
PAWS	Paraphrase Adversaries from Word Scrambling
QA	Question Answering
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RTE	Recognising Textual Entailment
SemEval	International Workshop on Semantic Evaluation
SGD	Stochastic Gradient Descent
STS	Semantic Textual Similarity
SVM	Support Vector Machine
tanh	Hyperbolic Tangent
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Semantic Similarity Detection The term Semantic Similarity Detection refers to a collection of text pair classification tasks which focus on automatically recognising the presence of certain semantic relations between two short texts (we provide a detailed definition in Chapter 3). Modelling the semantic similarity between text pairs is a challenging research problem because the same meaning can be conveyed through various lexical and syntactic expressions. Detecting semantically related texts despite *language variability* requires a deep understanding of language beyond superficial surface structure and simple heuristics. While recently proposed models have reported impressive performance gains, *dataset biases* make it difficult to realistically understand current model limitations and train resilient models for real-world applications. Multiple studies in natural language understanding have highlighted biases within existing datasets which lead to an inflation of evaluation scores and overestimation of model capabilities (Wadhwa et al., 2018a; Rajpurkar et al., 2018). For example, Kaushik and Lipton (2018) showed that reading comprehension models could exploit specific dataset properties to achieve high evaluation scores even when crucial task information, such as the question or the passage, was withheld. Similarly, Gururangan et al. (2018) showed that natural language inference model performance could be inflated by annotation artefacts, enabling models to predict the correct class by only looking at the hypothesis without observing the premise. However, biases across the broader area of Semantic Similarity Detection are not well understood. Therefore, this work sets out to investigate existing biases and proposes resilient evaluation methods and models for Semantic Similarity Detection.

Context A wide range of natural language processing problems require modelling the semantic similarity between two texts. This includes applications such as plagiarism detection, automated answer grading, question answering and text generation. Plagiarism detection systems compare sections from different documents to identify suspicious segments. Although copy-paste strategies can be easily detected based on word overlap, more subtle approaches, such as paraphrasing or bilingual plagiarism, require more sophisticated Semantic Similarity Detection methods (Gripp et al., 2014). In automated short answer grading, the semantic similarity between student answers and one or more correct answers can be estimated to automatically assess the quality of learner responses (Mohler et al., 2011). In question answering, estimating the semantic similarity between questions can be used to utilise existing answers to previously asked questions (Nakov et al., 2017). In text generation, evaluation of natural language generation (NLG) systems is an unsolved problem. Manual evaluation of generated texts is expensive and not easily scalable. Therefore generated candidate sentences are often compared with a reference sentence through automatic metrics. As commonly used metrics (such as ROUGE) are based on exact match and do not robustly capture paraphrases, Semantic Similarity Detection models are a promising direction towards more robust automatic NLG evaluation (Zhang et al., 2019b). This thesis proposes a generic framework for detecting certain relations between text pairs based on semantic similarity, dubbed Semantic Similarity Detection, which is introduced in Chapter 3.

Approach Many techniques have been proposed for semantic similarity tasks, including statistical, feature-engineered and - more recently - neural approaches. While earlier work has focused primarily on dataset- and task-specific methods, this thesis contributes to developing neural Semantic Similarity Detection models. This choice is motivated by the potential of neural architectures to work well across a variety of datasets and related tasks without the need for manual feature-engineering, which is time-consuming and does not generalise well. To harness the versatility of neural approaches, this work addresses the problem of Semantic Similarity Detection in a generic and flexible framework which mainly focuses on exploiting the provided text pairs, while ignoring any metadata which is highly dataset-specific and not available for all datasets. Instead, we experiment with incorporating additional information sources (such as topics and linguistically-enhanced embeddings) which are applicable across all Semantic Similarity Detection datasets to develop a more robust approach to neural Semantic Similarity Detection and decrease reliance on surface form similarity. At the same time, importance is placed on the practical and

particular challenging application of semantic similarity to Community Question Answering (CQA).

Challenges in CQA The internet has given rise to various online communities, such as web forums and specialised question answering websites (e.g. Quora, StackOverflow, YahooAnswers). These platforms allow users to seek and share information by asking and answering free text questions. The growing research area of CQA aims at harnessing this community-generated question answering content to answer new questions automatically. The questions discussed in CQA communities tend to be long, open-ended and domain-specific, as opposed to short fact seeking questions for which answers can be easily obtained with search engines. Therefore, improvements in CQA are of particular interest for the larger field of question answering, which currently still struggles to solve such complex questions. Automatically answering CQA questions typically involves two core Semantic Similarity Detection tasks: question paraphrase detection and answer selection, which are both addressed in this thesis. Analysing content from online communities poses unique challenges for Semantic Similarity Detection due to the use of *informal and domain-specific language*, making it a demanding research area with important practical applications.

1.2 Research Questions

The overall objective of this thesis is to better understand existing biases across current Semantic Similarity Detection datasets and to mitigate these by developing challenging evaluation metrics and more resilient models. More specifically, this thesis addresses the following research questions:

- **RQ1** Which biases exist in current semantic similarity datasets?
  - **OBJ 1.1** Select a range of well-known and commonly used Semantic Similarity Detection datasets.
  - **OBJ 1.2** Analyse semantic similarity dataset properties and identify bias patterns.
  - **OBJ 1.3** Demonstrate how models can exploit the identified biases.
  - **OBJ 1.4** Quantify the prevalence of identified biases across well-known semantic similarity datasets.
- **RQ2** What are ways to account for identified dataset biases during model evaluation?

  - **OBJ 2.1** Understand currently used evaluation metrics and their limitations.
  - **OBJ 2.2** Demonstrate how identified dataset biases impact evaluation.
  - **OBJ 2.3** Propose novel evaluation techniques which account for identified dataset biases.
  - **OBJ 2.4** Demonstrate the effectiveness of proposed evaluation techniques.

- **RQ3** Can alternative information sources be exploited to make Semantic Similarity Detection models more resilient to the identified biases?
  - **OBJ 3.1** Attempt to combine topics with neural models as an additional semantic similarity signal.
  - **OBJ 3.2** Attempt to incorporate additional linguistic information into pre-trained models.
  - **OBJ 3.3** Propose mechanisms for combining external information with existing models and evaluate which of these techniques are most effective.
  - **OBJ 3.4** Evaluate if the incorporation of addition information into pre-trained models improves model performance and robustness.
  - **OBJ 3.5** Understand the properties of cases in which proposed models improve.

- **RQ4** How can one develop a generic Semantic Similarity Detection model which performs well across a variety of tasks and datasets?
  - **OBJ 4.1** Identify well-known datasets covering a wide range of Semantic Similarity Detection tasks.
  - **OBJ 4.2** Investigate the existing literature in the area of Semantic Similarity Detection to understand common approaches and promising directions.
  - **OBJ 4.3** Develop robust and accurate neural models for Semantic Similarity Detection which do not require feature engineering or dataset-specific information.
  - **OBJ 4.4** Evaluate developed models against dataset-specific feature-engineered systems and other neural baselines.

1.3 Challenges

This thesis addresses the complex task of Semantic Similarity Detection, especially in the context of Community Question Answering (CQA). This involves the following challenges:

- **C1 Language variability** An inherent feature of natural languages is that they can express the same meaning in various ways (Dagan and Glickman, 2004; Dagan et al., 2009). Language can vary between individuals - or even the same individual in different contexts - in one or several aspects, including pronunciation, word choice, morphology and syntax. As a result, automatic reasoning about the relationship between text pairs requires the ability to understand different ways of expressing meaning, rather than simply matching similar surface forms. In this thesis, this challenge is addressed by combining neural text representations with additional complementary information sources for capturing similar meaning, such as topic models and linguistically enriched embeddings (RQ3 and Chapters 6-7).

- **C2 Lexical overlap in datasets** Semantic similarity between texts can arise from various factors, including lexical overlap, syntactic constructions and the use of synonyms. While lexical overlap may imply semantic similarity, there are common exceptions, for example when negation expressions or changes in word order are involved. Construction techniques for many paraphrase and semantic similarity datasets rely on lexical overlap, which can overemphasise the role of lexical overlap and under-represent more interesting cases of semantic similarity (Zhang and Patrick, 2005; Rus, 2014). Superficial word overlap in datasets could even provide clues for class membership and be exploited by models to achieve high evaluation scores without any deeper language understanding. This is problematic, especially in real-world situations where algorithms are more likely to be confronted with less apparent cases of semantic similarity. This thesis addresses this challenge by analysing surface structure similarity and quantifying lexical overlap bias in existing datasets. Based on this, the thesis further proposes alternative evaluation metrics which emphasise performance on more challenging cases of semantic similarity (RQ1, RQ2 and Chapter 5) which are used to evaluate subsequently proposed models (RQ3 and Chapters 6-7).

- **C3 Domain-specific language** CQA platforms offer users the opportunity to ask free text questions with few restrictions. However, receiving an answer

requires to wait for other users' responses, while search engines and automated question answering systems can provide immediate responses. This incentivises users to primarily ask questions for which they cannot obtain satisfactory answers through the aforementioned alternatives, which tends to be the case with long, complicated and domain-specific questions (Blooma and Kurian, 2011). Moreover, many online forums revolve around a shared specific interest, and it has been demonstrated that socialisation processes in such online communities lead to the development of specialised language, such as forum-specific jargon (Nguyen and Rose, 2011; Danescu-Niculescu-Mizil et al., 2013). (Pre-)Training models on general-purpose datasets and applying them to content from a specialised online community may lead to domain shift and out-of-vocabulary issues, resulting in reduced model performance. This challenge is addressed by incorporating topic models into pre-trained models which resulted in a performance improvement on domain-specific examples in several CQA datasets (RQ3 and Chapter 6.5).

- **C4 Class imbalance** Many existing real-world semantic similarity datasets are imbalanced, containing more examples of one class than another (Rus, 2014). This is especially prominent in CQA datasets, where the majority of existing posts on a CQA platform are not relevant to a newly posted question, and negative examples tend to outweigh positive examples. Evaluation on imbalanced datasets can lead to inflated evaluation scores for models which only predict the majority class. In this thesis, this is accounted for by evaluating models on various datasets with different label distributions and using F1 score as the main evaluation metric which balances performance on all classes (RQ3, RQ4 and Chapters 6-7).

1.4 Contributions

This thesis makes the following main contributions:

- We characterise current Semantic Similarity Detection datasets in terms of direct word overlap between sentence pairs and the prevalence of lexical overlap bias (RQ 1 and Chapter 5).

- We propose a criterion to distinguish between obvious and non-obvious instances of semantic similarity and analyse to what extent such instances exist in current datasets (RQ1, RQ2 and Chapter 5).

- We introduce an alternative evaluation metric which focuses on non-obvious cases of semantic similarity (RQ2 and Chapter 5).
- We propose a topic-aware CNN-LSTM architecture for Semantic Similarity Detection, which improves over other neural models (RQ3, RQ4 and Chapter 6.4).
- We propose a topic-enhanced Transformer model for semantic similarity prediction, which outperforms other neural models in both F1 and stricter evaluation metrics (RQ3, RQ4 and Chapter 6.5).
- We show in an error analysis that gains of our topic-enhanced Transformer model are prominent on domain-specific cases, such as those encountered in online question answering communities (RQ3 and Chapter 6.5).
- We propose a generic and lightweight method for injecting linguistically-enhanced embeddings into a pre-trained Transformer model (RQ3, RQ4 and Chapter 7).
- Through detailed error analysis, we highlight examples where our linguistically-enhanced Semantic Similarity Detection model provides improved performance, such as in cases of sentence pairs involving synonym pairs (RQ3 and Chapter 7).

1.5 Thesis Outline

Overall, this thesis follows a traditional structure which is illustrated in Figure 1.1. It starts with an introduction, setting out the motivation, goals and challenges of the undertaken research (Chapter 1). This is followed by the methodological (Chapter 2) and theoretical foundations (Chapter 3) which provide the necessary

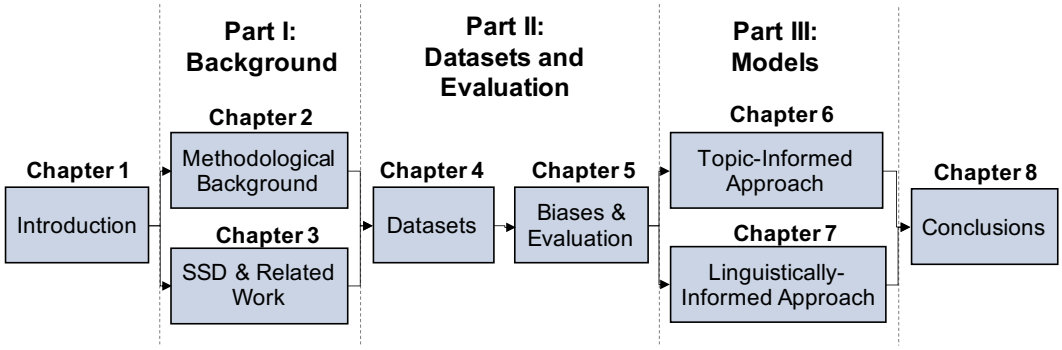


Figure 1.1: Roadmap of the thesis. SSD = Semantic Similarity Detection.

background for the comprehension of this thesis. Then, the datasets which are used in the following analysis and experiments are introduced (Chapter 4). The three main chapters introduce novel methods for detecting dataset biases, as well as evaluating and modelling semantic similarity (Chapters 5-7). The thesis ends with a conclusion and directions for future work (Chapter 8). More specifically, the individual chapters of this thesis cover the following content:

- **Chapter 1**, the current chapter, provides an introduction by discussing the motivation for investigating dataset biases and developing new evaluation and modelling techniques for Semantic Similarity Detection. It further highlights related challenges, sets out the core research goals and summarises the main contributions made in this thesis.

- **Chapter 2** provides a methodological background for this thesis. It introduces foundational methods and concepts in the area of natural language processing and machine learning, as well as other foundational concepts which will be used throughout this thesis.

- **Chapter 3** defines important concepts in the area of semantic similarity and provides a new theoretical framework of Semantic Similarity Detection, contrasting it with existing concepts and describing core tasks. It further provides a literature review for crucial Semantic Similarity Detection tasks.

- **Chapter 4** selects, describes and compares datasets which are used in the following chapters of this thesis for analysis and experiments in subsequent chapters.

- **Chapter 5** characterises Semantic Similarity Detection datasets in terms of surface form similarity between sentence pairs and the degree to which they contain lexical overlap bias. It further proposes a distinction criterion for identifying obvious and non-obvious cases of semantic similarity. Based on that, novel evaluation metrics are proposed which focus on model performance on non-obvious dataset instances. This chapter has been published as an ACL 2019 short paper (provided in Appendix A).

- **Chapter 6** proposes a topic-informed neural approach to Semantic Similarity Detection and consists of two parts. The first part (Chapter 6.4) introduces a model which combines topics with a CNN-BiLSTM architecture. The second part (Chapter 6.5) presents a framework for combining topics with pre-trained Transformers. This chapter is based on our arXiv paper (provided in Appendix B) and our ACL 2020 paper (provided in Appendix C).

- **Chapter 7** proposes a linguistically-informed neural approach. It describes a novel method for injecting linguistically-enriched embeddings into a pre-trained Transformer model for Semantic Similarity Detection. This chapter is currently under submission for NAACL 2021 (provided in Appendix D).
- **Chapter 8** concludes the thesis by discussing the main findings, as well as remaining challenges and directions for future work.

Part I

Background

CHAPTER 2

Methodological Background

2.1 Natural Language Processing

Overview Situated at the intersection between Linguistics and Computer Science (see Figure 2.1), the field of Natural Language Processing (NLP) studies computational methods which 'take as input or produce as output unstructured, natural language data' (Goldberg, 2017). NLP is often used synonymously with Computational Linguistics. However, these two closely related research areas can be distinguished based on their motivation and main focus: NLP aims at developing accurate and efficient computational methods to process language, while computational linguistics uses computational tools to understand human language and its properties better (Tsujii, 2011). In contrast to artificial languages (e.g. programming languages) which have been designed for a specific purpose based on predefined principles, natural language has evolved organically between humans through use and repetition. Therefore, natural language does not follow static rules and strict logic but is highly arbitrary, ambiguous, and variable, representing considerable challenges for NLP.

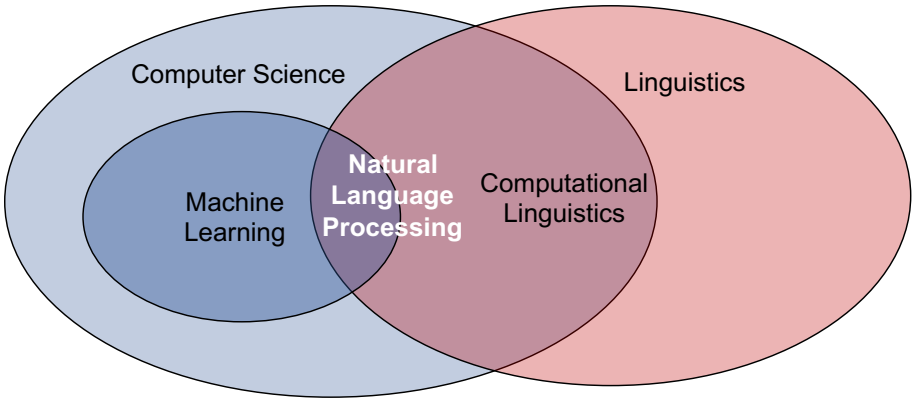


Figure 2.1: Overview of NLP and related disciplines.

A broad range of methods has been developed to automatically process language, reaching from simple regular expressions to complex machine learning algorithms. In recent years, machine learning has become a standard approach to NLP problems. Machine learning approaches aim at learning patterns from data in the form of a model to perform a certain task automatically.

Typical workflow This thesis focuses on using NLP and machine learning techniques to predict semantic similarity. Figure 2.2 illustrates a typical pipeline for this task consisting of the following steps: problem definition, data collection, preprocessing and a repeated process of model design, training, evaluation and tuning:

1. The first step involves **defining the problem** of interest. Conceptualising the problem within a machine learning framework requires identifying the involved variables and determining the relationship between input and output. For supervised problems, the machine learning model learns to predict the output based on the input, while for unsupervised problems, there are no supervising outputs and the model is trained solely based on the input. Furthermore, if output variables exist, they can be continuous (regression problem) or discrete (classification problem). As this thesis focuses on text classification problems (see Chapter 3.1), the following sections primarily focus on the classification scenario.

2. Based on the defined problem, the **data collection** phase involves obtaining suitable data. This can include aggregating material from existing sources (e.g. social media, websites or digital books), asking humans to produce language or annotations in a predefined setup (e.g. crowdsourcing platforms) or a combination of both.

3. Usually, the collected text data requires some form of **preprocessing** to transform the potentially noisy text into more meaningful data. The specific preprocessing steps depend on the defined task and collected data, but often include lowercasing, stemming, stopword removal and tokenisation.

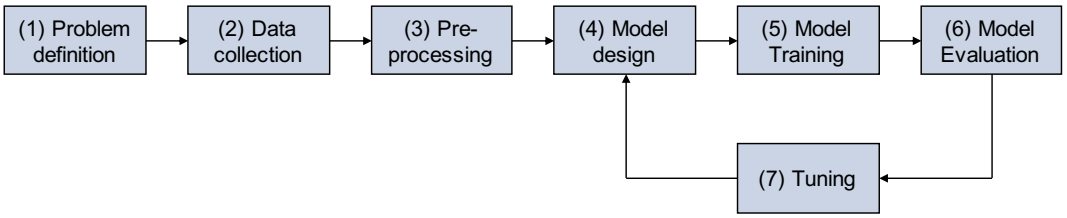


Figure 2.2: Typical steps in a machine learning-based NLP system.

4. **Model design** traditionally involves hand-crafting of features (e.g. n-grams or presence of emojis) and choosing a suitable machine learning classifier. In deep learning models, this step mostly focuses on architecture engineering (e.g. the number of hidden layers or the choice of encoders).

5. This is followed by **model training** on the training portion of the dataset. In the supervised setup, this involves learning weights w of a function $y = f(x, w)$ which maps the input x to the output y based on given input-output pairs (x, y) from the training set.

6. To assess how well the model is performing, **evaluation** metrics are calculated which measure the discrepancy between the predicted labels $\hat{y}$ and the gold labels y.

7. During the **tuning** phase, evaluation metrics are calculated on a separate development set, to choose the best model settings and hyper-parameters. For traditional machine learning models, this step primarily consists of manual feature selection, while in neural models, tuning typically focuses on architecture choices and hyper-parameters. Finally, the model is evaluated on a held-out test set.

The following sections of this chapter present foundational NLP methods based on the order of the above pipeline, covering preprocessing techniques (section 2.2), text representations (section 2.3), machine learning algorithms (section 2.4) and evaluation methods (section 2.5) as a basis for the following chapters of this thesis.

2.2 Preprocessing

Preprocessing focuses on preparing the collected data before the actual learning step. Data may be missing, noisy or contain outliers which can negatively impact the data representation and the resulting machine learning model. Various preprocessing steps can reduce noise, simplify the data and reduce variability, which is especially helpful for smaller datasets. However, there is no ideal universal preprocessing procedure as preprocessing steps are dependent on various factors, such as data source, collection method and final task. Therefore, preprocessing decisions require human expertise to ensure that no important information is removed in this step, which could hurt model performance. In practice, a tradeoff may be necessary to clean and standardise the data but avoid overly specialised preprocessing steps which can limit the transferability of the NLP system. Moreover, when using pretrained representations or models, it is important to match such tools' preprocessing

decisions to avoid a mismatch between the processed data at hand and the pre-trained components.

Lowercasing Through lowercasing, all letters in the text are converted to lower case, e.g. 'Input', 'InPut' and 'INPUT' are all transformed into 'input'. This reduces the vocabulary size and maps tokens which only differ in casing but are otherwise identical to the same concept.

Tokenisation Tokenisers segment a string of characters into individual tokens (e.g. 'in the rain' → ['in' 'the' 'rain']). A simple baseline for English tokenisation uses white space as a splitting criterion, but more sophisticated tokenisers are available which deal more elegantly with special cases such as named entities or hyphenation. In this work, we use the NLTK toolkit tokeniser (Bird and Loper, 2004). An alternative tokenisation approach is to split tokens into smaller units, such as word pieces (Wu et al., 2017b) and byte pair encoding (Sennrich et al., 2016). Using subword units is beneficial for handling out-of-vocabulary items by breaking them down into known components.

Stopword removal Removing stopwords refers to deleting all tokens which appear in a predefined list of words. Stopword lists usually include frequent function words such as 'to' or 'the' which carry little meaning and are not crucial for a specific task. Different lists are possible depending on the purpose, but we use the stopword list provided by the NLTK toolkit (Bird and Loper, 2004) when training topic models in Chapter 6.

Pattern substitution Regular expressions can be used to replace strings which have a regular format such as images, email addresses or webpages (e.g. 'www.bbc.co.uk' or 'www.google.com') with a reserved string, such as '<IMG>', '<EMAIL>' , '<URL>'. This allows obtaining a more meaningful shared representation for such inputs, rather than using the original string which is likely to obtain a poor representation due to data sparsity.

Stemming Stemmers use heuristics to chop off word endings and reduce tokens to their approximate word stem. However, in contrast to lemmatisation (see below) the process is relatively crude and the resulting outputs usually are not actual words. For example, 'beauty' and 'beautiful' would both be mapped to the stem 'beauti'.

Lemmatisation Lemmatisers reduce inflectional forms of a word with a similar motivation to stemming but employ a more deliberate approach based on vocabulary and morphological analysis to map tokens to lemmas (existing words in the dictionary). For example, the tokens 'crying' and 'cries' are mapped to the lemma 'cry'. This reduces data sparsity and can help with model training if data is limited.

In this thesis, we employ minimal preprocessing that involves lowercasing, tokenisation and URL substitutions for our models in Chapters 5-7. When training topic models (in Chapters 6.4 and 6.5), we additionally remove common stopwords.

2.3 Text Representation

In order to build machine learning models for text processing, the textual data needs to be represented as a numerical vector. This section discusses several text representation methods which have been popular in recent years and are being used across this thesis, including bag-of-words, static word embeddings, contextualised representations and topic models.

2.3.1 Sparse Representations

2.3.1.1 One-Hot Representation

The one-hot representation is a simple and intuitive method which represents words as indices of the vocabulary. Given a collection of texts (see Figure 2.3a), the sentences are tokenised to obtain all unique words in the text resulting in a fixed vocabulary $V = (w_1, w_2, ..., w_{|V|})$ with size $|V|$ (see Figure 2.3b). Then each word in the vocabulary is assigned to a dimension of the representation vector. The representation of a given token t is therefore a $|V|$-dimensional binary vector $\mathbf{w}$ which is obtained by indicating the present word with '1' in the corresponding dimension and '0' in all other dimensions (see Figure 2.3c):

$$\mathbf{w_i} = \begin{cases} 1, & \text{if } t = \mathbf{w_i} \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{w} \in \mathbb{R}^{|V|} \tag{2.1}$$

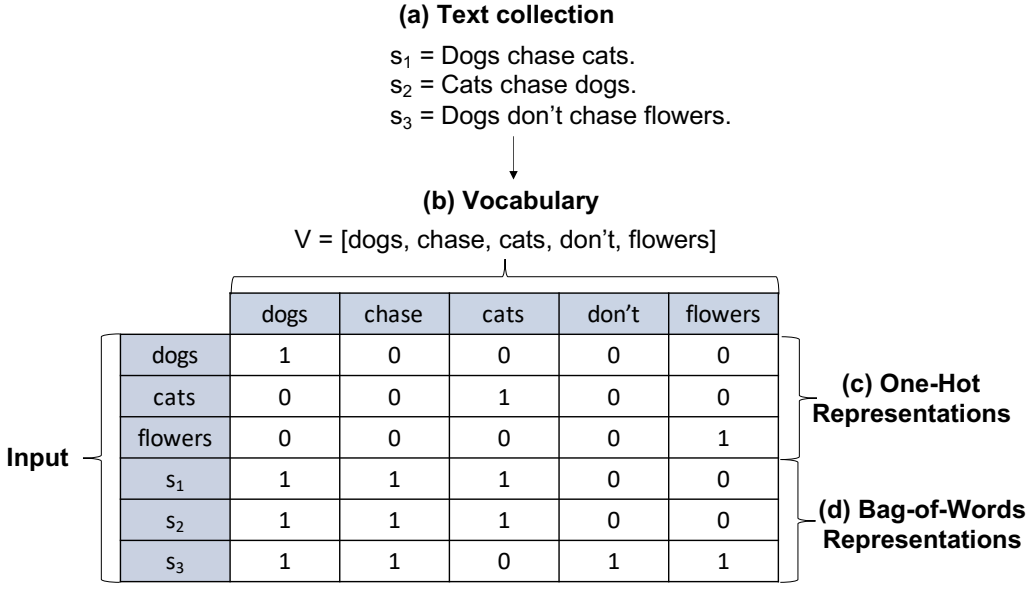


Figure 2.3: Toy example illustrating one-hot and bag-of-words representations.

For example, given a toy vocabulary V = ('dogs', 'chase','cats', 'don't', 'flowers'), individual words are represented as follows:

$$\begin{aligned} \mathbf{w}(\text{dogs}) &= [1,0,0,0,0] \\ \mathbf{w}(\text{cats}) &= [0,0,1,0,0] \\ \mathbf{w}(\text{flowers}) &= [0,0,0,0,1] \end{aligned} \tag{2.2}$$

While this representation can distinguish between multiple distinct words, it has several drawbacks. Firstly, the resulting vectors are very sparse with many dimensions but few non-zero entries, which is not computationally efficient. Secondly, the representation contains no semantic or syntactic information and can therefore provide no information regarding the relationship between words (e.g. 'dog' is as similar to 'flower' as it is to 'cat'). Thirdly, if new words are added to the vocabulary, existing representations need to be updated by adding additional dimensions.

2.3.1.2 Bag-of-Words Representation

The bag-of-words representation is an extension of the one-hot representation. While the one-hot representation only represents a single word, the bag-of-words representation represents a sequence of words $\mathbf{s} = (w_1, w_2, ...w_l)$ with length l by simply

summing over the one-hot representations of all occurring words (see Figure 2.3d):

$$\begin{aligned} \text{BOW}(\mathbf{s}) &= \sum_{i=1}^{l} \mathbf{w_i} \\ \mathbf{w}, \text{BOW} &\in \mathbb{R}^{|V|} \end{aligned} \tag{2.3}$$

For example, the sentence s_1 = 'Dogs chase cats.' would be represented as:

$$\text{BOW}(\mathbf{s}) = [1, 1, 1, 0, 0] \tag{2.4}$$

In addition to the previously mentioned drawbacks of one-hot representations, the bag of word representation does not preserve any information regarding word order. Therefore, the sentence s_2 = 'Cats chase dogs.' would have the same representation as BOW($\mathbf{s_1}$) despite the opposite meaning. Overall, sparse representations such as one-hot and bag-of words are simple and commonly used representations, but have many limitations.

We use bag-of-word representations to measure word overlap in our proposed evaluation metric in Chapter 5.

2.3.2 Static Distributed Representations

Distributed word representations were proposed to address the limitations of the sparse representations (such one-hot and bag-of-word representation which were discussed above). Distributed word representations embed each word into a continuous real-valued vector where the word's meaning is represented based on a combination of multiple dimensions (Liu et al., 2020). In contrast to sparse representations, distributed representations result in dense representations where the dimensionality of $\mathbf{w}$ is much smaller than $|V|$. Distributed representations are generally obtained with methods which are based on the distributional hypothesis ('You shall know a word by the company it keeps', Firth 1957) - the idea that hat words which occur in the same contexts tend to have similar meanings (Harris, 1954). Two particularly popular approaches for obtaining distributed representations are count-based and prediction-based methods.

2.3.2.1 Count-based Approaches

A long line of research has aimed at capturing distributional properties of words based on word-context matrices (Goldberg, 2017). Such count-based approaches

loop over documents to populate a matrix with co-occurrence probabilities between words which indicate the strength of association between them. A potential downside of these approaches is that the resulting matrix is very large and sparse. However, this can be mitigated through applying dimensionality reduction techniques such as singular value decomposition to obtain dense vectors.

2.3.2.2 Prediction-based Approaches

Prediction-based techniques train a model in an unsupervised way to obtain dense word representations. A particular influential example of this approach is word2vec (Mikolov et al., 2013) which is based on Feed-Forward Neural Networks (see Chapter 2.4.2). Word2vec proposes two language modelling architectures: the Continuous Bag-of-Words (CBOW) model and the Skip-gram model. CBOW aims at predicting a word from its surrounding context (Figure 2.4 b), while Skip-gram predicts the preceding and following words of a given word (Figure, 2.4 c). Training instances are generated by moving a sliding window over a text (see Figure 2.4 a). The models

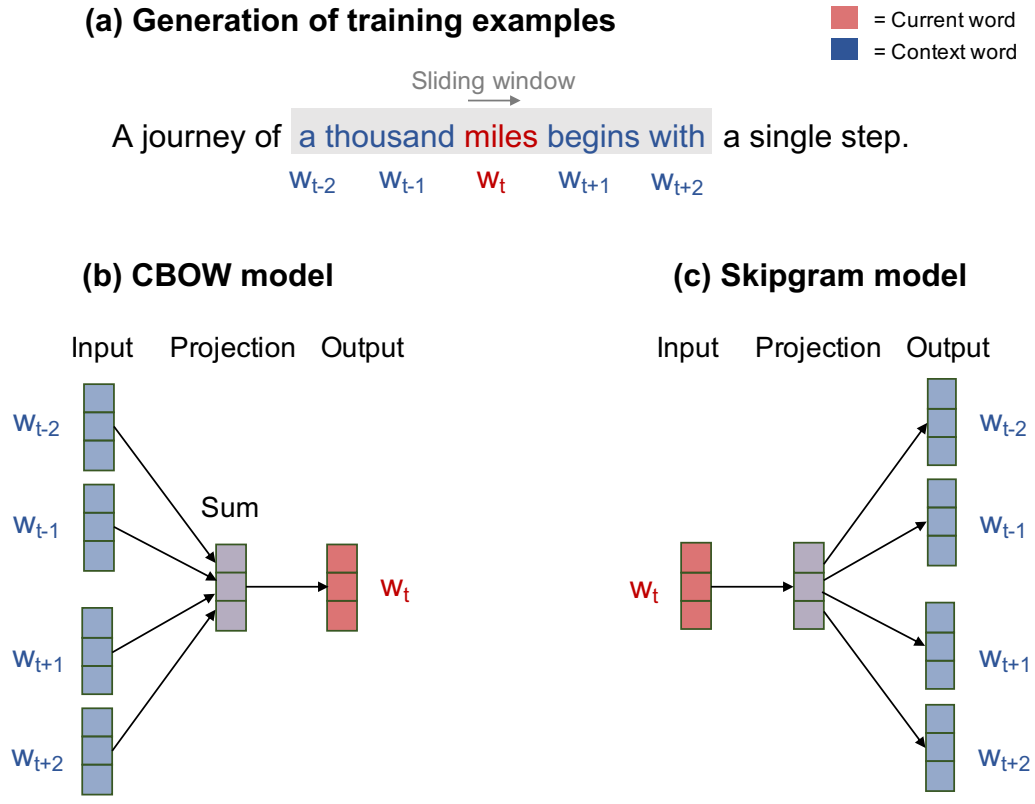


Figure 2.4: Continuous Bag-of-Words (CBOW) and Skip-gram models used in word2vec.

are trained on the Google News corpus, which consists of 100 billion words. After training is complete, individual word representations are extracted from the trained model. These word representations can be inserted into task-specific models and updated during training. This method has successfully boosted model performance across many NLP tasks and achieved enormous popularity. Moreover, word2vec representations have been shown to capture certain semantic and syntactic properties of words in word analogies tasks, such as:

$$\mathbf{w}(King) - \mathbf{w}(Male) + \mathbf{w}(Female) = \mathbf{w}(Queen)$$

Distributed representations for individual words have sparked research into various word embedding improvements and distributed representations beyond the word level, such as doc2vec (Le and Mikolov, 2014).

2.3.2.3 Combined Approaches

Global Vectors for Word Representation (Glove, Pennington et al. 2014) is a combination of count-based and prediction-based methods. The count-based component consists of obtaining global corpus statistics, which are transformed into sparse word representations through matrix factorisation techniques. The prediction-based component encompasses using a loss function to optimise the resulting embeddings.

2.3.3 Contextualised Representations

Contextualised representations build on distributed neural representations. While individual word embeddings such as word2vec (Mikolov et al., 2013) have been very successful in the past, these word representations are static in the sense that they are context-independent. In contrast, more recently proposed contextualised models aim at obtaining context-sensitive word representations. This section discusses two prominent contextualised models - ELMo and BERT - which are also used in this thesis.

2.3.3.1 ELMo

Embeddings from Language Models (ELMo, Peters et al. 2018) is a notable example of contextualised models that model word representations as functions of the entire input sentence. The core idea was to pre-train an entire text encoder in contrast to

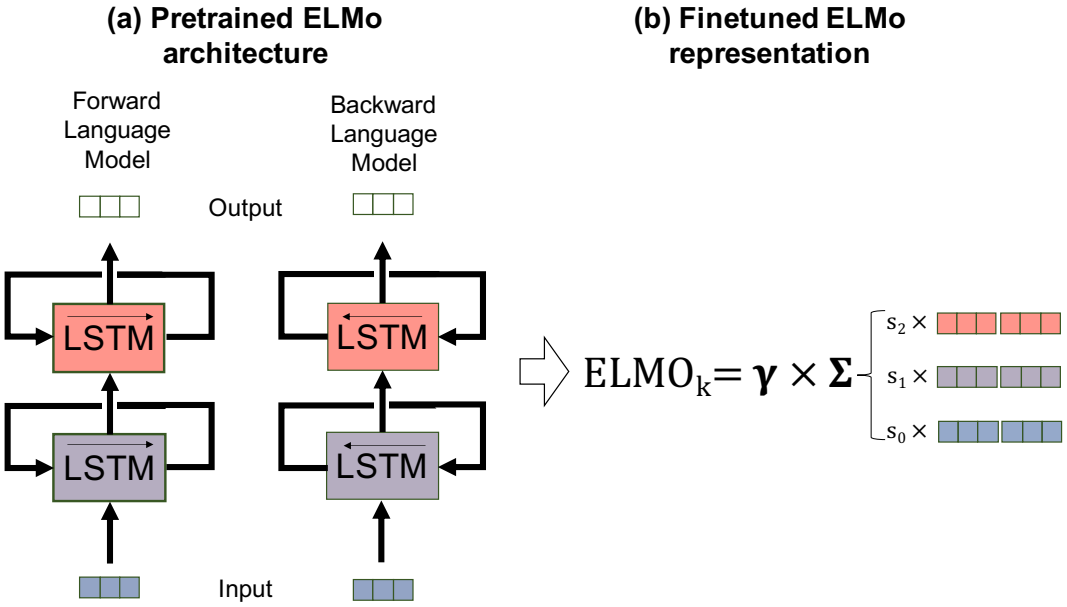


Figure 2.5: Architecture of ELMo with a two-layer BiLSTM. The context-aware representation for a token is obtained based on a γ-scaled weighted sum between the intermediate language model representations and the layer-specific weights (s_0, s_1 and s_2).

earlier work which had focused on pre-training individual word embeddings, while encoders still needed to be learned from scratch. Peters et al. (2018) pre-trained a two-layer bidirectional LSTM (BiLSTM, refer to Chapter 2.4.4) on large amounts of textual data with a language modelling objective (see Figure 2.5 a) and demonstrated performance gains on various downstream tasks. Rather than just using the final layer of the BiLSTM, the authors proposed using a linear combination of representations from the intermediate model layers based on layer-specific weights learned during finetuning on downstream tasks (see Figure 2.5 b). The resulting representations can be used instead or in combination with static word embeddings to boost the performance of existing architectures.

2.3.3.2 BERT

Contextual word representations were further refined by Bidirectional Encoder Representations from Transformers (BERT, Devlin et al. 2019) which achieved state-of-the-art performance on a wide range of tasks. BERT has lead to the creation of the new NLP subfield of BERTology (refer to Rogers et al. 2020 for an overview) and had a considerable impact on subsequent research. Similarly to ELMo, BERT

is a pre-trained contextual language model, but it differs greatly with respect to its architecture, pre-training objectives and finetuning approach.

Architecture BERT's architecture is based on a Transformer encoder model (refer to Chapter 2.4.6 for a detailed description) with minor modifications. Contrary to the majority of previous NLP models, BERT operates on word pieces (Wu et al., 2017b), rather than tokens. For example, the input 'playing' would is tokenised into the word pieces 'play' and '##ing'. This allows the model to maintain a fixed vocabulary of the most frequent word pieces and handle out-of vocabulary words gracefully.[1] Furthermore, BERT encodes two sentences at a time which are separated by a special [SEP] token. In contrast to a vanilla Transformer embedding layer, BERT calculates the representations for the word pieces in the input by summing not only word piece embeddings $\mathbf{E^W}$ and positional embeddings $\mathbf{E^P}$, but also including segment embeddings $\mathbf{E^S}$ which indicate whether the word pieces occur in the first or second sentence:

$$\begin{aligned} \mathbf{H^0} &= \text{LayerNorm}(\mathbf{E^W} + \mathbf{E^P} + \mathbf{E^S}). \\ \mathbf{E^W}, \mathbf{E^P}, \mathbf{E^S}, \mathbf{H^0} &\in \mathbb{R}^{n \times d} \end{aligned} \tag{2.5}$$

The resulting representation is then passed to a Transformer encoder consisting of 12 (for $\text{BERT}_{\text{BASE}}$) or 24 transformer blocks (for $\text{BERT}_{\text{LARGE}}$).

Pre-training During pre-training, the Transformer model is trained with two unsupervised tasks: a masked language modelling task and a next sentence prediction (NSP) pre-training task. For the masked language modelling objective, 15% of the input is replaced by '[MASK]' tokens, where the model needs to predict the original words in the input sequence (see red text in Figure 2.6). The intuition behind this task is to train a fully bidirectional model, as opposed to the traditional language

[1] In the worst case scenario, the model can use individual characters as fallback to avoid an uninformative ['UNK'] representation.

Input: [CLS] the man went to [MASK] bank [SEP] he withdrew a [MASK] amount of cash
Label: IsNext

Input: [CLS] the man went to [MASK] bank [SEP] the [MASK] is an endangered bird
Label: NotNext

Figure 2.6: Two examples illustrating BERT's pre-training tasks: masked language modelling (in red) and next sentence prediction (in blue).

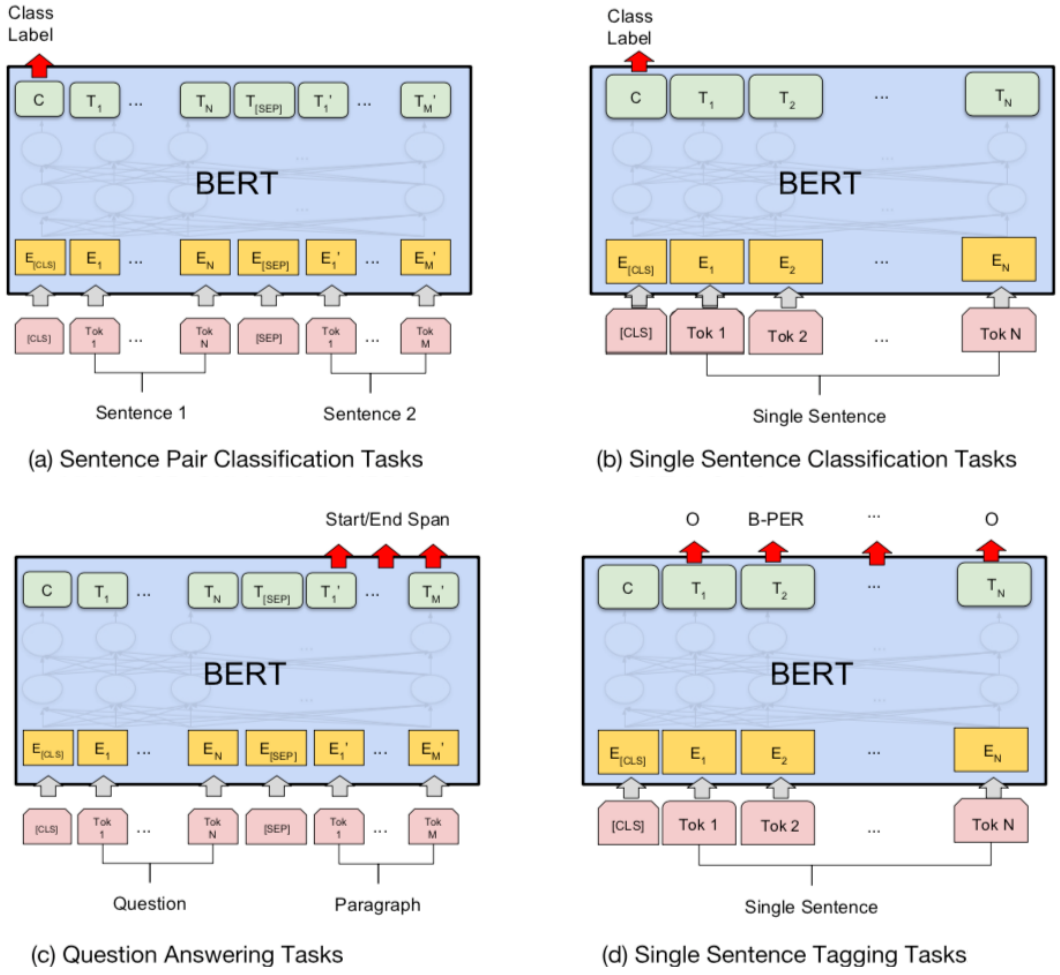


Figure 2.7: Usage of BERT for downstream tasks (Devlin et al., 2019).

modelling setup, which predicts the next word from left to right. For the NSP objective, the model is provided with two sentences A and B, separated by a special '[SEP]' token to indicate sentence boundaries. In 50% of cases, B followed A in the corpus, whereas 50% of the time B was randomly sampled from the corpus. The task is to predict whether the second sentence follows the first sentence (compare blue text in Figure 2.6). The prediction is based on the $\mathbf{c} \in \mathbb{R}^h$ vector which corresponds to the representation of the '[CLS]' token in BERT's last layer (where h indicates the number of hidden units). The NSP task was introduced in order to improve text pair prediction tasks.

Finetuning Due to a flexible finetuning approach, BERT can be used for a variety of downstream tasks. The model can encode one sentence (see Figure 2.7 b) or

a sentence pair by concatenating the sentences with '[SEP]' tokens which indicate sentence boundaries (see Figure 2.7 a,c and d). For segment level prediction tasks, the **c** vector from the final layer is used (see Figures 2.7 a and b), while for token level prediction tasks individual token representations from the final layer are used (see Figure 2.7 d). It is customary to finetune BERT for three epochs and with a small learning rate on down-stream task training data. In contrast to ELMo, all weights in the model are updated during finetuning.

Distributed representations have become the predominant text representation in contemporary NLP and led to creating the subfield of representation learning. We use Glove and ELMo representations in Chapter 6.4 and employ BERT in our experiments in Chapters 6.5 and 7.

2.3.4 Topic Models

Probabilistic topic modelling includes a range of models which analyse texts intending to discover similar themes within the documents. Topic models have been widely used to group related documents together or as features in machine learning models.

2.3.4.1 LDA

The arguably best-known and most commonly-used probabilistic topic model is Latent Dirichlet Allocation (LDA, Blei et al. 2003). It defines a topic as a probability distribution over a fixed vocabulary. The core idea is that each document consists of multiple words which are associated with different topics. Therefore, a document contains a mixture of topics in different proportions that can be used to characterise the whole document. LDA is based on the assumption that every document was generated by an imaginary process which constructs a document d by iteratively sampling words based on the following generative process (Blei, 2012):

1. Randomly choose a distribution over topics (θ_d) with hyperparameter α.
2. For each word n of the total N words in the document,
   (a) Randomly choose a topic ($z_{d,n}$) from the distribution over topics in step 1.
   (b) Randomly choose a word ($w_{d,n}$) from the corresponding distribution over the vocabulary β_k.

First, a random topic proportion for the document is initialised based on a Dirichlet distribution (step 1). From this distribution over all topics, one topic is sampled

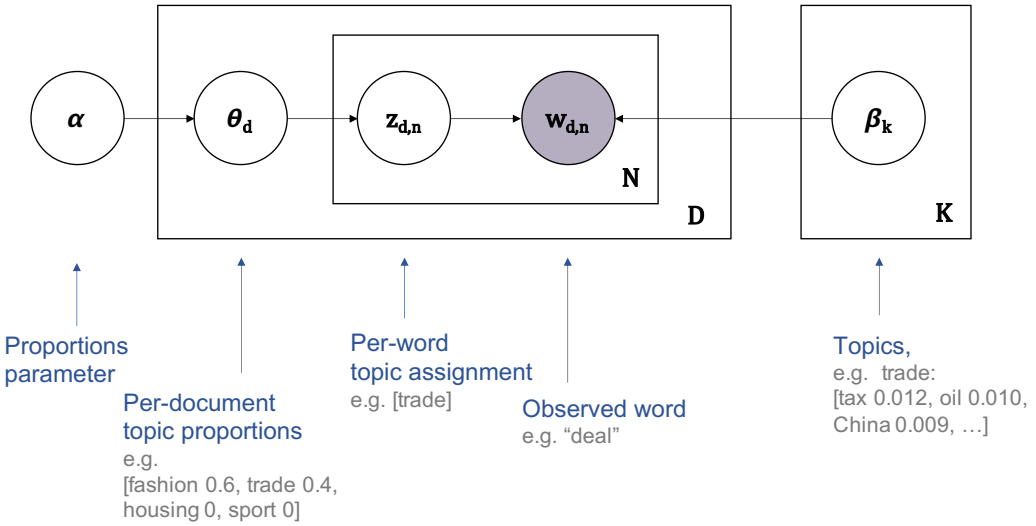


Figure 2.8: Graphical model representation of the generative process in LDA.

(step 2a). Based on this topic and its distribution over the vocabulary, a word is sampled (step 2b). Steps 2a and 2b are executed iteratively until the document generation is completed. In this process, the words within the documents can be directly observed, while the internal topic structure remains hidden. The graphical model representation visualises the connection between the LDA variables (see Figure 2.8). Words are modelled as observable variables ($w_{d,n}$) and hence shaded, while topics (β), topic-proportions for the dth document (θ_d) and topic assignments for the nth word in the dth document ($z_{d,n}$) are modelled through latent variables and not shaded. The generative process corresponds to the joint distribution of the hidden and the observable variables:

$$P(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{k=1}^{K} P(\beta_k) \prod_{d=1}^{D} P(\theta_d)(\prod_{n=1}^{N} P(z_{d,n}|\theta_d) P(w_{d,n}|\beta_{1:K}, z_{d,n})) \tag{2.6}$$

where K denotes the number of topics. However, in practice LDA's generative process is used reversed. Given a collection of existing documents, the aim is to infer the latent topic variables based on the observed variables. This amounts to computing the posterior distribution:

$$P(\beta_{1:K}, \theta_{1:D}, z_{1:D}|w_{1:D}) = \frac{P(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{P(w_{1:D})} \tag{2.7}$$

which can be approximated with sampling-based (e.g. Gibbs sampling) or varia-

tional approaches. Apart from the hidden and observable variables, LDA has a hyperparameter α (see Figure 2.8), which controls the per-document topic proportion. A high alpha value means that the documents are more likely to contain a mixture of all topics, while a low alpha means that documents are likely to contain only a few topics. The hyperparameter α and the number of topics K need to be provided to the algorithm, but can be tuned for example based on topic coherence scores (Röder et al., 2015). In this work, we use the Gensim package (Rehurek and Sojka, 2010) to train topic models and extract K-dimensional per-document topic distributions θ_d.

LDA extensions LDA has been successfully employed for a wide range of applications from text mining (Tong and Zhang, 2016) to bioinformatics (Liu et al., 2016), social science (Brauer et al., 2014) and beyond (Boyd-Graber et al., 2017). It has also inspired a wealth of subsequent work, such as dynamic topic models (Blei and Lafferty, 2006) which extend LDA by modelling the evolution of topics over time. Other studies have extended LDA by introducing hierarchical topic structure (HDP, Teh et al. 2006) or developing a supervised version of LDA (sLDA, Mcauliffe and Blei 2008).

2.3.4.2 GSDMM

Another line of work focuses on LDA's tendency to perform suboptimally for short texts because they only contain very limited co-occurence information (Hong and Davison, 2010; Vayansky and Kumar, 2020). Short texts are ubiquitous in social media due an informal mode of communication or platform restrictions (e.g. character limit for Tweets). An increased interest in clustering methods which are designed to work well for such texts has given rise to the research area of short text topic modelling (Jipeng et al., 2019). Yin and Wang (2014) proposed a Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model (GSDMM) which adjusts topic modelling for short texts by assuming that there exists only one topic per document, rather than a mixture of topics as in LDA. The model proposes a so-called movie group process and estimates the total number of topics based on an provided upper bound K.

We experiment with integrating LDA and GSDMM topic models into multiple neural architectures in Chapter 6.

2.4 Models

This section provides an overview of commonly used machine learning models for classification in the area of NLP, as this thesis focuses on classification problems. Popular baselines have used traditional classifiers such as Logistic Regression, Support Vector Machines and Random Forests. As this thesis aims at developing neural models, this section only discusses neural machine learning approaches. We begin by discussing Logistic Regression as a specific case of a neural network. Then, we review basic concepts and components in Feed-Forward Neural Networks. Building on these foundational neural network components, we introduce more sophisticated architectures, including Convolutional Neural Networks, Recurrent Neural Networks and Transformer models.

2.4.1 Logistic Regression

Logistic Regression (LR) is a binary classification model which given the dataset instance (x, y) predicts the probability $\hat{y}$ that the example represented by the feature vector $\mathbf{x}$ belongs to the class:

$$\hat{y} = P(y = 1|x) = \sigma(\sum_{i=1}^{n}(\mathbf{w_i x_i}) + b) \\ \mathbf{w}, \mathbf{x} \in \mathbb{R}^n, b \in \mathbb{R} \tag{2.8}$$

where b is the bias term and σ is the sigmoid (logistic) function. The sigmoid function is an S-shaped function which transforms the values into the range between 0 and 1 (Goldberg, 2016):

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.9}$$

LR can be understood as a one-layer neural network where the activation function is sigmoid (see Figure 2.9).

2.4.2 Feed-Forward Neural Network

A Feed-Forward Neural Network (FFN) is an artificial neural network in which one layer's output is provided as input to the next layer. An FFN consists of an input and an output layer with optional additional hidden layers between them. If the network has only one hidden layer (as shown in Figure 2.10) or very few layers it is considered 'shallow', while neural networks with a large number of hidden layers are

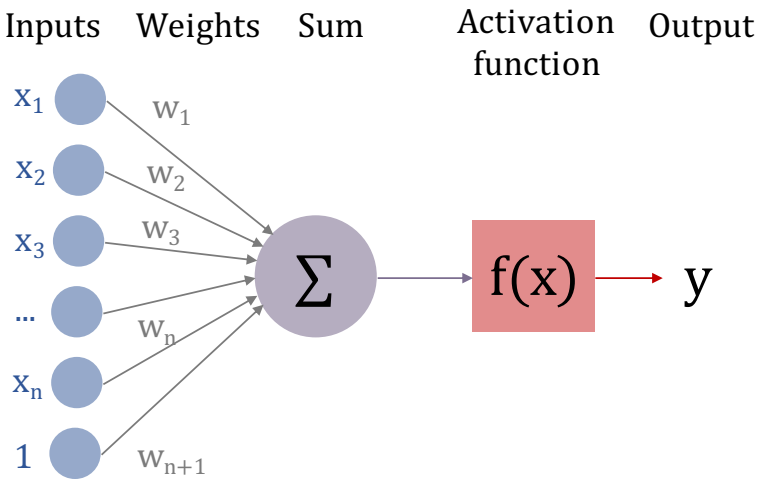


Figure 2.9: Architecture of a neuron.

considered 'deep'. Each layer is made up of multiple neurons where every neuron is connected to all neurons in the next layer and is hence referred to as dense or fully-connected layer (Goldberg, 2017).

Neuron A Neuron is a generic computation unit which takes n inputs and produces a single output (illustrated in Figure 2.9). Each neuron is associated with its own n-dimensional weight vector $\mathbf{w}$ and a bias term b. Given an n-dimensional input $\mathbf{x}$, the neuron performs a weighted summation of the input and its internal weights (which can be rewritten as a dot product), before applying an activation function f:

$$\begin{aligned} \hat{y} &= f(\sum_{i=1}^{n}(\mathbf{w_i x_i}) + b) \\ &= f(\mathbf{w} \cdot \mathbf{x} + b) \\ \mathbf{x} \in \mathbb{R}^n, &\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R} \end{aligned} \tag{2.10}$$

where $\cdot$ indicates the dot product. Common activation functions include sigmoid (see equation 2.9 above) and hyperbolic tangent (tanh) - an S-shaped function which transforms the values into the range between -1 and 1 (Goldberg, 2016):

$$\tanh(z) = \frac{2}{1 + e^{-2z}} - 1. \tag{2.11}$$

Dense layer Given an input $\mathbf{x}$ with *in* dimensions and a randomly initialised weight matrix $\mathbf{W}$ which consists of stacked up weights from its neurons, a fully connected layer performs a vector-matrix multiplication and applies an nonlinear activation function (Goldberg, 2017):

$$\begin{aligned} Dense(\mathbf{x}) &= f(\mathbf{x}\mathbf{W} + \mathbf{b}) \\ \mathbf{x} \in \mathbb{R}^{in}, \mathbf{W} &\in \mathbb{R}^{out \times in}, \mathbf{b} \in \mathbb{R}^{out} \end{aligned} \tag{2.12}$$

where *out* is the dimensionality of the next layer. An FFN with one hidden layer and an output layer performs the following computation:

$$\begin{aligned} FFN(\mathbf{x}) &= g(f(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2) \\ \mathbf{x} \in \mathbb{R}^{in}, \mathbf{W}^1 \in \mathbb{R}^{h \times in}, \mathbf{b}^1 \in \mathbb{R}^h, \mathbf{W}^2 &\in \mathbb{R}^{out \times h}, \mathbf{b}^2 \in \mathbb{R}^{out} \end{aligned} \tag{2.13}$$

where the weights $\mathbf{W}^1$, bias term $\mathbf{b}^1$ and activation function f belong to the h-dimensional hidden layer and the weights $\mathbf{W}^2$, bias term $\mathbf{b}^2$ and activation function g are associated with the *out*-dimensional output layer.

Training algorithm Training a neural network consists of two steps: forward propagation and backpropagation. During forward propagation, the neural network computes the predicted output $\hat{y}$ given the input $\mathbf{x}$ and a randomly initialised weight matrix $\mathbf{W}$. In the case of an FFN, $\hat{\mathbf{y}}$ is calculated using equation 2.13. During backpropagation, the loss E is calculated using a loss function L to capture the

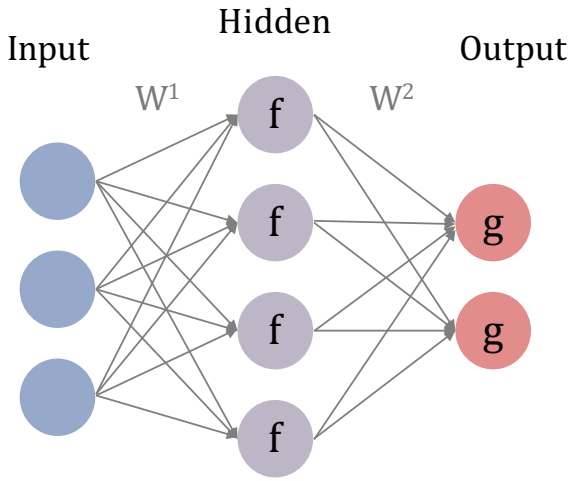


Figure 2.10: Architecture of a Feed-Forward Network with one hidden layer.

discrepancy between the predicted label and the gold label $\mathbf{y}$:

$$E = L(\mathbf{y}, \hat{\mathbf{y}}). \tag{2.14}$$

Based on the calculated loss, the weights of the network are learned through Stochastic Gradient Descent (SGD, see LeCun et al. 1998b):

$$\begin{aligned} \mathbf{W} = \mathbf{W} - \eta \frac{\partial E}{\partial \mathbf{W}}, \ \forall \ \mathbf{W} \in [\mathbf{W}^1, \mathbf{W}^2] \\ E \in \mathbb{R} \end{aligned} \tag{2.15}$$

where η denotes the learning rate and partial derivatives are calculated based on the chain rule. Weight updates are typically made after multiple training examples (referred to as minibatch) with the goal of learning better weights by iteratively minimising the loss.

This section focused on the core components and the basic training procedure for a simple neural network. Subsequent research has proposed multiple extensions of these concepts. For example, apart from sigmoid and tanh, other activation functions can be used, such as ReLu (Glorot et al., 2011) which clips values $x < 0$ at 0 and has been shown to work well despite its simplicity (Goldberg, 2016). Moreover, alternative optimisation algorithms to SGD have been proposed, such as Adadelta (Zeiler, 2012) and Adam (Kingma and Ba, 2017). The following sections build on the basic principles discussed here to introduce increasingly complex neural networks which are used in this thesis.

2.4.3 Convolutional Neural Network

Convolutional Neural Networks (CNN) are neural networks which consist of convolutional and pooling layers and are designed to extract certain local patterns from the input (Goldberg, 2017). Originally, CNNs were developed for Computer Vision, were the input consisted of matrices with greyscale values representing images. However, CNNs have also been successfully applied to NLP, where a typical input consists of concatenated word embeddings representing a sequence of words (see Figure 2.11). This section focuses on using CNNs for NLP, which has employed CNNs successfully as encoders and feature extractors.

Convolutional layer The idea behind a convolution is that a sliding window is moved over a given input matrix and applies a so-called filter to the values within

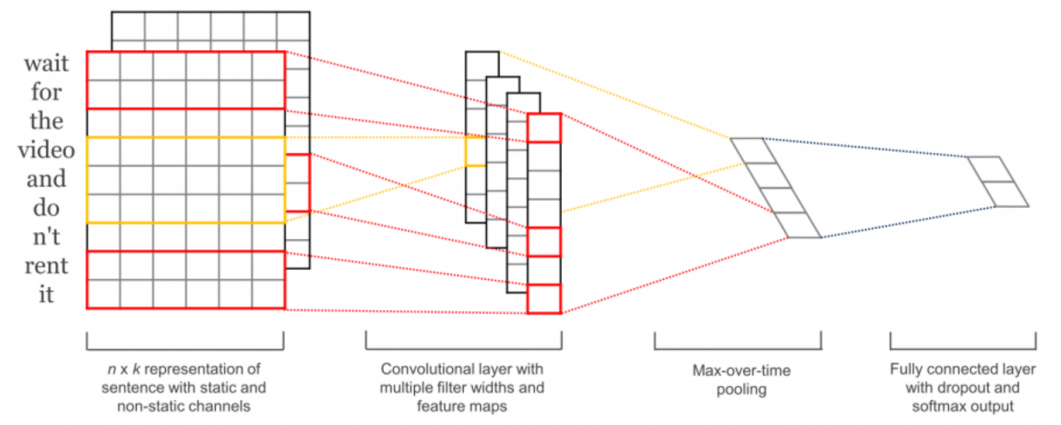


Figure 2.11: Architecture of a CNN for text classification (Kim, 2014).

the current window (see Figure 2.12). A convolutional filter is a function which performs an element-wise multiplication between the current input $\mathbf{x_i}$ and its filter weights $\mathbf{w}$, sums up the result and applies an activation function f to introduce non-linearity:

$$\begin{aligned} c &= f(\mathbf{w} \cdot \mathbf{x_i} + b) \\ c \in R, \mathbf{w}, \mathbf{x_i} &\in R^n, b \in R \end{aligned} \tag{2.16}$$

where $\cdot$ denotes a dot product and b is the bias term. Convolutions can be applied to inputs and filters of different dimensions. For example, in computer vision, a convolution may be performed on an image matrix, with an $n \times n$ filter resulting in a 2D convolution (see Figure 2.12 b). In CNNs for text processing, the input is typically a sentence matrix consisting of the embeddings corresponding to each word in the sentence, where the filter width is determined by the embedding dimension, resulting in a 1D convolution (see Figure 2.12 a). More specifically, given a sentence matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ representing a sentence with m words where each row corresponds to the distributed representation of a word, the dimensionality of the filter weights $\mathbf{w}$ is given by the number of words in the sliding window n and the dimensionality of the word embedding d. While the filter is gradually moved over $\mathbf{X}$, the notation $\mathbf{X}[i:j]$ indicates the sub-matrix from row i to j and the convolution is computed as follows (Kim, 2014):

$$\begin{aligned} c_i &= f(\mathbf{X}[i:i+n-1] \cdot \mathbf{w} + \mathbf{b}) \\ c_i \in R, \mathbf{w} \in \mathbb{R}^{n \cdot d}, \mathbf{X}[i:i+n-1] &\in \mathbb{R}^{n \cdot d}, b \in R \end{aligned} \tag{2.17}$$

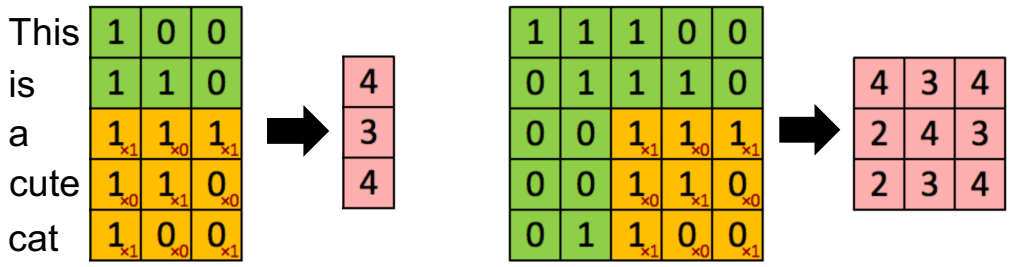


Figure 2.12: Typical convolutional operations with a 2D filter in computer vision and text classification. The input matrix is represented in green, filters in yellow and resulting feature maps in red. (a) shows a 1D convolution over a sentence matrix. (b) shows a 2D convolution over an image matrix.

where $\mathbf{c} \in \mathbb{R}^{s-h+1}$ indicates the resulting feature map. However, typically not just one but multiple filters are used:

$$\begin{aligned} \mathbf{c_i} &= f(\mathbf{X}[i:i+n-1]\mathbf{W} + \mathbf{b}) \\ \mathbf{c_i} \in R, \mathbf{W} \in \mathbb{R}^{l \times n \cdot d}, &\mathbf{X}[i:i+n-1] \in \mathbb{R}^{n \cdot d}, b \in \mathbb{R}^l \end{aligned} \tag{2.18}$$

where l indicates the number of filters. These filters start with different random initialisations and are updated during learning in order to detect distinct local patterns which are helpful for the task at hand. For example, in the case of a sentiment classifier this could be detecting positive or negative ngrams in a text.

Pooling layer Pooling is used to aggregate the outputs of feature maps from different convolutional filters by applying an aggregation function. Commonly used aggregation methods are max-pooling:

$$\hat{c} = \max_{1<i\leq m} (\mathbf{c_i}) \tag{2.19}$$

and average-pooling:

$$\hat{c} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{c_i} \tag{2.20}$$

Output layer After one or multiple CNN layers (which consist of a convolutional and a subsequent pooling layer), the output of the pooled feature maps are typically concatenated, before appying a final dense layer.

This section gave a brief overview on CNNs and their core components, but is by no means exhaustive, as many different CNN architectures and variations have been proposed. For text processing, this includes the use of filters with varying ngram sizes (Zhang and Wallace, 2015), the use of multiple channels for alternative representations (Kim, 2014) or and character-level convolutions (Zhang et al., 2015).

2.4.4 Recurrent Neural Network

Textual input is inherently sequential, as operational units consist of words (sequence of letters), sentences (sequence of words) or documents (sequence of sentences). The previously discussed models can encode such textual sequences in a single vector but have certain limitations: CBOW encodes the context of a given word, but doesn't capture word order. By comparison, CNNs are more sensitive to word order, but are limited to local patterns. In contrast, Recurrent Neural Networks (RNNs) are particularly suitable for modelling sequential inputs with arbitrary length and can capture word order over larger contexts.

2.4.4.1 Simple RNN

A Recurrent Neural Networks (RNN) is a type of artificial neural network which uses an internal hidden state to encode sequential inputs. Given a sequence of *in*-dimensional vectors $\mathbf{x_{i:n}} = (\mathbf{x_1}, ..., \mathbf{n})$, the RNN can encode the whole sequence in a single *out* -dimensional $\mathbf{y_n}$ (Goldberg, 2017):

$$\mathbf{y_n} = RNN(\mathbf{x_{1:n}}) \tag{2.21}$$

The output vector $\mathbf{y_n}$ can then be used for further prediction (e.g. by adding a dense layer). The hidden state is a vector which is used to keep track of previously seen inputs and updated at every time step. For every time step t of the sequential

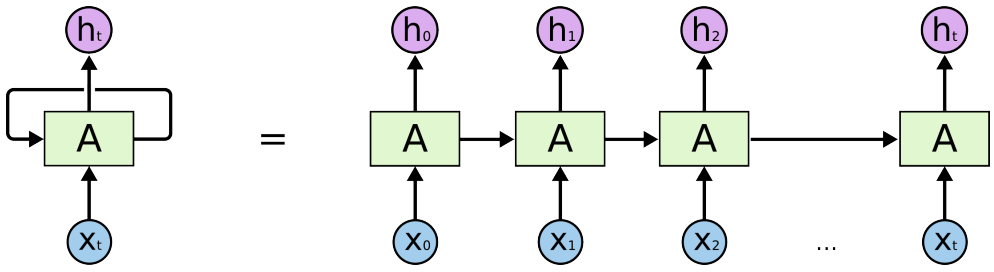


Figure 2.13: Recursive (left) and unrolled representation (right) of an RNN (Olah, 2015).

input, the RNN recursively calculates a new state vector $\mathbf{s_t}$ based on an input vector $\mathbf{x_t}$ and the previous state vector $\mathbf{s_{t-1}}$:

$$\mathbf{s_t} = RNN(\mathbf{x_t}, \mathbf{s_{t-1}}) \tag{2.22}$$

RNNs are often visualised recursively, as in the left part of Figure 2.13. This representation can be 'unrolled' as interconnected Feed-Forward Networks over the input sequence (right part of Figure 2.13). Given a sequence of input vectors $\mathbf{x_{i:n}} = (\mathbf{x_1}, ..., \mathbf{n})$, the forward pass in a simple RNN (Elman, 1990) is defined as follows:

$$\begin{aligned} \mathbf{s_t} = SRNN(\mathbf{x_t}, \mathbf{s_{t-1}}) &= f([\mathbf{s_{t-1}}; \mathbf{x_t}]\mathbf{W^h} + \mathbf{b^h}) \\ \mathbf{y_t} &= g(\mathbf{s_t}\mathbf{W^y} + \mathbf{b^y}) \\ \mathbf{x_t} \in \mathbb{R}^{in}, \mathbf{s_t} \in \mathbb{R}^{st}, \mathbf{y_t} \in \mathbb{R}^{out}, \mathbf{W^h} \in \mathbb{R}^{st\times(in+st)}, &\mathbf{b^h} \in \mathbb{R}^{st}, \mathbf{W^y} \in \mathbb{R}^{out\times st}, \mathbf{b^y} \in \mathbb{R}^{out} \end{aligned} \tag{2.23}$$

where $[;]$ indicates concatenation and f and g are activation functions. As can been seen in equation 2.23, the network shares the weight matrices $\mathbf{W^h}$ and $\mathbf{W^y}$ across all time steps, which makes it more efficient and enables it to operate on inputs with arbitrary length. While conceptually intuitive, simple RNNs are hard to train. Therefore, several variants with gated structures have been proposed, including Long-Short Term Memory Networks (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (Cho et al., 2014).

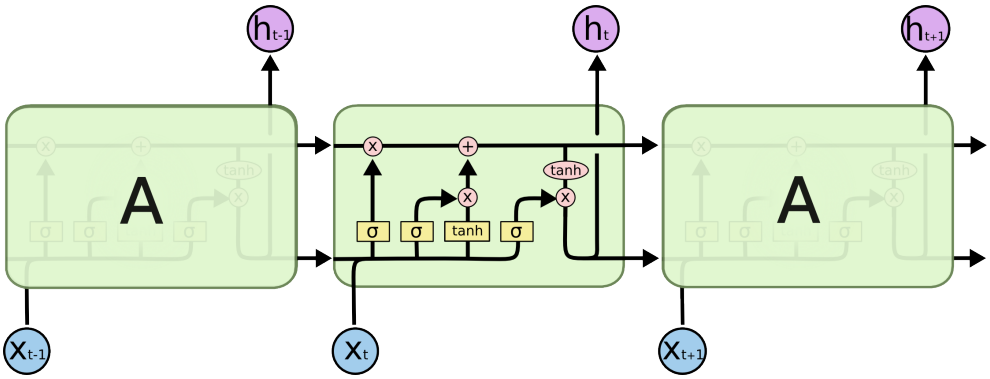


Figure 2.14: Architecture of an LSTM (Olah, 2015).

2.4.4.2 Long-Short Term Memory

Long-Short Term Memory networks (LSTMs, see Hochreiter and Schmidhuber 1997) were developed to address the vanishing gradient problem which occurs in simple RNNs. For this purpose, LSTMs have a cell state $\mathbf{c_t}$ in addition to the hidden state $\mathbf{h_t}$. At each time step t, gates are used to determine how much of the current input should be saved in the cell state and how much should be forgotten. Forward propagation through an LSTM is defined by the following calculations (Goldberg 2017, illustrated in Figure 2.14):

$$\begin{aligned}
\mathbf{s_t} = LSTM(\mathbf{s_{t-1}}, \mathbf{x_t}) &= [\mathbf{c_t}; \mathbf{h_t}] \\
\mathbf{c_t} &= \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{i} \odot \mathbf{z} \\
\mathbf{h_t} &= \mathbf{o} \odot \tanh(\mathbf{c_t}) \\
\mathbf{i} &= \sigma([\mathbf{x_t}; \mathbf{h_{t-1}}]\mathbf{W^i} + \mathbf{b^i}) \\
\mathbf{f} &= \sigma([\mathbf{x_t}; \mathbf{h_{t-1}}]\mathbf{W^f} + \mathbf{b^f}) \\
\mathbf{o} &= \sigma([\mathbf{x_t}; \mathbf{h_{t-1}}]\mathbf{W^o} + \mathbf{b^o}) \\
\mathbf{z} &= \tanh([\mathbf{x_t}; \mathbf{h_{t-1}}]\mathbf{W^z} + \mathbf{b^z}) \\
\mathbf{y_t} &= \mathbf{h_t} \\
\mathbf{s_t} \in \mathbb{R}^{2h}, \mathbf{x_i} \in \mathbb{R}^{in}, \mathbf{c_t}, \mathbf{h_t}, \mathbf{i}, \mathbf{f}, \mathbf{o}, \mathbf{z} \in \mathbb{R}^h, &\mathbf{W^i}, \mathbf{W^f}, \mathbf{W^o}, \mathbf{W^z} \in \mathbb{R}^{(in+h)\times h}, \\
\mathbf{b^i}, \mathbf{b^f}, \mathbf{b^o}, \mathbf{b^z} &\in \mathbb{R}^h
\end{aligned} \tag{2.24}$$

The information flow in the LSTM is controlled by multiple gates. A gate is a structure which consists of (1) a dense layer with sigmoid activation which computes a vector of gating values and (2) an element-wise multiplication operation to reweigh an input based on the calculated gating values. Gating values range between 0 and 1. If all values in the gating vector are 0, the gate is closed and no information can pass through. If all values are 1, the gate is fully opened and all information can pass freely. In an LSTM, there exist three gates: the input gate $\mathbf{i}$, the forget gate $\mathbf{f}$ and the output gate $\mathbf{o}$. At each time step, the gate values are determined based on linear combinations of the input $\mathbf{x_t}$ and the previous hidden state $\mathbf{h_{t-1}}$. Then, an update candidate $\mathbf{z}$ for the cell state is calculated based on a linear combination between $\mathbf{x_t}$ and $\mathbf{h_{t-1}}$ with tanh activation. The memory cell $\mathbf{c_t}$ is updated by controlling how much of the previous memory should be forgotten ($\mathbf{f} \odot \mathbf{c_{t-1}}$) and how much of the proposed new cell state should be saved ($\mathbf{i} \odot \mathbf{z}$). The output is the hidden state which is calculated based on a tanh activation of the new cell state and the output gate ($\mathbf{o} \odot \tanh(\mathbf{c_t})$).

Modifications Research has demonstrated that LSTMs are very effective for NLP, but they have some limitations: They are one-directional, relatively slow and complicated. Therefore, several modifications have been proposed. A standard LSTM calculates hidden states by feeding the sequential input from left to right. It only takes the past into account, resulting in a one-directional representation. However, it is possible to obtain an alternative representation with a so-called bidirectional LSTM (BiLSTM) which feeds the input from left to right $\overrightarrow{LSTM}(\mathbf{x_{1:t}})$ and right to left to the LSTM $\overleftarrow{LSTM}(\mathbf{x_{n:t}})$, before concatenating both resulting representations. The output of the BiLSTM at time step t is given by:

$$\text{BiLSTM}(\mathbf{x_{1:n}}, \mathbf{t}) = \mathbf{y_t} = [\overrightarrow{LSTM}(\mathbf{x_{1:t}}); \overleftarrow{LSTM}(\mathbf{x_{n:t}})] \tag{2.25}$$

This bi-directional approach takes both the past and future into account. Another drawback of the LSTM's is its complexity. This has been addressed by the GRU (Cho et al., 2014) which is a simplified version of the LSTM with fewer gates and has been shown to perform comparably with the LSTM (Chung et al., 2014).

2.4.5 Attention Mechanism

Attention is a popular neural network component which was originally proposed for GRUs (Bahdanau et al., 2015) but has since been successfully employed in LSTMs (Rocktäschel et al., 2016), CNNs (Rush et al., 2015; Yin et al., 2016) and Transformer models (see Chapter 2.4.6) with slight alterations. This section describes the core idea behind attention based on the Bahdanau et al. (2015) attention mechanism. Attention was invented to improve encoder-decoder architectures (Sutskever et al., 2014) which are commonly used for language generation tasks such as machine translation. Among other models, RNNs can be used in an encoder-decoder setup, where a first RNN with n hidden units encodes the input sequence into a single context vector $\mathbf{c}$

$$\begin{aligned} \mathbf{h_t^{enc}} &= RNN^{enc}(\mathbf{x_t}, \mathbf{h_{t-1}^{enc}}) \\ \mathbf{c} &= \mathbf{h_T^{enc}} \\ \mathbf{h_t^{enc}}, \mathbf{c} \in \mathbb{R}^n, \mathbf{x_t} &\in \mathbb{R}^m \end{aligned} \tag{2.26}$$

which is then passed to a second RNN that acts as decoder (2.15 a). The probablility to generate a word at step i is conditioned on the encoded source sequence $\mathbf{c}$, the previously generated word $\mathbf{y_{i-1}}$ and the hidden state of the decoder $\mathbf{h_i^{dec}}$:

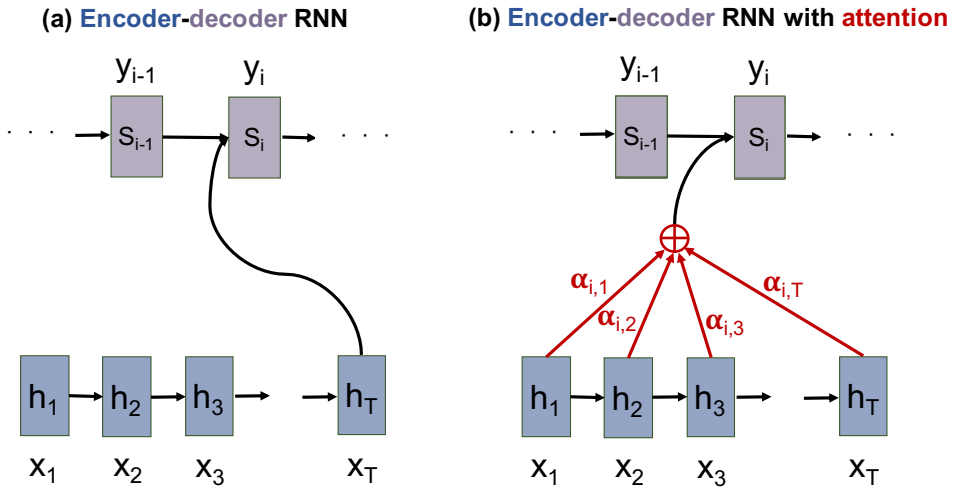


Figure 2.15: Architecture of an encoder-decoder RNN. The left part of the figure shows the model without attention and corresponds to equations 2.26 to 2.27. The right part of the figure shows an encoder-decoder RNN with attention (in red) corresponding to equations 2.28 to 2.30.

$$\begin{aligned} P(\mathbf{y_i}|(\mathbf{y_1}, ...\mathbf{y_{i-1}})) &= RNN^{dec}(\mathbf{y_{i-1}}, \mathbf{h_i^{dec}}, \mathbf{c}) \\ \mathbf{h_i^{dec}} &= RNN^{dec}(\mathbf{y_{i-1}}, \mathbf{h_{i-1}^{dec}}, \mathbf{c}) \\ \mathbf{h_i^{dec}}, \mathbf{c} &\in \mathbb{R}^n, \mathbf{y_i} \in \mathbb{R}^{|V|} \end{aligned} \tag{2.27}$$

The main bottleneck of this approach is that all information contained in the source sentence has to be encoded into a single vector before producing the target sentence. Here, the attention mechanism can help by allowing the decoder to pay varying degrees of attention to certain words in the source sentence depending on the current state of the encoder and the so far produced output. More specifically, rather than conditioning the decoder on a static $\mathbf{c}$ vector as in equation 2.27, Bahdanau et al. (2015)'s attention mechanism calculates a new context vector $\mathbf{c_i}$ at every time step i (Figure 2.15 b):

$$\begin{aligned} P(\mathbf{y_i}|(\mathbf{y_1}, ...\mathbf{y_{i-1}})) &= RNN^{dec}(\mathbf{y_{i-1}}, \mathbf{h_i^{dec}}, \mathbf{c_i}) \\ \mathbf{h_i} &= RNN^{dec}(\mathbf{y_{i-1}}, \mathbf{h_{i-1}^{dec}}, \mathbf{c_i}) \end{aligned} \tag{2.28}$$

as a weighted average of the encoder's hidden states based on attention weights α:

$$\mathbf{c_i} = \sum_{j=1}^{T} \alpha_{\mathbf{i,j}} \mathbf{h_j^{enc}}. \tag{2.29}$$

The attention weight $\alpha_{\mathbf{i,j}}$ is calculated by a Feed-Forward Neural Network with a softmax activation function between the hidden states of the encoder and decoder:

$$\begin{aligned} \alpha_{\mathbf{i,j}} &= \frac{\exp(\mathbf{e_{i,j}})}{\sum_{k=1}^{T} \exp(\mathbf{e_{i,j}})} \\ \mathbf{e_{i,j}} &= FFN(\mathbf{h_{i-1}^{dec}}, \mathbf{h_j^{enc}}). \end{aligned} \tag{2.30}$$

2.4.6 Transformer

Inspired by the success of attention in preceding models, the Transformer model (Vaswani et al., 2017) is a non-recurrent neural network which is primarily based on attention mechanisms. The Transformer was originally proposed as an encoder-decoder architecture (Figure 2.16 b), but can also be used as an encoder-only architecture (Figure 2.16 a) as in the popular BERT model (refer to section 2.3.3.2). This section concentrates mainly on describing the encoder-only setup as underlying architecture for BERT which is used and modified in subsequent chapters. The Transformer architecture consists of multiple blocks which employ multi-head attention directly to the input without recurrent connections. This reduces sequential dependencies in the model and enables more parallelisation.

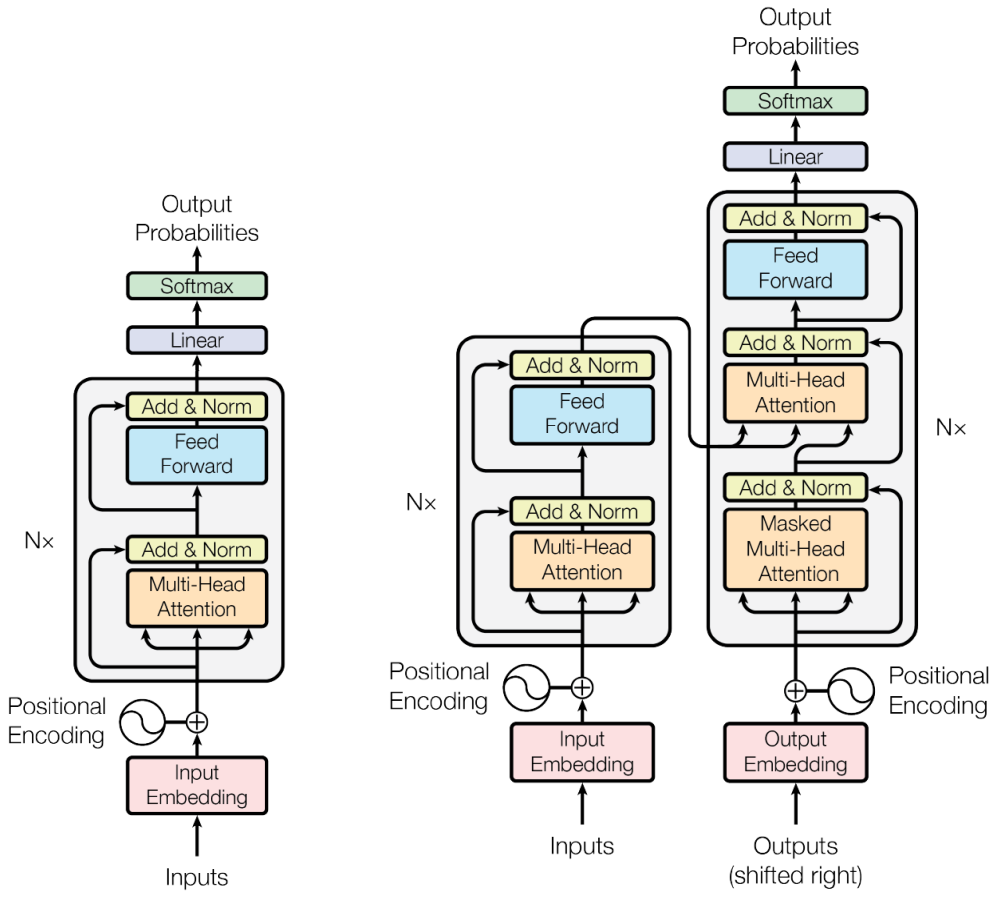
Encoding Layer Given a sequence of n embeddings $(\mathbf{w_1}, ..., \mathbf{w_n})$ with d dimension each of which is stacked into the matrix $\mathbf{E^W}$, the Transformer combines the input embeddings with a sequence of n d-dimensional positional embeddings $\mathbf{E^P} = (\mathbf{p_1}, ..., \mathbf{p_n})$, resulting in the following positionally aware representation of the input denoted by $\mathbf{H^0}$:

$$\begin{aligned} \mathbf{H^0} &= \text{LayerNorm}(\mathbf{E^W} + \mathbf{E^T}). \\ \mathbf{E^W}, \mathbf{E^T}, \mathbf{H^0} &\in \mathbb{R}^{n \times d} \end{aligned} \tag{2.31}$$

The positional embeddings are added to introduce information about sequence order since the Transformer is not a recurrent model and has otherwise no information regarding the order of its inputs. The positionally-aware input representation $\mathbf{H^0}$ is then passed on to the first Transformer block.

Transformer Block The Transformer architecture consists of L Transformer blocks.[2] Each transformer block i $(1 \leq i \leq L)$ consists of two layers: a multi-head attention layer and a feed-forward layer with residual connections and layer

[2]Vaswani et al. (2017) proposed an architecture consisting of 6 encoder blocks and 6 decoder blocks, while BERT_{BASE} uses 12 encoder blocks.



(a) Encoder Transformer (b) Encoder-Decoder Transformer

Figure 2.16: Architecture of the Transformer: (a) as encoder-only architecture as in BERT vs (b) encoder-decoder architecture. Figure adapted from Vaswani et al. (2017).

normalisation:

$$\begin{aligned} \mathbf{M^i} &= \text{LayerNorm}(\mathbf{H^{i-1}} + \text{MultiHeadAttention}(\mathbf{H^{i-1}}, \mathbf{H^{i-1}}, \mathbf{H^{i-1}})) \\ \mathbf{H^i} &= \text{LayerNorm}(\mathbf{M^i} + \text{FeedForward}(\mathbf{M^i})) \\ \mathbf{H_{i-1}}, \mathbf{H_i}, \mathbf{M_i} &\in \mathbb{R}^{n \times d} \end{aligned} \tag{2.32}$$

Each block calculates a new representation based on the output of the previous transformer block H^{i-1}. In the case of the first transformer block, the input is given by the encoding layer $\mathbf{E}$ in equation 2.31.

Multi-Head Attention Layer A multi-head attention layer has h attention heads which perform the following calculation:[3]

$$\begin{aligned} \text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= [head_1; ...; head_h]\mathbf{W}^O \\ \text{where } head_i &= \text{Attention}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) \\ \mathbf{W^O} &\in \mathbb{R}^{hv \times d} \end{aligned} \tag{2.33}$$

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are so-called queries, keys and values with dimensionalities k and v. These are placeholders for different inputs depending on the specific Transformer block. In the encoder, all queries, keys and values are the same input (self-attention). In the encoder-decoder setup, keys and values are given by the encoder representation, while queries come from the so far generated output (encoder-decoder attention). $\hat{\mathbf{Q}}$, $\hat{\mathbf{K}}$ and $\hat{\mathbf{V}}$ are obtained through matrix multiplication between the query, key and value and their associated weight matrices in head i:

$$\begin{aligned} \hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}} &= \mathbf{Q}\mathbf{W_i^Q}, \mathbf{K}\mathbf{W_i^K}, \mathbf{V}\mathbf{W_i^V} \\ \mathbf{W_i^Q} &\in \mathbb{R}^{d \times k}, \mathbf{W_i^K}, \mathbf{W_i^V} \in \mathbb{R}^{d \times v} \end{aligned} \tag{2.34}$$

which provide a different *view* on the respective $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ inputs. A single attention head then performs the following calculation:

$$\text{Attention}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) = \text{Softmax}(\frac{\hat{\mathbf{Q}}\hat{\mathbf{K}}^\mathbf{T}}{\sqrt{d_k}})\hat{\mathbf{V}}. \tag{2.35}$$

Similar to RNN attention (see section 2.4.5), $\hat{\mathbf{Q}}$ and $\hat{\mathbf{K}}$ are used to calculate a raw score which is normalised by a softmax function and used to reweigh $\hat{\mathbf{V}}$. The use of multiple attention heads could be compared to the use of multiple filters in a CNN which can specialise on recognising certain input patterns. The outputs of multiple attention heads are concatenated and passed through a linear transformation with the weight matrix $\mathbf{W^O}$ (compare equation 2.33). This serves the purpose of transforming the output of multiple attention heads with a dimensionality of $\mathbb{R}^{n \times h \cdot d}$ back to the original input dimensions $\mathbb{R}^{n \times d}$ and enables the stacking of Transformer layers.

[3] BERT_{BASE} uses $h = 12$ attention heads per layer with key and value dimensionality of $k = v = 64$.

Feed-Forward layer The point-wise feed-forward layer performs two linear transformations and a ReLu activation in between for each token in the sequence:

$$\begin{aligned} FFN(\mathbf{x}) &= ReLu(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})\mathbf{W^2} + \mathbf{b^2} \\ \mathbf{W^1} \in \mathbb{R}^{f \times d}, \mathbf{b^1} &\in \mathbb{R}^f, \mathbf{W^2} \in \mathbb{R}^{d \times f}, \mathbf{b^2} \in \mathbb{R}^d \end{aligned} \tag{2.36}$$

where f is the hidden dimension of the feed-forward layer. This has the benefit of being highly parallelisable.

This section discussed a broad range of neural architectures ranging from simple FFNs to complex Transformers. In this thesis, we combine CNNs with BiLSTMs in our model in Chapter 6.4. Moreover, we propose two Transformer-based models in Chapters 6.5 and 7.

2.5 Evaluation

2.5.1 Approach

This section discusses common evaluation techniques with a focus on machine learning models for classification. The evaluation process aims at assessing how well a given model performs on unseen data. The standard approach is to divide the data in a training, development and test set: The training set is used to train the model. The development set is used to tune different model settings and hyperparameter choices. The test set is a held-out portion of the data, which is only used to evaluate the final model.

2.5.2 Metrics

Standard evaluation metrics[4] are based on comparing the model predictions with the true gold labels to compute a single score which indicates model performance and can be used to rank different systems.

Confusion matrix A confusion matrix (Figure 2.17) categorises dataset instances by predicted label and true label. For binary classification, this results in four categories: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). TP have a positive gold label and were correctly predicted by the

[4]Following convention in the literature, we use the term evaluation metric in the sense of evaluation measure, rather than applying a strict metric definition which requires conditions such as symmetry and triangle inequality.

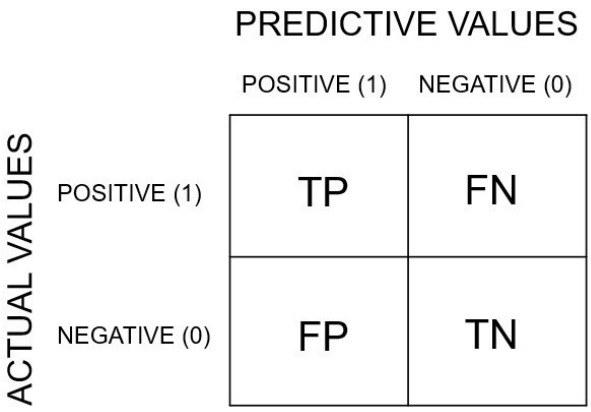


Figure 2.17: Confusion matrix for binary classification scenario.

model. Similarly, TN have a negative gold label and were correctly predicted by the model. In contrast, FP were predicted as positives, but belong the negative class and FN were predicted to be negatives, but belong to the positive class.

Accuracy reports the percentage of agreement between the true labels. In the binary classification scenario it is calculated as follows:

$$A = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.37}$$

Precision measures the portion of correctly identified positive cases out of all instances which were predicted to be positive:

$$P = \frac{TP}{TP + FP}. \tag{2.38}$$

Recall is the fraction of correctly identified positive cases out of all members of the positive class:

$$R = \frac{TP}{TP + FN}. \tag{2.39}$$

F1 combines recall and precision into a single figure:

$$F1 = 2\frac{P \cdot R}{P + R} = \frac{2TP}{2TP + FP + FN}. \tag{2.40}$$

For datasets with class imbalance F1 score is preferable over the other metrics listed above. Therefore, F1 is used as our main evaluation metric in Chapters 5 to 7. Recent work has demonstrated severe weaknesses of state-of-the-art models

despite achieving high performance on held-out datasets (Ribeiro et al., 2020). This highlights the need for more in-depth error analysis and evaluation. Therefore, we also employ a more challenging F1 score inspired metric which is introduced and discussed in Chapter 5.

CHAPTER 3

Semantic Similarity Detection: Related Work and a New Framework

This chapter provides an overview of the area of semantic similarity and defines key concepts. It first discusses an existing framework for addressing semantic similarity tasks in the form of Semantic Textual Similarity, which involves predicting graded relatedness scores for text pairs. Then, it proposes a novel more generic framework - Semantic Similarity Detection - which only requires binary classification labels and is applicable to many semantic similarity tasks and datasets without reannotation efforts. This is followed by a discussion of three core Semantic Similarity Detection tasks and previous methodological approaches. Finally, the connection between Semantic Similarity Detection and Community Question Answering is highlighted.

3.1 Semantic Similarity Detection: Overview and Tasks

Semantic Similarity Rus et al. (2013) define semantic similarity as quantifying and identifying the presence of semantic relations between two texts. Semantic similarity can be examined at different levels, such as words, phrases, sentences, paragraphs or documents (Rus, 2014). This thesis focuses on the similarity between short text snippets consisting of one or multiple sentences as this is of interest for many real-world applications. Traditionally, models for tasks involving semantic similarity have been developed separately, but Baudiš et al. (2016) argue that given the generic nature of neural approaches, various semantic similarity prediction tasks should be approached in a unified setup, rather than treated as isolated problems. One overarching framework for addressing semantic similarity across various tasks is semantic textual similarity (STS). The next section discusses STS and highlights

Score	Explanation
5	The two sentences are completely equivalent, as they mean the same thing.
4	The two sentences are mostly equivalent, but some unimportant details differ.
3	The two sentences are roughly equivalent, but some important information differs/missing.
2	The two sentences are not equivalent, but share some details.
1	The two sentences are not equivalent, but are on the same topic.
0	The two sentences are completely dissimilar.

Table 3.1: Explanation of Semantic Textual Similarity scale.

limitations, proposing Semantic Similarity Detection as a viable alternative.

Semantic Textual Similarity Semantic textual similarity (STS) captures the degree of semantic equivalence between two text snippets, ranging from exact semantic equivalence to complete unrelatedness with a graded score (Agirre et al., 2015). STS datasets have been constructed based on various existing datasets for tasks such as textual entailment, paraphrase detection and video captioning. The selected instances were then reannotated on a graded scale from 0 (no similarity) to 5 (complete semantic equivalence). While making fine-grain predictions on a graded scale can be useful, this comes at the cost of reannotating datasets which usually do not provide annotations compatible with the STS scale. With the rise of neural models, the size of semantic similarity datasets has increased rapidly (from MSRP with 5k+ examples in 2005 to Quora with 400k+ examples in 2017). Given the recent success of neural networks and their need for large amounts of training data, reannotating new large-scale datasets such as Quora is not practical. Similar to STS, this thesis aims to address closely related semantic similarity prediction tasks in a unified framework but relaxes the requirement for a graded STS annotation to make use of available large-scale datasets without reannotation efforts. To clearly distinguish this setup from STS, we use the term Semantic Similarity Detection which is defined in the following section.

Semantic Similarity Detection Semantic similarity arises from the presence of certain relations between two texts (Majumder et al., 2016; Rus et al., 2013). Depending on the specific use case, various sentence-level semantic relations have been proposed in the literature, ranging from as little as three relations (equivalence, contradiction and entailment) in Dagan et al. (2013) to five semantic similarity relations in Marsi and Krahmer (2007), to 18 relations in the CST Bank (Radev et al., 2004) or up to 24 relations in (Radev, 2000). This thesis focuses on three essential

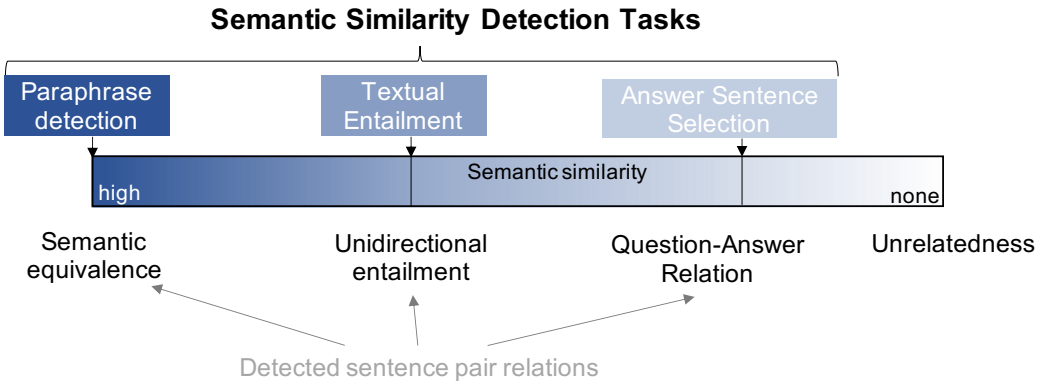


Figure 3.1: Scale for semantic similarity between sentence pairs in various Semantic Similarity Detection tasks.

semantic relations in the area of Natural Language Understanding (NLU) based on evidence provided in the literature: semantic equivalence (bidirectional entailment, McCarthy et al. 2009), unidirectional entailment (Ta˘tar et al., 2009) and question-answer relation (George, 2011; Benz and Jasinskaja, 2017). These relations are connected by the concept of semantic similarity and can be visualised on a scale of decreasing semantic similarity between text pairs (see Figure 3.1): starting from the highest degree of semantic similarity in the form of complete semantic equivalence in bidirectional entailment relations (e.g. strict paraphrases), decreasing to unidirectional entailment (e.g. in the case of loose paraphrases or textual entailment), followed by a certain semantic relatedness (e.g. between question-answer pairs) until dropping to complete unrelatedness. While early approaches have developed individual specialised models for related NLU tasks such as Paraphrase Detection, Textual Entailment and Answer Sentence Selection, a growing number of recent studies has proposed more generic approaches. Studies use varying terminology to refer to this problem, including text pair scoring (Baudiš et al., 2016), sentence pair modelling (Yin et al., 2016), natural language sentence matching (Wang et al., 2017) and text matching (Pang et al., 2016), but often do not offer a precise definition for these terms. Others (Adouane et al., 2019) use the name STS although predicting binary classes instead of the scaled STS scores. This thesis pursues a similar goal of developing generic approaches to model semantic similarity and uses the term Semantic Similarity Detection which in this thesis is defined as follows:

Semantic Similarity Detection refers to a collection of binary text pair classification tasks which aim at recognising the presence of a predefined semantic relationship between two given texts. This includes semantic equivalence, entailment and question-answer relations - which correspond to three well-known NLU tasks: paraphrase detection, Recognising Textual Entailment and Answer Sentence Selection.

Rather than qualifying the degree of semantic relatedness on a graded similarity scale (as in STS), Semantic Similarity Detection simply aims at detecting the presence of an underlying semantic relation in a binary classification setting. This is a more generic approach and opens up the use of any annotated semantic similarity dataset without the need for time-consuming manual reannotation. Existing graded or multi-class datasets can be easily converted into binary labels without manual effort. The following sections discuss three Semantic Similarity Detection tasks in more detail and review previously proposed approaches from the literature.

3.2 Paraphrase Detection

3.2.1 Definition

McCarthy et al. (2009) define paraphrasing as 'the restating of a sentence such that both sentences would generally be recognised as lexically and syntactically different while remaining semantically equal'. The three main criteria in this definition and the degree to which they apply in existing paraphrase detection datasets will be discussed in more detail below:

- (1) A strict definition of *semantic equivalence* requires bidirectional entailment between two sentences, with both sentences containing exactly the same amount of information, such as (Rus et al., 2014):
  - Text A: York had no problem with MTA's insisting the decision to shift funds had been within its legal rights.
  - Text B: York had no problem with MTA's saying the decision to shift funds was within its powers.

  However, in practice this definition is often relaxed to the looser semantic equivalence criterion of 'roughly equivalent' where one sentence may provide additional information (unidirectional entailment) (Rus et al., 2014):

  - Text A: Ricky Clemons brief, troubled Missouri basketball career is over.
  - Text B: Missouri kicked Ricky Clemons off its team, ending his troubled career there.

- (2) A second important criterion is *lexical variation*. Traditionally, a paraphrase is defined as the restating of the same meaning in different words. However, multiple studies (McCarthy et al., 2009; Rus et al., 2014) have pointed out that existing paraphrase detection datasets, such as MSRP (described in Chapter 4.2) do not meet this criterion due to high lexical overlap between sentence pairs.

- (3) The third criterion encompasses *syntactic variation* between the two sentences. Most paraphrase detection datasets do not specifically discuss this criterion. A notable exception is PAWS (described in Chapter 4.4) which focuses on creating adversarial paraphrase examples with high word overlap, but syntactic variation.

The definition of paraphrase detection or identification tasks usually does not emphasise lexical or syntactic diversity and focuses primarily on the semantic similarity criterion. Traditionally, paraphrase detection has focused mainly on detecting semantic equivalence between declarative sentences. However, recent research in the area of Community Question Answering (refer to section 3.6) has sparked interest in recognising question paraphrases with the aim of improving automatic question answering systems. This is a challenging new area of paraphrase detection, as Rodrigues et al. (2018) found paraphrase detection for interrogatives more difficult than for declaratives due to lower levels of lexical overlap. In this thesis, paraphrase detection is defined as a binary classification task to determine if two short texts - declaratives or interrogatives - have the same meaning (Vrbanec and Meštrović, 2020).

3.2.2 Approaches

Paraphrase detection is a core NLP task which has been widely studied. Previous studies have proposed various methods for identifying paraphrases which leverage lexical similarity (Zhang and Patrick, 2005; Abacha and Demner-Fushman, 2017), syntax (Filice et al., 2017; Feng et al., 2017), topics (Nandi et al., 2017; Xie et al., 2017) and metadata information (Xie et al., 2017; Wu et al., 2017a). The following sections provide an overview of previously proposed methods, categorising them into feature-engineered, neural and hybrid approaches (compare Table 3.2). Most of the

reviewed studies have focused on the MSRP dataset (described in Chapter 4.2), the SemEval 2017 CQA dataset Task B (described in Chapter 4.5) or the Quora question pair dataset (described in Chapter 4.3), which are also used in this thesis.

Feature-engineered methods Unsupervised metrics have been very popular in *early paraphrase detection approaches*. For example, Fernando and Stevenson (2008) compared six existing metrics for calculating word similarities based on WordNet with the intuition of exploiting synonyms and other semantic relations for paraphrase detection. They used a similarity matrix approach and determined a classification threshold based on the training data. Other studies repurposed existing machine translation metrics and leveraged other lexical similarity features to identify paraphrases. For example, Madnani et al. (2012) successfully combined eight different machine translation metrics within a meta classifier. Nguyen-Son et al. (2015) proposed a new metric based on identifying identical phrases and similar words which discards less important words. They combined their SimMat metric with eight standard machine translation metrics in a logistic regression model. Zhang and Patrick (2005) proposed transforming text with a similar meaning into a similar surface form using transformational grammar. They then applied a decision tree with various lexical similarity features to the transformations to identify paraphrases. *More recent feature-engineered methods* concentrated on paraphrase detection within several shared SemEval Community Question Answering (CQA, refer to section 3.6) tasks, exploring different features to identify question paraphrases through textual similarity measures, topic modelling, lexical and syntactic similarity and translation approaches (Nakov et al., 2017). Many participants engineered a variety of elaborate features for their SVM or logistic regression models. The winning system (based on MAP of primary submission) by Charlet and Damnati (2017) was a supervised combination of unsupervised textual similarity measures including soft-cosine similarities (based on generic Wikipedia and dataset-specific embeddings) and edit distance between posts. Another competitive approach by Filice et al. (2017) proposed a tree-kernel SVM combining syntactic (e.g. similarity between part of speech tag n-grams, parse trees), semantic (e.g. word embedding cosine similarity), lexical (e.g. longest common substring, Jaccard index, greedy string tiling) and ranking (e.g. search engine ranking) features. The provided IR baseline from the dataset construction stage was a frequently used feature across many systems (Galbraith et al., 2017; Filice et al., 2017). CQA-specific metadata was also commonly exploited, such as the position of the post in the thread (Xie et al., 2017) or the author of the post (Nandi et al., 2017). However, these fea-

Publication	Type	Approach	Dataset
Zhang and Patrick (2005)	FE	Decision tree with lexical similarity features of canonicalised text	MSRP
Fernando and Stevenson (2008)	FE	Metric using wordnet similarities calculated with JCN	MSRP
Madnani et al. (2012)	FE	SVM, LR with machine translation metrics and meta classifier	MSRP
Ji and Eisenstein (2013)	FE	Discriminative term-weighting metric (TF-KLD)	MSRP
Nguyen-Son et al. (2015)	FE	LR with SimMat metric and machine translation metrics	MSRP
Abacha and Demner-Fushman (2017)	FE	LR with lexical and morpho-syntactic features	SemEval B
Agustian and Takamura (2017)	FE	SVM with word importance and similarity features	SemEval B
Charlet and Damnati (2017)	FE	LR with semantic textual similarity measure features	SemEval B
Filice et al. (2017)	FE	SVM with syntactic, semantic, lexical, ranking features	SemEval B
Galbraith et al. (2017)	FE	Ensemble of syntactic, semantic, and IR features	SemEval B
Nandi et al. (2017)	FE	SVM with string similarity, word embedding, topic model, domain, dialogue, keyword features	SemEval B
Wu et al. (2017a)	FE	LR with lexical,topic model, search engine and metadata features	SemEval B
Xie et al. (2017)	FE	LR, SVM with semantic vectors and metadata features	SemEval B
Feng et al. (2017)	H	BiLSTM with word embedding cosine distance, lexical, syntactic, ranking, translation features	SemEval B
Qi et al. (2017)	H	Eependency-based keyword extraction, followed by CNN	SemEval B
Socher et al. (2011)	NN	Autoencoder with pooling	MSRP
He et al. (2015)	NN	CNN with pooling & similarity metrics	MSRP
Yin and Schütze (2015)	NN	CNN with attention	MSRP
Tomar et al. (2017)	NN	Pretrained FFN as encoder and attention components	Quora

FE = feature engineering, NN = neural networks, H=hybrid, LR = Logistic Regression

Table 3.2: Overview of paraphrase detection approaches. Systems are ordered by type, dataset, publication year and author.

tures are very dataset-specific and cannot easily be transferred to other tasks and datasets. Other successful but more generic features were topic-based features, such as the presence of topic key words or topic-based distances between posts (Nandi et al., 2017).

Hybrid methods Some studies experimented with hybrid approaches which combined neural networks with manually engineered features. In the SemEval 2017 challenge, Qi et al. (2017) extracted keywords from a weighted graph which they built based on syntactic dependency relations. These keywords were then fed into a complex CNN-based architecture. Furthermore, Feng et al. (2017) experimented with a combination of lexical (e.g. word overlap, tf-idf, longest common subsequence), syntactic (e.g. tree kernels), ranking (IR baseline ranking), translation (translation probability) features with neural components (BiLSTM encoder, similarity matrix, convolutional and pooling layers). However, they abandoned the neural components based on the results of an ablation study which indicated that these deteriorated results. Generally, hybrid architectures could not outperform the best feature-engineered paraphrase detection systems and participants identified the small dataset size as the main obstacle to training neural components.

Neural methods Neural network approaches can be classified based on the choice of encoder. Socher et al. (2011) used recursive autoencoders to encode sentences and computed a similarity matrix followed by dynamic pooling. Inspired by the success in computer vision, CNN-encoders have become popular text encoders in NLP and are frequently used for paraphrase detection due to the fewer parameters required in comparison to RNNs. Yin and Schütze (2015) presented a CNN-based paraphrase detection architecture consisting of multiple convolutions, averaging and pooling layers. Concurrently, He et al. (2015) proposed a CNN-based encoder followed by pooling and comparison layers. In order to reduce model complexity further, Tomar et al. (2017) adapted the popular Decomposable Attention Model (Parikh et al., 2016) which was originally proposed for textual entailment and requires fewer parameters through the use of Feed-Forward Networks. The authors extended the Decomposable Attention Model through the use of n-gram embeddings and noisy pre-training.

3.3 Textual Entailment

3.3.1 Definition

Dagan et al. (2013) defines textual entailment as a directional relationship between pairs of text expressions: Given the entailing Text T and the entailed Hypothesis H, we say that T entails H if humans reading T would typically infer that H is most likely true. The task of Recognising Textual Entailment (RTE) requires to predict entailment or non-entailment between two text snippets automatically. The unidirectional property of entailment relations is highlighted in the example below Dagan et al. (2013):

- T: The drugs that slow down or halt Alzheimer's disease work best the earlier you administer them.
- H: Alzheimer's disease is treated using drugs.

where information about halting or stopping the disease in T entails treating the disease in H, but not vice versa. The original RTE task setup distinguishes only between entailment and non-entailment and is therefore a binary classification task. Later research added a three-way entailment classification setup which is commonly used in more recent natural language inference (NLI) research and requires a distinction into entailment, contradiction and a neutral class. This thesis focuses on the binary textual entailment setup as Semantic Similarity Detection focuses on identifying the presence of certain semantic relations, which is more generic.

3.3.2 Approaches

Table 3.3 provides an overview of previously proposed methods for Recognising Textual Entailment. While we have not worked on entailment for the scope of this thesis, it is captured by the proposed framework. We provide a literature review for textual entailment below because many paraphrase detection and Answer Selection approaches are based on entailment models. Moreover, multiple methods have addressed all three tasks together to inform the development of new generic models (section 3.5).

Feature-based models Previous studies have explored various lexical, semantic and syntactic features to recognise textual entailment. For example, lexical similarity between text and hypothesis was exploited by Perez and Alfonseca (2005) based

on BLEU and by Adams et al. (2007) using bag-of-word similarity features. Syntactic similarity features were used by Kouylekov and Magnini (2007) who calculated the tree edit distance between syntactic trees of the text and the hypothesis. Raina et al. (2005) proposed a graph matching method based on dependency trees. Jijkoun and de Rijke (2005) combined word similarity measures based on dependencies and lexical chains in WordNet with frequency-based term weighting. Iftene (2008) used rules and several semantic resources to map the text to the hypothesis, calculating global fitness values to distinguish between different entailment classes. Mehdad et al. (2010) proposed syntactic semantic tree kernels in combination with features from WordNet, Levenshtein distance and idf.

Neural models LSTMs have been very popular for addressing textual entailment due to their ability to capture long-range dependencies. Bowman et al. (2015) proposed an LSTM-based entailment model along with the popular SNLI dataset. Rocktäschel et al. (2016) further extended an LSTM encoder by adding word-to-word attention between a premise and a hypothesis. Moreover, Chen et al. (2017) proposed a sequential inference model based on tree-LSTMs, harnessing syntactic

Publication	Type	Approach	Dataset
Jijkoun and de Rijke (2005)	FE	Lexical similarity measures	PASCAL-RTE
Perez and Alfonseca (2005)	FE	Lexical similarity based on BLEU	PASCAL-RTE
Raina et al. (2005)	FE	Graph matching between dependency trees	PASCAL-RTE
Adams et al. (2007)	FE	Decision tree with lexical overlap features	PASCAL-RTE
Kouylekov and Magnini (2007)	FE	Tree edit distance	PASCAL-RTE
Iftene (2008)	FE		PASCAL-RTE
Bowman et al. (2015)	NN	LSTM	SNLI
Rocktäschel et al. (2016)	NN	LSTM with word-by-word attention	SNLI
Parikh et al. (2016)	NN	FFN with decomposable attention	SNLI
Chen et al. (2017)	NN	Tree-LSTMs based on syntax trees	SNLI

T=transformation-based, FE = feature engineering, NN = neural networks, H=hybrid, LR = Logistic Regression

Table 3.3: Overview of task-specific approaches to textual entailment. Systems are ordered by type, publication year and author.

parsing information. However, despite their effectiveness for textual entailment, LSTMs require a large number of parameters which makes them expensive to train. Therefore, Parikh et al. (2016) proposed to decompose the natural language inference problem into three steps: attend, compare and aggregate, which are modelled with multiple - less expensive - Feed-Forward Networks. In another line of work, Ghaeini et al. (2018) focused on the interpretability of neural entailment models.

3.4 Answer Sentence Selection

3.4.1 Definition

Answer Sentence Selection (or: Answer Selection) refers to the task of detecting sentences which contain the correct answer among different sentence candidates (Yu et al., 2014) and is a core component of a typical automatic question answering system (Lai et al., 2018). Answer Sentence Selection is closely related to but distinct from the task of answer extraction which - as a final step of factoid question answering systems - aims to identify the correct answer span to a question within a given sentence. For example, for the question 'Who established the Nobel Prize?' with the following possible answers:

1. 'The Nobel Prize was established more than 100 years ago.'
2. 'The Fields Medal, established in 1936, is often described as the Nobel Prize of mathematics.'
3. 'The Nobel Prize was established in the will of Alfred Nobel.'

an Answer Selection system should select the third sentence as correct (Lai et al., 2018), while an answer extraction system would be required to select the text span 'Alfred Nobel' from the sentence. Traditionally, Answer Sentence Selection research has focused primarily on *factoid questions* as illustrated in the example above. Such questions ask for a single fact or piece of information and have been the primary focus of popular Answer Selection datasets such as TrecQA (Wang et al., 2007), WikiQA (Yang et al., 2015) and Natural Questions corpus (Kwiatkowski et al., 2019). In contrast, *non-factoid questions* ask for reasons, opinions, suggestions or interpretations. (Dulceanu et al., 2018). Automatically answering such questions is more challenging due to the increased complexity and length of both questions and answers, as well as less text overlap between them (Cohen and Croft, 2016). Another difficulty is subjectivity, as there is not always one objectively correct answer. Datasets for non-factoid Answer Selection have only recently become available, such

as InsuranceQA (Feng et al., 2015), the SemEval CQA dataset (Nakov et al., 2017) and PhotoshopQuiA (Dulceanu et al., 2018). These datasets were collected from online question answering communities which adds the challenge of informal language use. An example for non-factoid Answer Sentence Selection from the SemEval 2017 dataset is shown below (Lai et al., 2018):

- Question: 'Hi;Can any one tell me a place where i can have a good massage drom philipinies????? yesterday i had a massage in Bio-Bil they charged me 300qr for 01 hour bt it is totally waste... pls advice me if theres any philipinos....'
- Positive Answer: 'Try Magic Touch in Abu Hamour (beside Abu Hamour Petrol Stn)it will just cost you 60QR per hour and I've seen a lot of Qataris as their customers.'
- Negative Answer: 'I dont know the name; you can call them. Do it fast; they have sooooo many reservations ;)'

Furthermore, there exist two major setups for Answer Sentence Selection: as a ranking and as a classification task. In the ranking scenario, given a question and a list of possible answers, the goal is to rerank the answers so that more relevant answers are ranked above less relevant ones. In the classification scenario, given a question and candidate answer, the goal is to predict whether the candidate answer provides the answer to the question (Feng et al., 2015). Typically, the relevance of an answer for a question is determined based on the semantic similarity between question and answer. Within the Semantic Similarity Detection framework, the classification setup is preferred as it is more generic.

3.4.2 Approaches

Previous studies have proposed various approaches to Answer Sentence Selection, leveraging lexical similarity (Yang et al., 2015; Nandi et al., 2017), semantic similarity (Koreeda et al., 2017) syntactic similarity (Punyakanok et al., 2004), and metadata (Filice et al., 2017). Table 3.4 provides an overview of previous approaches to Answer Selection, distinguishing between feature-engineered, hybrid and neural methods which are discussed in more detail below. While there exist a wide range of Answer Selection datasets, this thesis describes SemEval A and C (compare Chapter 4.5) and uses them for experiments. We chose to focus on these datasets because they are among the most commonly used non-factoid Answer Selection datasets.

Publication	Type	Approach	Dataset
Punyakanok et al. (2004)	FE	Approximate tree matching based on syntactic and semantic features	Trec
Wang et al. (2007)	FE	Scoring of alignment between question and answer after syntactic transformations	Trec
Severyn and Moschitti (2013)	FE	Tree-kernel SVM with lexical and structural features	Trec
Filice et al. (2017)	FE	SVM with syntactic, semantic, lexical, ranking, metadata features	SemEval A&C
Nandi et al. (2017)	FE	SVM with string similarity, word embedding, topic model, domain, dialogue, keyword, stacking features	SemEval A&C
Xie et al. (2017)	FE	LR, SVM with semantic vectors and metadata features	SemEval A&C
Yang et al. (2015)	H	CNN with lexical features	QASent, WikiQA
Bonadiman et al. (2017)	H	Multitask CNN with ranking feature	SemEval C
Feng et al. (2017)	H	BiLSTM with word embedding cosine distance, lexical, syntactic, ranking, translation features	SemEval A
Koreeda et al. (2017)	H	Decomposable attention model with comment plausibility, answer adequacy and meta data features	SemEval A&C
Wu et al. (2017a)	H	CNN/Adaboost with lexical,topic model, search engine and metadata features	SemEval A&C
Yu et al. (2014)	NN	CNN with convolution, pooling and summing layer	TrecQA
Feng et al. (2015)	NN	CNN with hidden, convolution, pooling and similarity scoring layer	InsuranceQA
Tan et al. (2016)	NN	LSTM,CNN with attention	InsuranceQA, TrecQA
Deriu and Cieliebak (2017)	NN	Siamese CNN with attention	SemEval A
Garg et al. (2019)	NN	Transformer	Natural Questions

T=transformation-based, FE = feature engineering, NN = neural networks, H=hybrid, LR = Logistic Regression

Table 3.4: Overview of task-specific approaches to answer selection. Systems are ordered by type, publication year and author.

Feature-engineered methods *Early Answer Selection models* predominately relied on syntactic matching between parse trees (Yu et al., 2014). One approach is to transform answers into questions as in Wang et al. (2007)'s translation-inspired model. Another option is to extract features from syntactic parse trees for approximate tree matching as in Punyakanok et al. (2004). Severyn and Moschitti (2013) proposed another feature-engineered approach based on syntactic parse trees, which is further enriched with additional semantic features. Many *recent feature-engineered systems* have focused on Answer Sentence Selection in the context of the SemEval 2017 CQA shared task (Nakov et al., 2017). The competition featured two Answer Selection scenarios: internal Answer Selection (subtask A), which aims to identify useful comments within the same thread and external Answer Selection (subtask C) which requires detecting helpful replies originating from a different thread than the question. Systems in the former subtask received significantly higher scores than in the latter, indicating that this is an easier task. Participating systems used a range of approaches including matching syntactic structures to candidate answers, tree edit distances, machine translation models, answer thread related information, neural networks (LSTMs, CNNs) and Integer Linear Programming (Nakov et al., 2017). The winning system for subtask A was by Filice et al. (2016) who explored the stacking of different classifiers and metadata features (e.g. length, forum category) in addition to their lexical, semantic and syntactic features from subtask B (compare section 3.2). Most systems tackled subtask C by combining their subsystems from subtask A and B (the Paraphrase Detection subtask, refer to section 3.2). The winner of subtask C was a heavily engineered feature-based approach by Nandi et al. (2017) which combined string similarity, word embedding, topic modelling, domain, dialogue and keyword features.[1] Xie et al. (2017) employed SVM and logistic regression models with multiple semantic similarity (e.g. word embeddings, syntax-based similarity, topic similarity) and metadata features (e.g. position of the post in the thread, presence of links, author of post).

Hybrid methods Multiple hybrid approaches combining features and neural components have been proposed for the SemEval task 2017. Bonadiman et al. (2017) presented a multitask learning approach by mutually training the same network architecture for the three tasks before combining it with the IR ranking feature. Feng et al. (2017) proposed a model combining lexical (e.g. word overlap, tf-idf, longest common subsequence), syntactic (e.g. tree kernels), ranking (IR baseline ranking),

[1]The system is slightly different from the paraphrase detection system proposed in section 3.2 and therefore treated as a task-specific system rather than a generic approach.

translation (translation probability) features with a BiLSTM encoder, followed by a similarity matrix, convolutional and pooling layers. In contrast to their paraphrase model, the final Answer Selection model made use of all neural components. Koreeda et al. (2017) combined the Decomposable Attention Model (Parikh et al., 2016) with comment plausibility, answer adequacy and metadata features. Wu et al. (2017a) combined word matching, topic model, search engine, metadata features with a CNN for subtask A and with Adaboost for subtask C. Hybrid models have been proposed outside of the SemEval task as well. For example, Yang et al. (2015) released the popular WikiQA dataset along with several baselines (including Paragraph Vectors, CNNs and hybrid modifications). Among these, they found a CNN combined with lexical features (word and lemma matching) worked best.

Neural methods Many recently proposed Answer Selection models have used purely neural architectures. As the only purely neural participating system in SemEval 2017, Deriu and Cieliebak (2017) applied the attention-based CNN model proposed by Yin et al. (2016) (refer to section 3.1). They achieved medium success for the internal Answer Selection task, but poor results on the external Answer Selection task, indicating room for improvement for neural models on this dataset. Yu et al. (2014) presented a model consisting of a convolutional, pooling and a final summing layer to model n-gram similarities between question and the answer candidates. Feng et al. (2015) modelled the similarity between answer and question with a joint hidden layer and convolutional layer, followed by separate pooling and activation layers. Finally, they calculate a similarity score between the two representations with a custom similarity function based on Euclidean similarity and sigmoid dot product. Tan et al. (2016) encoded the question and answer with two separate bi-LSTM-CNN encoders and applied attention before calculating a cosine similarity score between the resulting representations. Garg et al. (2019) explored BERT fine-tuning approaches for Answer Selection by transferring the pre-trained model for a general task before adapting it to the target domain, highlighting the need for domain adaptation in CQA transfer learning.

3.5 Generic Approaches to Semantic Similarity Detection

The previous sections defined and discussed the individual tasks of Paraphrase Detection, Recognising Textual Entailment and Answer Sentence Selection. In section 3.1, we argued that these tasks are closely connected based on the concept of se-

mantic similarity and proposed the encompassing framework of Semantic Similarity Detection. In sections 3.2 to 3.4, we discussed several previously proposed models for these tasks. Across these tasks, similar information sources (lexical similarity, syntax, semantics and metadata) were exploited and similar technical methods (feature-engineered, neural, hybrid) were employed. This suggests that it is reasonable to develop generic approaches for these tasks, rather than separate task-specific models. Generic models are desirable because they avoid the 'reinvention' of highly similar models for closely related tasks which saves time and resources in the long run. We define generic approaches as methods which can be successfully applied to multiple Semantic Similarity Detection tasks without requiring adjustments in features, architecture or training method. More recently, a number of such generic methods have been proposed, ranging from simple embedding averaging to complex pre-trained neural networks which are summarised in Table 3.5 and will be discussed below.

Simple embedding-based methods In contrast to many recently proposed complex architectures, one line of work has explored simple embedding averaging approaches which do not require any computationally expensive encoders. For example, Shen et al. (2018b) introduced a simple word embedding model without encoders, by summing, averaging or concatenating word embeddings. In a similar approach, Wieting et al. (2016) proposed a simple word vector averaging-based approach and demonstrated similar results to LSTM-based methods for training general-purpose paraphrastic sentence embeddings for semantic textual similarity tasks.

Hybrid approaches While hybrid approaches which combine features and neural architectures are popular for task-specific models, relatively few such models have been proposed for semantic similarity detection as many task-specific features do not generalise well. Therefore, hybrid Semantic Similarity Detection models tend to focus on generic features, such as syntactic or lexical ones. Gong et al. (2018) integrated lexical (as exact match) and syntactic features (in the form of part-of-speech tags) in their neural architecture consisting of a two-layer highway network with attention, followed by a similarity matrix and CNN feature extractor in their DIIN model. Wu et al. (2018) also incorporated syntactic information in a self-attention network which leverages separate syntactic blocks for encoding phrase structure. Moreover, Gupta and Zhang (2018) proposed an attention-enriched tree-LSTMs on the basis of constituent and dependency trees.

Publication	Type	Approach	Paraphrase Detection	Answer Selection	Textual Entailment
Shen et al. (2018b)	S	Averaging, summing and concatenating word embeddings	Quora, MSRP	WikiQA, Trec	SNLI, MultiNLI
Gong et al. (2018)	H	Highway network, self-attention, CNN feature extractor with syntactic and exact match features	Quora		SNLI, MultiNLI
Gupta and Zhang (2018)	H	Tree-LSTM with attention	MSRP, Quora		
Wu et al. (2018)	H	Self-attention network with syntactic features	MSRP		SNLI
Pang et al. (2016)	NN	Matching matrix followed by CNN feature extractor	MSRP		
Wang and Jiang (2016)	NN	LSTM with attention, followed by CNN		WikiQA	SNLI
Yin et al. (2016)	NN	CNN with attention	MSRP	WikiQA	
Conneau et al. (2017)	NN	InferSent: BiLSTM with maxpooling	MSRP		SNLI
Wang et al. (2017)	NN	BiLSTM, matching, aggregation layer	Quora	WikiQA, TrecQA	SNLI
Kovaleva et al. (2018)		Autoencoder	MSRP		SNLI
Shen et al. (2018a)	NN	CNN with context-aware filters	Quora	WikiQA, SelQA	
Rao et al. (2019)	NN	CNN-LSTM with co-attention	Quora	TrecQA	
Tay et al. (2019)	NN	Quaternion Transformer	Quora	WikiQA	SNLI, MNLI
Peters et al. (2018)	P	ELMo	Quora		SNLI
Devlin et al. (2019)	P	BERT	MSRP, Quora	QNLI	MNLI
Reimers and Gurevych (2019)	P	BERT with siamese and triplet network structures	MSRP	Trec	
Wang et al. (2020b)	P	BERT-based matching and aggregation	Quora		MNLI, SNLI
Sajjad et al. (2020)	P	pruned BERT, RoBERTa, XLNet	MSRP, Quora		MNLI, RTE

S = simple word embedding-based model, H=hybrid model, NN = neural network,
P = pretrained language models

Table 3.5: Overview of generic approaches to Semantic Similarity Detection.

Neural approaches There exist a variety of neural components which are commonly used in NLP. We broadly categorise recent neural approaches based on their main neural network components into LSTM-based (Hochreiter and Schmidhuber, 1997), CNN-based (LeCun et al., 1998a) and Transformer-based (Vaswani et al., 2017) architectures. *LSTMs* have been very popular text encoders for paraphrase detection, entailment and answer sentence prediction due to their ability to model long-range dependencies without suffering from the same vanishing gradient problems as previous RNNs. Wang et al. (2017) proposed a generic semantic similarity model with bilateral perspective matching using multiple stacked biLSTMs. *CNNs* have been very popular for sentence classification tasks and have the benefit of relatively few parameters compared to LSTMs. Studies have used them for mainly two purposes: as an encoder and as a feature extractor. An example for CNN-based encoders is Shen et al. (2018a) who proposed to enhance CNNs with context-aware filters. Other works have incorporated CNNs as feature extractors on top of similarity score matrices. For example, Pang et al. (2016) proposed a CNN on top of a pairwise word similarity score matrix. In a similar approach, Gong et al. (2018) employed a CNN to extract interaction patterns between encoded word representations. Other work has aimed at enriching CNNs with attention (Yin et al., 2016), which was inspired by the success of attention in RNNs (Bahdanau et al., 2015). *Hybrid CNN-LSTM* approaches have proposed architectures which combine both components due to their individual success: Rao et al. (2019) proposed a hybrid CNN-LSTM model with co-attention. Conneau et al. (2017) found a hybrid LSTM-CNN architecture to work best for their universal sentence encoder purpose. Wang and Jiang (2016) proposed an LSTM encoder with attention, followed by a CNN feature extractor. *Transformer-based* architectures have become popular as they offer the benefits of neural attention but are highly parallelisable. Recent work by Tay et al. (2019) has proposed an even more lightweight variant in the form of Quaternion Transformer models.

Pre-trained models Neural models have gained wide-spread popularity in NLP in recent years, but due to their large number of model parameters, such architectures need to be trained on sufficiently large datasets and tend not to perform well on smaller datasets. This problem has resulted in a new pretrain-finetune paradigm: (1) a deep neural model is pre-trained on a large generic dataset with one or more self-supervised tasks before (2) fine-tuning it on a smaller task-specific downstream dataset. While pre-training has successfully been used for individual word embeddings (see Chapter 2.3.2) such as word2vec (Mikolov et al., 2013) in

the past, these representations were static and context-independent. In contrast, more recent contextualised pre-trained models (see Chapter 2.3.3) offer the benefit of providing context-sensitive word representations. Peters et al. (2018) trained a multi-layer bidirectional LSTM model based on a language modelling objective and demonstrated that their resulting ELMo model performed well when fine-tuned on a range of downstream tasks (see Chapter 2.3.3.1 for more details). This was followed by Devlin et al. (2019) who proposed the popular pre-trained Transformer model BERT (refer to Chapter 2.3.3.2 for a detailed description). They introduced a masked language modelling objective and a next sentence prediction task as pre-training objectives. BERT has set a new benchmark for many NLP tasks and sparked an entire subfield of research that seeks to understand this model's internal workings and its limitations. One line of subsequent work explores optimal pre-training and tuning decisions for BERT, such as RoBERTa (Liu et al., 2019). Another line of work aims to reduce the size of pre-trained models through novel compression approaches, such as model distillation (Sanh et al., 2019) and pruning techniques (Sajjad et al., 2020). Many other studies have proposed methods which directly incorporate BERT representations, such as Reimers and Gurevych (2019) and Wang et al. (2020b).

3.6 Bringing together Semantic Similarity Detection Tasks: The Case of Community Question Answering

Community Question Answering Online question answering communities, such as web forums or question answering websites (e.g. StackExchange, Yahoo! Answers, Quora or Zhihu) provide a wealth of information to questions on a wide range of topics. The typical user behaviour on community question answering (CQA) platforms follows the following pattern (Blooma and Kurian, 2011): A user asks a question, for which other users can then post answers. An answer can receive comments from other members of the community. After receiving a satisfying answer, the initial asker may thank other users for their responses or - depending on the platform - choose the best answer and mark the question as resolved. Some websites further provide community members with the option to cast positive and negative votes for given answers. The richness of CQA data offers the opportunity to address a wide range of research topics, including question quality (complexity, popularity, long-term value), answer quality (irrelevant, incomplete, partial, incorrect, biased),

question and answer quality assessment (identify high vs low quality questions and answers), question topic classification, best answer selection, identifying abusive behaviour, question retrieval (question paraphrase detection), answer summarisation (Srba and Bielikova, 2016) and fact-checking (Mihaylova et al., 2018). However, the main goal of automated CQA systems is to provide the most suitable answers on the recently posted questions in the shortest possible time (Srba and Bielikova, 2016).

Pipeline A full CQA system may address a new incoming question with the following steps depicted in Figure 3.2: (1) retrieving similar existing questions, (2) identifying answers within the thread of the question paraphrases, (3) identifying answer candidates for the new question based on the previous two steps and (4)summarising multiple useful answers to create the final answer, which in the simplest scenario could just be returning the answer candidate with the highest score. This thesis focuses on the first three of these tasks. These question paraphrase detection and Answer Selection tasks are of crucial importance for CQA and have been featured in multiple recent SemEval competitions (Nakov et al., 2017):

A) Question-comment similarity[2]: Given a question from a question-comment thread, rank the comments according to their relevance (similarity) with respect to the question.

[2]The classification scenario of this task corresponds to Answer Sentence Selection.

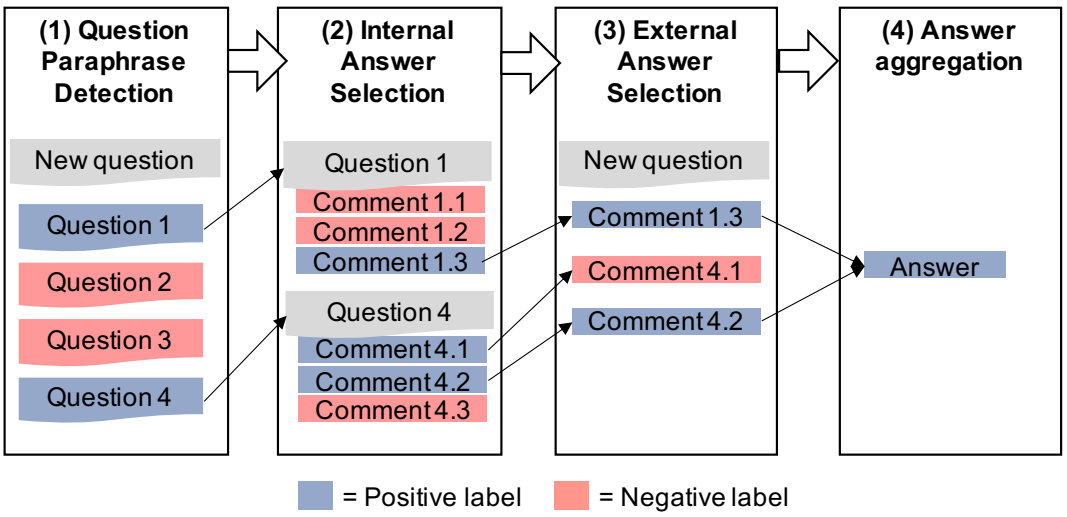


Figure 3.2: Automated Community Question Answering pipeline consisting of 4 subtasks.

B) Question-question similarity[3]: Given a new question, rerank all similar questions retrieved by a search engine, assuming that the answers to similar questions should also answer the new question.

C) Question-external comment similarity[2]: Given (i) a new question and (ii) a large collection of question-comment threads created by a user community, rank the comments that are most useful for answering the new question.

The original formulation of these tasks is as a ranking problem based on the provided binary annotations (indicating paraphrases or relatedness between the text pairs). This thesis focuses on the binary classification setup, which is more generic and fits into the Semantic Similarity Detection framework.

Connection Figure 3.3 visualises the relationship between Semantic Similarity Detection and CQA. Both research areas overlap in the tasks of Answer Selection and - partially - Paraphrase Detection. While Semantic Similarity Detection focuses on Paraphrase Detection in general (including both declarative and interrogative sentences), CQA is primarily concerned with detecting question paraphrases to exploit previously answered questions. In direct comparison, Rodrigues et al. (2018) found that Paraphrase Detection in interrogatives is more challenging than in declaratives due to lower levels of lexical overlap. Previous studies reported that general-purpose Paraphrase Detection datasets such as MSRP lack examples with such properties. Including these more challenging examples is beneficial for developing robust models, while addressing them within a generic framework together

[3]The classification scenario of this task corresponds to Paraphrase Detection.

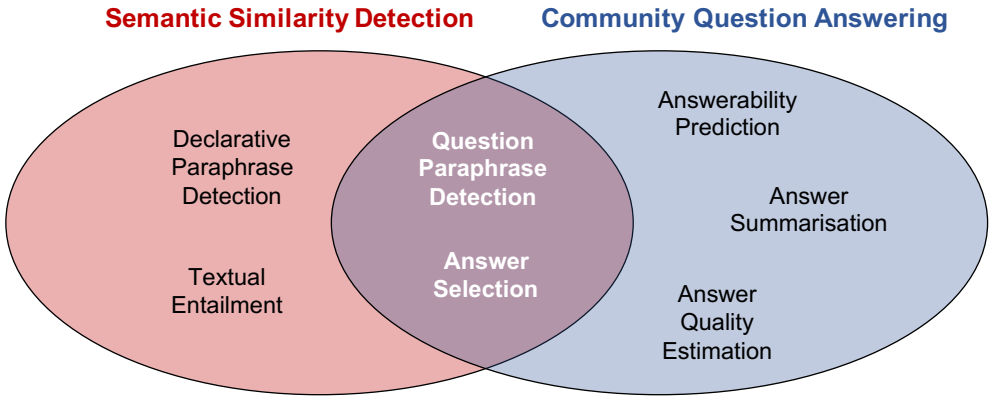


Figure 3.3: Relationship between Semantic Similarity Detection and Community Question Answering

with other closely related tasks avoids replicating work and encourages developing models with generic abilities. Moreover, CQA can provide a real-world application area for generic and robust Semantic Similarity Detection models.

Part II

Datasets and Evaluation

CHAPTER 4

Datasets

4.1 Overview

Selection This section provides an overview on several well-known Semantic Similarity Detection datasets which are used in subsequent chapters. The selected datasets cover a wide range of sizes (small vs large), document lengths (single vs multi-sentence) and data sources (online forums vs question answering websites vs news websites). A comparison of essential properties is provided in Table 4.1. While our main focus is on CQA datasets which introduce particular challenges as discussed in chapter 1, we also include a general-purpose paraphrase detection dataset in our experiments to verify if our methods generalise to Semantic Similarity Detection beyond CQA. We discuss the particularities of each dataset in the following sections and place a special emphasis on why and how the dataset was constructed, as this may introduce certain biases.

Construction Obtaining high-quality naturally occurring text pairs with a certain degree of semantic similarity is non-trivial. Generally, the process of construct-

Dataset	Task	Source	Positive instances	Size
MSRP	Paraphrase detection	News websites	67%	5k
Quora	Duplicate question identification	Q&A website	37%	404k
PAWS	Adversarial paraphrase detection	Q&A website, synthetic	31%	12k
SemEval	(A) External answer selection	Online forum	39%	26K
	(B) Paraphrase detection	Online forum	36%	4K
	(C) External answer selection	Online forum	9%	47K
STS	Similarity scoring	various	23%	8K

Table 4.1: Properties of selected semantic similarity detection data sets.

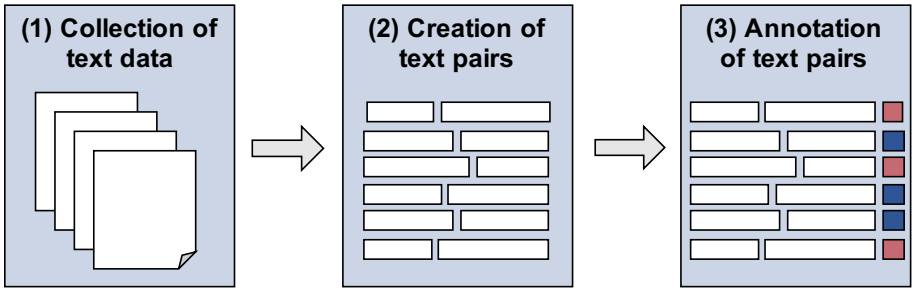


Figure 4.1: Overview of typical dataset creation stages.

ing datasets for Semantic Similarity Detection can be broken down into three main stages (see Figure 4.1): (1) collecting text data from online sources, (2) creation of text pairs and (3) text pair annotation by human judges. Stage 1 usually starts with a manual selection of initial source material, e.g. a list of news websites or forum posts (Agirre et al., 2012). As the manual selection is time-consuming and does not scale well, this initial collection is often used to retrieve a large set of possibly related texts based on similarity metrics or search engines. Stage 2 involves the combination of individual texts to pairings. Naively, this can be done by combining each text with each other, creating a large number of pairings which can be narrowed down through filtering criteria such as text overlap. However, filtering criteria need to be chosen carefully, as too strict criteria can filter out interesting examples and too lenient criteria may result in a large number of clearly irrelevant pairings. It is also important to consider how it can be ensured that the dataset contains challenging distractors. Another pairing method is to compute similarity measures between sentences based on dense representations with a certain cut-off value. Alternatively, a search engine can be used to retrieve likely candidates, which can introduce less transparent selection biases. Stage 3 involves the collection of gold labels. This is usually done through annotation by multiple human annotators based on carefully developed annotation guidelines. In an ideal setup, multiple expert annotators are asked to annotate the same instance, discarding any instances with conflicting annotations. However, given practical time or monetary constraints, compromises need to be made (e.g. using crowd workers or asking annotators to resolve disagreements).

4.2 MSRP

Motivation As the first reasonably sized general purpose paraphrase detection dataset, the Microsoft Research Paraphrase Corpus (MSRP, Dolan and Brockett

Sentence 1	Sentence 2	Label
Charles O. Prince, 53, was named as Mr. Weills successor.	Mr. Weills longtime confidant, Charles O. Prince, 53, was named as his successor.	is_paraphrase

Table 4.2: Example from the MSRP dataset.

2005) is very well-known and widely used. It consists of more than 5k sentence pairs with binary human paraphrase judgements. At the time of creation, there was a lack of sufficiently large datasets for automatic paraphrase detection, the authors aimed at collecting naturally-occurring paraphrases as opposed to crafting them manually. Based on the observation that there exists coverage of the same event from various sources online, the creators set out to collect multiple reports from different news websites to extract natural paraphrases at scale automatically.

Construction First, they collected 13 million sentence pairs from 32k news clusters over a two year period. Then, an initial database of 61k sentence pairs was constructed from the sentence pairs by filtering for low word-based Levenshtein distance between sentences. A number of additional heuristics were applied to filter out sentence pairs which were unlikely to be paraphrases, narrowing down the number of candidate sentence pairs to around 50k. Then, they used a supervised feature-based classifier to identify likely paraphrases resulting in 5801 sentence pairs. Finally, the potential paraphrase pairs were annotated by two human judges to indicate whether the sentences were semantically equivalent (positive label) or not (negative label). A third judge was consulted in case of disagreements, and labels were assigned through a majority vote.

Task Given two sentences from news websites, the task is to predict whether they are semantically equivalent. The dataset applies a loose paraphrase definition, as one sentence can contain additional information to the other sentence and still qualify as a paraphrase (compare Table 4.2).

4.3 Quora

Motivation The Quora question pair dataset is a large-scale duplicate question identification dataset which was released in 2017 by the community-based question answering website Quora.[1] It is very popular, especially for training neural models

[1] https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning

due to its currently unmatched size of more than 400k sentence pairs. An important principle of Quora is to have only one page per logically distinct question and prevent the creation of duplicate question pages which lead to inefficiencies for both information seekers (searching for answers in many locations) and content writers (reproducing answers to similar questions). As many new pages are created every day, this dataset was built to speed up the detection of duplicate questions through machine learning models.

Construction Although the data collection process is not published in detail, the gold labels are based on platform users' votes to merge duplicate questions. The creators pointed out that the initial sampling method resulted in an imbalanced dataset with more positive than negative examples. Therefore, related questions were used as additional negative examples to create a more balanced label distribution. As the original distribution does not provide an official dataset split, we use the same training / development / test set partition as Wang et al. (2017).

Task Given two sentences, the task is to predict whether they are duplicate questions in a binary classification task.

4.4 PAWS

Motivation Paraphrase Adversaries from Word Scrambling (PAWS, Zhang et al. 2019c) is an adversarial paraphrase detection dataset. Zhang et al. (2019c) observe that some existing datasets like Quora contain too few examples with high word overlap for models to learn word order, resulting in a tendency of models assign positive labels to sentences pairs with high word overlap rather than learning word

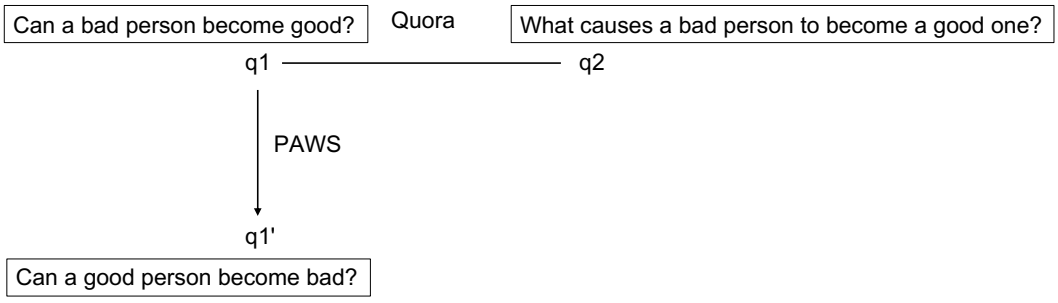


Figure 4.2: Relationship between Quora and PAWS. Quora consists of question pairs such as $q1$ and $q2$. For PAWS, a question $q1$ from Quora was automatically altered to produce a new question $q1'$ with high lexical overlap.

order. Therefore, they set out to create adversarial paraphrases and non-paraphrases by automatically creating sentence pairs with high overlap based on source sentences from the Quora dataset.

Construction For automatic paraphrase generation, they use two methods: (1) word order permutations which mainly produce negative examples and (2) back translation which tends to result in positive examples. The generated sentence pairs are then annotated by five human raters and only retained if at least four annotators agreed. The final dataset consists of more than 12k sentence pairs.

Task The task is similar to Quora: given two questions predict if they are paraphrases in a binary classification setup.

4.5 SemEval

Motivation SemEval 2017 Task 3 provides a Community Question Answering dataset with five subtasks (Nakov et al., 2017). In this work, we focus on the English language subtasks A-C which feature (A) internal answer selection, (B) question paraphrase identification and (C) external answer selection in an online forum. Online forums provide users with a platform to freely ask and answer questions with little moderation. While these platforms provide users with great flexibility,

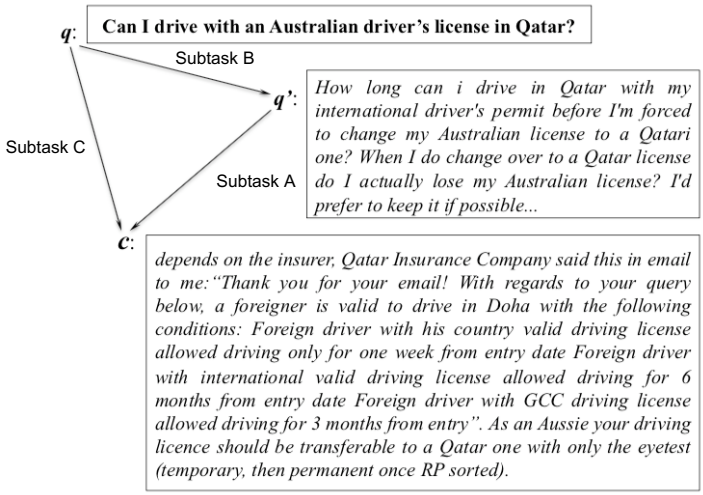


Figure 4.3: Connection between the three SemEval subtasks, showing pairwise interactions between the original question q, the related question q' and a comment c in the related question's thread. Adapted from Nakov et al. (2015).

it can also be difficult to find relevant information which may be scattered between many different threads. Therefore, Nakov et al. (2017) focus on question paraphrase identification and answer selection tasks to facilitate efficient information access on such platforms.

Construction The dataset was constructed by manually selecting a set of existing question posts from the forum of the expat website Qatar Living as original questions. For each original question, a search engine was used to retrieve 200 possibly related question-comment threads from Qatar Living. These were filtered based on certain length criteria, resulting in a set of ten remaining threads (consisting of 10 comments each) per original question. Based on an original question and ten retrieved posts from the forum, three subtasks are defined which require to distinguish relevant posts from non-relevant ones. In subtask A, the retrieved posts come from the same thread as the original question. Subtask B involves an original question and ten possibly related question posts. Subtask C is similar to A, but the retrieved posts come from external threads which increases the difficulty of the task. The relationship between the three subtasks is depicted in Figure 4.3. To obtain gold labels, three crowd workers per question are asked to annotate the relevancy of posts with respect to the original question.

Task The official SemEval setup is a ranking task requiring relevant posts to be ranked above non-relevant ones with classification metrics as secondary performance measures. However, we focus on the classification setup, as gold labels are only provided as binary judgements and this is more generic. The 2017 challenge only provides new test sets. Therefore, we use the training and test set from the 2016 challenge as training and development set (Nakov et al., 2016), while using the 2017 data (Nakov et al., 2017) as test set.

4.6 STS

Motivation The Semantic Textual Similarity benchmark dataset (STS, Cer et al. 2017) contains a collection of various English-language semantic similarity tasks from previous SemEval competitions which were held between 2012 and 2017. STS aims to predict the degree to which two sentences are semantically equivalent and measures graduations of meaning overlap on a continuous numeric scale (see Figure 4.4). It is motivated by the need to evaluate modelling techniques for semantic similarity on a wide range of domains against an interpretable benchmark.

- (5) The two sentences are completely equivalent, as they mean the same thing.
  The bird is bathing in the sink.
  Birdie is washing itself in the water basin.
- (4) The two sentences are mostly equivalent, but some unimportant details differ.
  In May 2010, the troops attempted to invade Kabul.
  The US army invaded Kabul on May 7th last year, 2010.
- (3) The two sentences are roughly equivalent, but some important information differs/missing.
  John said he is considered a witness but not a suspect.
  "He is not a suspect anymore." John said.
- (2) The two sentences are not equivalent, but share some details.
  They flew out of the nest in groups.
  They flew into the nest together.
- (1) The two sentences are not equivalent, but are on the same topic.
  The woman is playing the violin.
  The young lady enjoys listening to the guitar.
- (0) The two sentences are on different topics.
  John went horse back riding at dawn with a whole group of friends.
  Sunrise at dawn is a magnificent view to take in if you wake up early enough for it.

Figure 4.4: Examples from the STS dataset. Figure adapted from Agirre et al. (2013)

Construction The training and development data includes examples which were selected from 26 machine translation, summarisation, question answering and short answer datasets. For testing, the authors reuse SNLI sentences but repair them based on a word embedding similarity selection heuristic which averages the word embeddings in each sentence and computes the cosine similarity between the resulting sentence representations. The newly paired sentences are then annotated by crowd workers according to the STS scale.

Task The main difference between STS and the other tasks listed here is that labels are graded on a scale from 0 (sentences are completely dissimilar) to 5 (sentences are completely equivalent). In total, the dataset consists of 8628 sentence pairs.

CHAPTER 5

Evaluating Non-Obvious Cases of Semantic Similarity

5.1 Introduction

Evaluating Semantic Similarity Detection This thesis focuses on the problem of Semantic Similarity Detection - a collection of sentence pair prediction tasks which aim at automatically recognising semantic relationships between two given texts, such as semantic equivalence, entailment and question-answer relations (see Chapter 3.1). Modelling such text pair relationships is a fundamental NLP task with applications ranging from plagiarism detection to question answering (Baudiš et al., 2016). One particularly interesting application area is Community Question Answering (CQA) which is concerned with the automatic answering of questions based on user-generated content from Q&A websites and requires modelling the semantic similarity between question and answer pairs (Nakov et al., 2017; Bonadiman et al., 2017). Another popular Semantic Similarity Detection task is Paraphrase Detection (Socher et al., 2011; He et al., 2015; Tomar et al., 2017), which models the semantic equivalence between a pair of sentences. Evaluation for these tasks is usually performed with metrics, such as mean average precision (MAP), F1 or accuracy (refer to section 2.5.2), which give equal weight to all examples in a dataset, regardless of their difficulty. However, as highlighted by the examples in Table 5.1, not all items within Semantic Similarity Detection datasets are equally difficult to resolve. For example, it is easier to detect paraphrases which share most of their vocabulary compared to those with low lexical overlap.

Biases in NLU Recent work in natural language understanding (NLU) has highlighted the need to shed light on biases within existing datasets to understand actual capabilities of current models (Wadhwa et al., 2018a; Rajpurkar et al., 2018).

ID	Case	Sentences
160174	P_o	what's the origin of the word o'clock? what is the origin of the word o'clock?
115695	P_n	which is the best way to learn coding? how do you learn to program?
193190	N_o	what are the range of careers in biotechnology in indonesia? how do you tenderize beef stew meat?
268368	N_n	what is meant by 'e' in mathematics? what is meant by mathematics?

Table 5.1: Examples for difficulty cases from the development set of the Quora dataset. o=obvious, n=non-obvious, N=negative label, P=positive label

Kaushik and Lipton (2018) demonstrated that reading comprehension models could exploit certain dataset properties to achieve high performance even when crucial task information (such as question or passage) was withheld. Similarly, Gururangan et al. (2018) showed that natural language inference model performance could be inflated by annotation artefacts, enabling models to predict the correct class by only looking at the hypothesis without observing the premise. Weissenborn et al. (2017) illustrated that seemingly complex examples of span prediction question answering datasets could be resolved with a simple context-type heuristic, requiring no real language understanding. Moreover, Yaghoub-Zadeh-Fard et al. (2019) reported common issues in crowd-sourced paraphrasing datasets.

Lexical overlap Multiple NLU studies have reported lexical overlap bias. Wadhwa et al. (2018b) showed that competitive neural reading comprehension models are susceptible to shallow patterns, such as lexical overlap. In addition, McCoy et al. (2019) highlighted lexical overlap bias in natural language inference datasets. Lexical overlap has also been scrutinised in the context of Paraphrase Detection: Rodrigues et al. (2018) found that interrogative sentence pairs are more difficult to resolve than declarative pairs in Paraphrase Detection datasets as the latter contain more word overlap. Moreover, multiple studies (Weeds et al., 2005; Zhang and Patrick, 2005) have criticised the high word overlap in the Paraphrase Detection dataset MSRP. However, currently, there is no systematic study of lexical overlap across a wider range of Semantic Similarity Detection datasets and the influence of lexical overlap on example difficulty remains unclear.

Research Questions This chapter focuses on the following two research questions (RQ1 and RQ2 from Chapter 1): *First, which biases exist in current semantic*

similarity datasets? And second, what are ways to account for identified dataset biases during model evaluation? To address these questions, we investigate the role of superficial word overlap in existing Semantic Similarity Detection datasets. We quantify lexical overlap bias in existing datasets, discuss implications for the difficulty of individual dataset instances and propose alternative evaluation metrics which take lexical overlap into account.

Contributions Specifically, we make the following contributions:

1. We propose a criterion to distinguish between obvious and non-obvious instances in semantic similarity datasets (section 5.6).

2. We characterise current datasets in terms of lexical overlap bias (section 5.4) and the extent to which they contain obvious vs non-obvious items (section 5.6).

3. We propose alternative evaluation metrics based on example difficulty (section 5.7) and provide a reference implementation.[1]

This chapter is based on our ACL 2019 paper which is provided as Appendix A.

5.2 Datasets and Tasks

This chapter analyses a wide range of well-known semantic similarity datasets which were described in detail in the subsections of Chapter 4. Among these are three popular Paraphrase Detection datasets: the general purpose news-based paraphrase dataset **MSRP** (introduced in section 4.2), the duplicate question dataset provided by the CQA platform **Quora** (refer to section 4.3) and the synthetically-generated adversarial counterpart to Quora **PAWS** (compare section 4.4). We also include three CQA datasets originating from the online forum *Qatar Living*: the internal Answer Sentence Selection dataset **SemEval A**, the question Paraphrase Detection dataset **SemEval B** and the external Answer Sentence Selection dataset **SemEval C** (described in section 4.5). We further include the semantic textual similarity dataset **STS** (section 4.6) which incorporates a mix of previously published datasets that were reannotated for semantic similarity on a scale from 1 to 5. A task applicable to all of these datasets is predicting the semantic similarity between two text snippets in a binary classification setting. We focus on a binary classification scenario, as the ranking problem is only applicable to some of the datasets. Binary

[1] Our code is available at https://github.com/wuningxi/LexSim.

labels are already provided for all tasks except for STS. In the case of STS, we convert the scores into binary labels. Based on the description of the relatedness scores in Cer et al. (2017), we assign a positive label if relatedness ≥ 4 and a negative one otherwise to use a similar criterion as in the other datasets.

5.3 Measuring Lexical Overlap between Text Pairs

To measure superficial overlap between text pairs in semantic similarity datasets, we experiment with two well-known metrics: Dice coefficient and Jensen-Shannon divergence. Although this measurement could be performed based on smaller components (e.g. characters) or larger units (e.g. ngrams), we use tokens, which represent a middle-ground. We focus on tokens because they are a commonly used unit for text overlap, well defined (in contrast to, e.g. subword units) and represent the building blocks for many NLP models.

Dice coefficient Dice coefficient (DC) is a well-established set metric which measures the similarity between two sets A and B as follows (Dice, 1945):

$$DC(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

with results ranging between 0 (low similarity) and 1 (high similarity). Applied to language, DC is widely used to measure text similarity (Abacha and Demner-Fushman, 2017; Stefanovič et al., 2019). It makes an intuitive lexical overlap measure by dividing the number of overlapping words between the sentences by the total number of tokens in both sentences. When calculating the Dice coefficient to measure lexical overlap, we represent each sentence as a set of tokens. For example, the sentence pair 'which is the best way to learn coding?' and 'how do you learn to program?' can be represented as

$$A = \{\text{'which'}, \text{'is'}, \text{'the'}, \text{'best'}, \text{'way'}, \text{'to'}, \text{'learn'}, \text{'coding'}\} \tag{5.1}$$

and

$$B = \{\text{'how'}, \text{'do'}, \text{'you'}, \text{'learn'}, \text{'to'}, \text{'program'}\} \tag{5.2}$$

resulting in an intersection of $A \cap B = \{\text{'to'}, \text{'learn'}\}$ and $DC = \frac{2*2}{8+6} = 0.286$.

Jensen-Shannon divergence Alternatively, we can use Jensen-Shannon divergence (JSD, Lin 1991) which measures the distance between two probability distri-

butions P and Q as follows:

$$\begin{aligned} \text{JSD}(P,Q) &= \frac{1}{2}\text{KLD}(P,M) + \frac{1}{2}\text{KLD}(Q,M) \\ M &= \frac{1}{2}(P+Q) \end{aligned} \tag{5.3}$$

where $\text{KLD}(R,S)$ is Kullback-Leibler divergence between two probability distributions R and S which is defined as:

$$\text{KLD}(R,S) = \sum_{x \in X} R(x) log(\frac{R(x)}{S(x)})$$

JSD is a symmetric version of KLD with bounds between 0 (low divergence/high similarity) and 1 (high divergence/low similarity) and has been used to measure text similarity in previous studies (Wartena, 2013; Gerlach and Font-Clos, 2020). In contrast to DC, it can capture lexical overlap while incorporating a notion of multiple word occurrences. To represent text pairs as two probability distributions, we use a bag-of-words representation (see Chapter 2.3.1.2) for each sentence over the joint sentence pair vocabulary which is converted into probabilities by dividing word counts by frequencies. To illustrate this on the previous example, we first represent the two sentences with a bag-of-words representation over their sorted joint vocabulary as

$$\begin{aligned} C = \{&\text{'best'} : 1, \text{'coding'} : 1, \text{'do'} : 0, \text{'how'} : 0, \text{'is'} : 1, \text{'learn'} : 1, \\ &\text{'program'} : 0, \text{'the'} : 1, \text{'to'} : 1, \text{'way'} : 1, \text{'which'} : 1, \text{'you'} : 0\} \end{aligned} \tag{5.4}$$

and

$$\begin{aligned} D = \{&\text{'best'} : 0, \text{'coding'} : 0, \text{'do'} : 1, \text{'how'} : 1, \text{'is'} : 0, \text{'learn'} : 1, \\ &\text{'program'} : 1, \text{'the'} : 0, \text{'to'} : 1, \text{'way'} : 0, \text{'which'} : 0, \text{'you'} : 1\} \end{aligned} \tag{5.5}$$

Then, we normalise the word counts by dividing word frequencies by the total number of tokens in the sentence to obtain probability distributions P and Q:

$$P = [0.125, 0.125, 0., 0., 0.125, 0.125, 0., 0.125, 0.125, 0.125, 0.125, 0.] \tag{5.6}$$

and

$$Q = [0., 0., 0.16667, 0.16667, 0., 0.16667, 0.16667, 0., 0.16667, 0., 0., 0.16667] \tag{5.7}$$

This results in $\text{JSD}(P,Q) = \frac{1}{2}0.6944 + \frac{1}{2}0.7309 = 0.7126$ for our example.

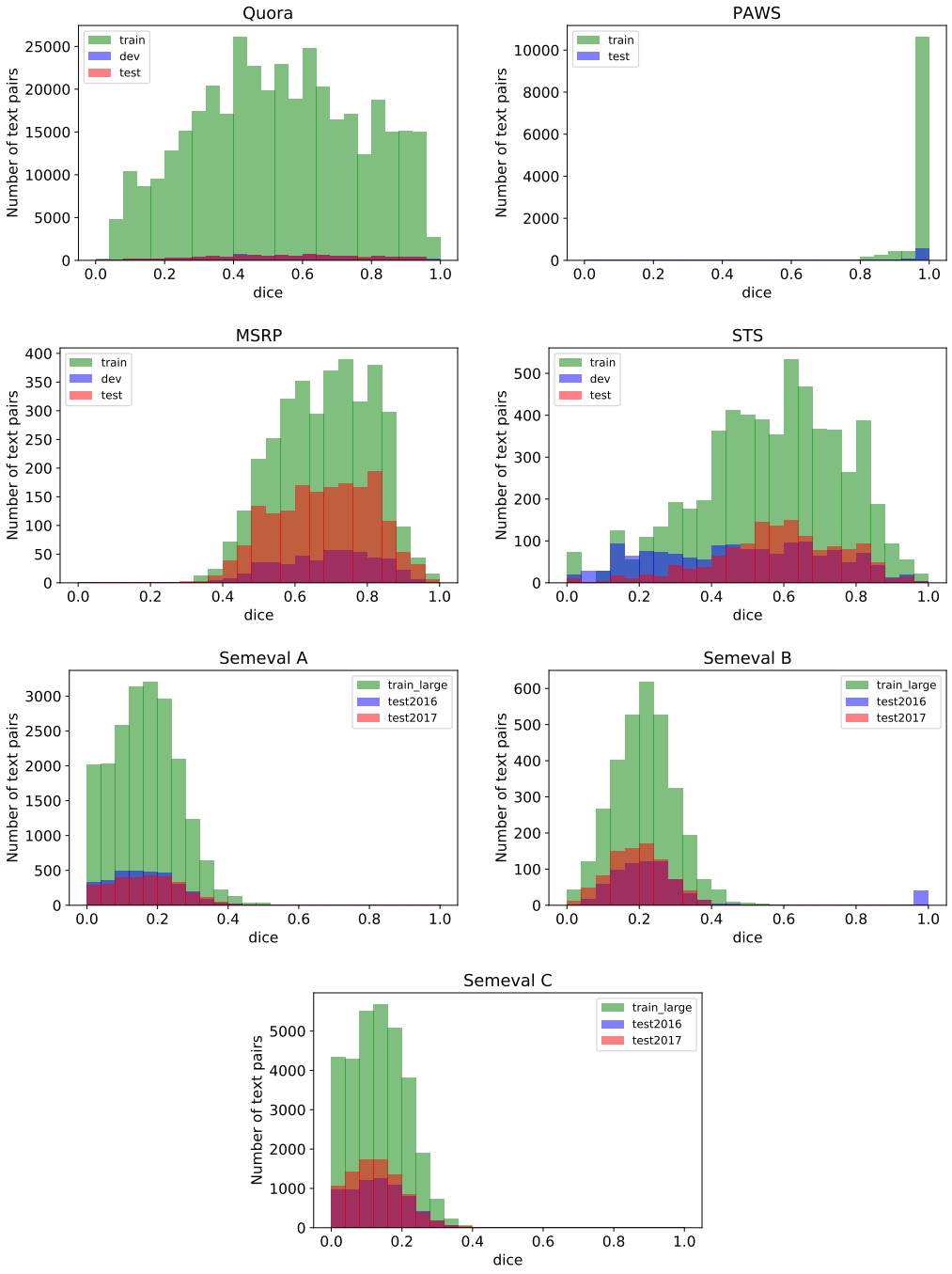


Figure 5.1: Dice coefficient distribution by training, development and test set across different semantic similarity datasets.

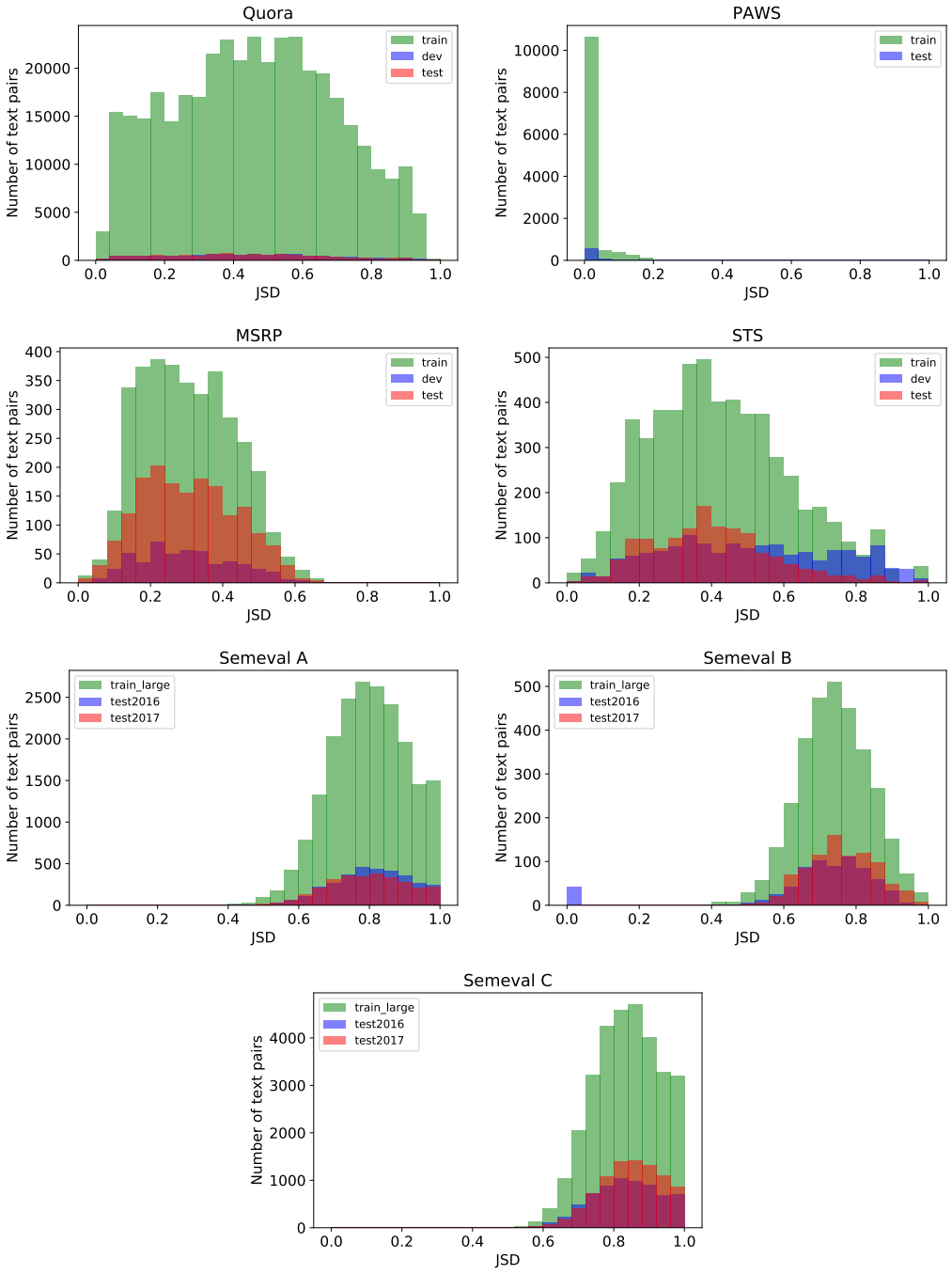


Figure 5.2: Lexical divergence distribution by training, development and test set across different semantic similarity datasets. JSD=Jensen-Shannon divergence.

5.4 Lexical Overlap in Current Datasets

Lexical divergence Figures 5.1 and 5.2 show the distribution of Dice Coefficients and Jensen-Shannon Divergence by dataset splits respectively. In direct comparison, the distributions of the two metrics appear mirrored because JSD measures lexical divergence, while DC captures lexical similarity.[2] There are slight differences for cases with multiple occurrences of the same token, but overall the metrics capture very similar patterns across the datasets. The datasets clearly differ with respect to the degree of lexical overlap they contain: PAWS contains the lowest degree of lexical divergence between text pairs (majority of instances < 0.2), as examples in this dataset were automatically generated to have high word overlap (refer to Chapter 4 for details on dataset construction). Text pairs in MSRP tend to have low to medium JSD scores (majority of instances < 0.6), as examples with low lexical similarity were filtered out during dataset construction. The three SemEval CQA datasets show a high degree of lexical divergence (majority of instances > 0.6), especially in the external QA scenario (task C). Quora shows a wide range of lexical overlap, possibly due to the data collection process which made use of platform users flagging questions as 'duplicates' or 'related questions' , which appears to have resulted in a greater variation of lexical overlap between question pairs. STS also contains text pairs with various degrees of lexical divergence which is likely related to the wide range of tasks included in the dataset. However, there also appears a different lexical divergence pattern between development and test set examples which originate from different datasets.

Relationship between lexical divergence and gold labels Figures 5.3 and 5.4 visualise the distribution of Dice Coefficients and Jensen-Shannon Divergence by gold labels. As above, the distributions of the two metrics appear mirrored but consistent. Overall, pairs with negative labels tend to be associated with higher JSD scores than pairs with positive labels. This pattern is especially prominent in Quora, MSRP and STS, where distinct distributions emerge for positive vs negative labels. Therefore, lexical overlap between sentence pairs provides direct clues for label assignment (lexical overlap bias). In contrast, lexical divergence is much less distinct for positive and negative example pairs in SemEval and the adversarial PAWS dataset, making Semantic Similarity Detection on these datasets more challenging as models cannot rely on superficial text overlap as a heuristic for class membership.

[2] To avoid repetition, we focus on describing characteristics based on lexical divergence.

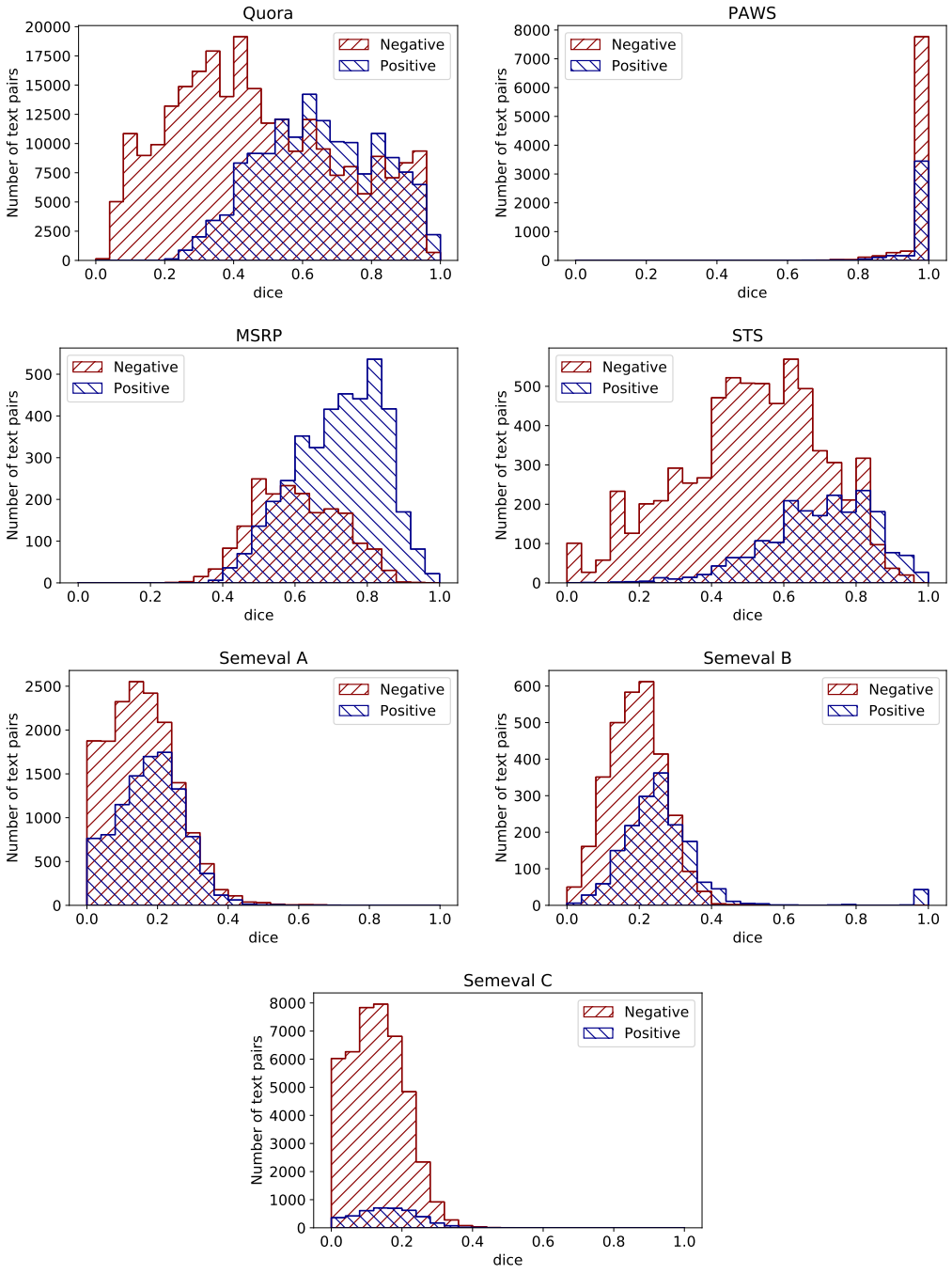


Figure 5.3: Distribution of dice coefficients by labels across datasets.

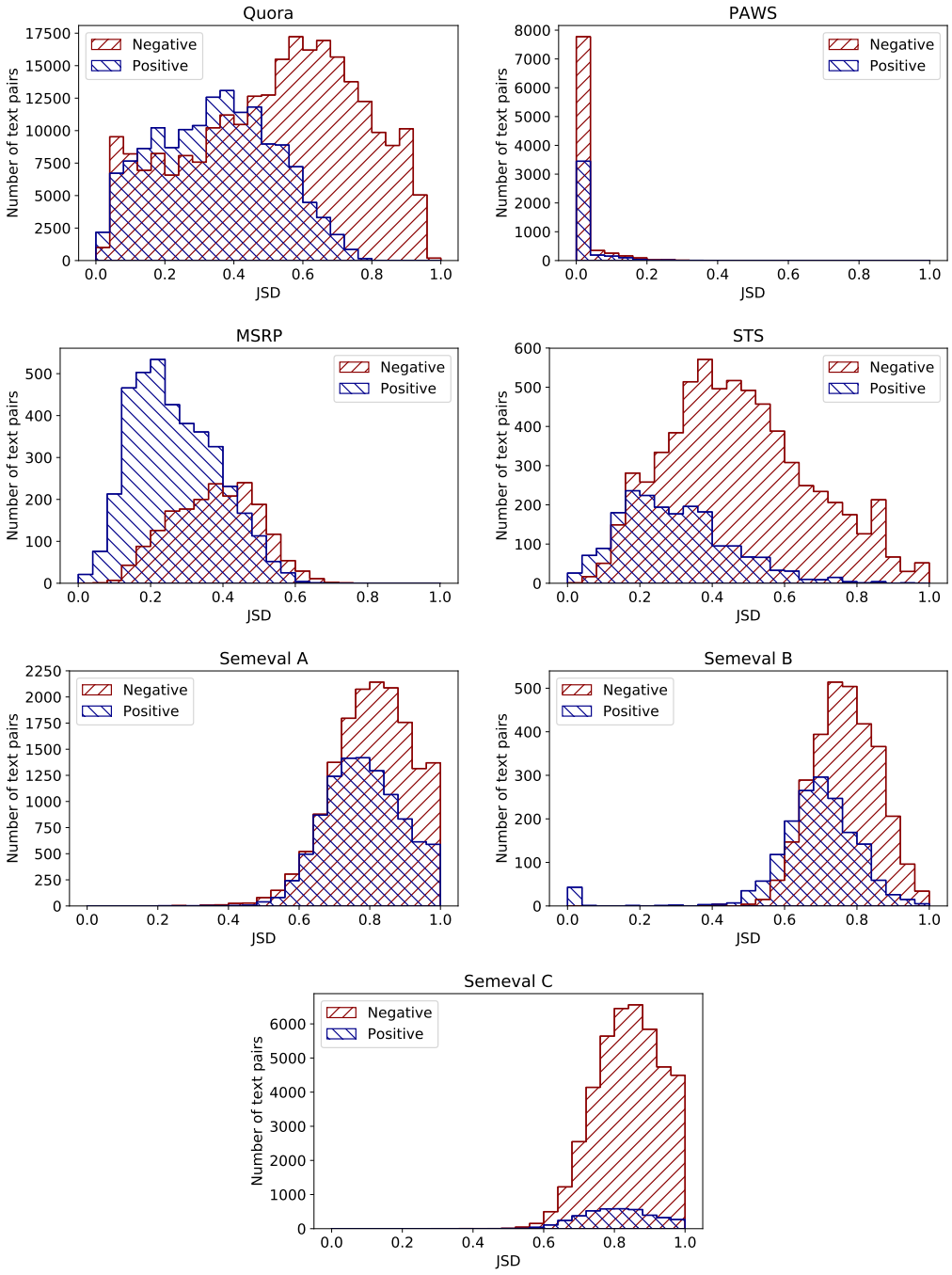


Figure 5.4: Distribution of lexical divergence scores by labels across datasets. JSD=Jensen-Shannon divergence.

Lexical overlap metric Measuring lexical similarity and divergence based on DC and JSD, we find consistent results between the two measures. We give preference to the probability distribution-based metric JSD, which is symmetric and can take into account multiple word occurrences. We hence use JSD for the remainder of this chapter.

5.5 Lexical Overlap Bias

Above, we described that lexical overlap between sentence pairs can provide direct clues for class membership, resulting in lexical overlap bias. To further quantify the prevalence of lexical overlap bias in the datasets, we use a baseline which calculates the mean JSD on the training set and uses it as a threshold for assigning labels (predicting a positive label if the JSD of a dataset instance is lower than the average JSD of the training set and a negative label otherwise). We found that this simple baseline achieved high F1 scores on Quora and MSRP (around 0.7), medium performance on STS, SemEval A and SemEval B (between 0.4 and 0.6 F1), and low performance on SemEval C and PAWS ($<$ 0.1 F1), see Table 5.2. This demonstrates how models can exploit superficial surface similarity in many existing Semantic Similarity Detection datasets. One could object that lexical similarity is to some extent a naturally occurring feature of semantic similarity, rather than a dataset bias. However, we argue that while semantic similarity models should without question be able to capture lexical similarity, the main focus of training and evaluating Semantic Similarity Detection models should be on identifying less obvious kinds of similarity. Using datasets in which the labels of the majority of instances can be predicted solely based on lexical overlap has negative consequences for both training and evaluating models. Using such datasets during training risks producing models which focus only on shallow patterns. Using biased datasets during evaluation risks producing inflated scores and overestimating model capabilities. The rest of this chapter therefore proposes how lexical overlap can be accounted for during model evaluation.

	SemEval						
	A	B	C	Quora	PAWS	MSRP	STS
overlap baseline (JSD)	.5728	.4325	.0813	.6928	.000	.7002	.5179

Table 5.2: Performance (F1) of lexical overlap baseline.

5.6 Distinguishing Obvious from Non-Obvious Examples

Proposed criterion As shown in the previous sections, pairs with high lexical divergence tend to have a negative label in the analysed datasets (e.g. N_o in Table 5.1), while low lexical divergence is associated with a positive label (e.g. P_o in Table 5.1). Intuitively, these are cases which should be relatively easy to identify. More difficult to correctly classify are text pairs with a positive label but high lexical divergence (e.g. P_n in Table 5.1), or a negative label despite low lexical divergence (e.g. N_n in Table 5.1). We propose to use the following criterion (Table 5.3) to categorise cases in terms of their difficulty level:

	positive label	negative label
low divergence	obvious pos (P_o)	non-obvious neg (N_n)
high divergence	non-obvious pos (P_n)	obvious neg (N_o)

Table 5.3: Defining obvious and non-obvious similarity cases based on labels and lexical overlap.

Pairs are categorised into high and low lexical divergence categories by comparing their JSD score to the median of the entire JSD distribution in order to account for differences between datasets (>median: high divergence, ≤median: low divergence). To verify whether this automatic difficulty distinction corresponds to real-world difficulty, we annotated the semantic relatedness of 100 randomly chosen pairs from the Quora development set and measured the inter-annotator agreement based on Fleiss' Kappa (see Table 5.4). The agreement for non-obvious cases (P_n and N_n) is significantly lower (p-value< 0.01 with permutation test) than for obvious cases (P_o and N_o) and the average annotation time per item is longer for non-obvious cases, confirming the validity of this distinction.

	Fleiss' Kappa	Average time per pair	Instances
P_o	0.6429	11.58s	35
P_n	0.0878	11.68s	15
N_o	0.3886	12.50s	34
N_n	0.0892	13.83s	16
total	0.6267	12.27s	100

Table 5.4: Statistics for manual annotation on Quora. o=obvious, n=non-obvious, N=negative, P=positive

	SemEval A	SemEval B	SemEval C	Quora	PAWS	MSRP	STS
P_o	5893	1162	2492	107612	3283	2398	1597
P_n	4428	531	1590	41691	684	1502	409
N_o	8842	1843	22155	160410	1264	1398	3900
N_n	7377	1213	21253	94632	7434	503	2719
obvious cases	56%	63%	52%	66%	36%	65%	64%
median JSD	0.80	0.74	0.84	0.46	0.00	0.30	0.41

Table 5.5: Difficulty case splits across datasets (train, dev and test combined).

Obvious vs non-obvious cases in datasets Table 5.5 shows the number of instances in the four cases across datasets. Across the analysed human-generated datasets, there are more obvious positives (P_o) than non-obvious positives (P_n) and more obvious negatives (N_o) than non-obvious negatives (N_n), with all obvious cases combined (P_o+N_o) making up more than 50% of the pairs across all datasets. Solely the automatically constructed PAWS dataset contains more non-obvious cases than obvious ones as adversarial examples with low lexical overlap were an explicit design criterion. It, therefore, makes a suitable evaluation counterpart to the other datasets.

5.7 Evaluating Model Predictions based on Difficulty

Obvious vs non-obvious TPR and TNR Based on the distinction between obvious and non-obvious cases, we now propose alternative evaluation metrics and compare them with currently used metrics. Due to the lack of openly available model prediction files, we only present our analysis for the SemEval CQA subtasks (see Tables 5.6 to 5.8).[3] First, we calculate the true positive rate TPR (for P_o and P_n) and true negative rate TNR (for N_o and N_n) to analyse model performance within each difficulty category. Systems across the three SemEval 2017 CQA subtasks consistently perform worse on the hard cases compared to the obvious cases ($TPR_n \leq TPR_o$ and $TNR_n \leq TNR_o$), while there are comparatively small changes in the random baseline which predicts all classes with equal probability.

Obvious vs non-obvious F1 To compare how well models perform overall on obvious vs non-obvious cases, we compute non-obvious F1 scores ($F1_n$) similar to

[3]Prediction files were obtained from http://alt.qcri.org/semeval2017/task3/index.php?id=results.

	KeLP	Beihang MSRA	IIT UHH	ECNU	bunji	EICA	Swiss Alps	FuRong Wang	FA3L	Snow Man	random
TPR_o	.652	**1.00**	.800	.790	.681	.328	.333	.562	.691	.677	.501
TPR_n	.496	**1.00**	.676	.636	.575	.269	.223	.399	.478	.469	.499
TNR_o	.909	.000	.731	.877	.894	.959	**.984**	.913	.787	.900	.515
TNR_n	.908	.000	.676	.820	.851	**.953**	.950	.892	.751	.757	.536
$F1_o$	.751	.682	.781	**.829**	.765	.480	.494	.684	.731	.765	.513
$F1_n$	.628	.686	.686	**.707**	.672	.410	.352	.533	.560	.555	.519
F1	.698	.684	.739	**.777**	.725	.450	.433	.621	.659	.673	.516
MAP	**.884**	.882	.869	.867	.866	.865	.862	.843	.834	.818	.623

Table 5.6: Proposed evaluation metrics for top 10 primary submissions on SemEval Task A. The systems are ordered in columns according to their MAP ranking. Bold indicates the highest value for each metric. We indicate the 2^{nd} and 3^{rd} systems based on MAP, F1 and $F1_n$.

	Sim Bow	LearningToQuestion	KeLP	Talla	Beihang MSRA	NLM NIH	Uinsuska TiTech	IIT UHH	SCIR QA	FA3L	random
TPR_o	.976	**1.00**	.920	.760	**1.00**	.880	.752	.704	.912	.448	.552
TPR_n	.842	**1.00**	.632	.763	**1.00**	.500	.421	.737	.842	.263	.395
TNR_o	.609	.000	.831	.684	.000	.841	.858	.682	.709	**.861**	.495
TNR_n	.197	.000	.432	.467	.000	.397	.552	.403	.352	**.756**	.521
$F1_o$	.604	.383	**.746**	.548	.383	.736	.681	.516	.641	.473	.348
$F1_n$	.198	.195	.199	**.247**	.195	.154	.164	.221	.234	.160	.147
F1	.424	.312	**.506**	.426	.312	.473	.467	.390	.464	.365	.280
MAP	**.472**	.469	.467	.457	.448	.446	.434	.431	.427	.422	.298

Table 5.7: Proposed evaluation metrics for top 10 primary submissions on SemEval Task B. The systems are ordered in columns according to their MAP ranking.

commonly used F1 scores (refer to Chapter 2.5.2) as follows:

$$P_n = \frac{TP_n}{TP_n + FP_n}$$

$$R_n = \frac{TP_n}{TP_n + FN_n}$$

$$F1_n = \frac{2P_nR_n}{P_n + R_n}$$

where TP_n is non-obvious true positives, FP_n is non-obvious false positives and FN_n non-obvious false negatives. Similarly, F1 scores for obvious items ($F1_o$) are calculated as:

$$P_o = \frac{TP_o}{TP_o + FP_o}$$

$$R_o = \frac{TP_o}{TP_o + FN_o}$$

$$F1_o = \frac{2P_oR_o}{P_o + R_o}$$

where TP_o is obvious true positives, FP_o is obvious false positives and FN_o obvious false negatives. Calculating separate F1 scores for obvious vs non-obvious instances is necessary as the high percentage of obvious cases (observed in section 5.6) can inflate the overall F1 score. We find that the models' $F1_n$ scores are consistently lower than their $F1_o$ scores. This difference is especially pronounced in subtask B, which contains the highest proportion of obvious cases (63%) of the SemEval subtasks (refer to Table 5.5).

	IIT UHH	bunji	KeLP	EICA	FuRongWang	ECNU	random
TPR_o	.570	.246	.911	.006	**1.00**	.173	.520
TPR_n	.358	.045	.836	.000	**1.00**	.045	.433
TNR_o	.898	.991	.720	**.998**	.000	.994	.502
TNR_n	.779	.965	.538	**.999**	.000	.955	.502
$F1_o$	.283	**.339**	.209	.011	.077	.260	.076
$F1_n$	.047	.028	**.054**	.000	.031	.022	.027
F1	.144	**.197**	.121	.008	.054	.136	.053
MAP	**.155**	.147	.144	.135	.132	.105	.058

Table 5.8: Proposed evaluation metrics for all primary submissions on SemEval Task C. The systems are ordered in columns according to their MAP ranking

Impact on system rankings Moreover, we compare system rankings based on the official SemEval evaluation metrics MAP and F1 against a ranking based on non-obvious F1 scores by indicating the top three systems in Tables 5.6 to 5.8 as **1st**, $\underline{\underline{2^{nd}}}$ and $\underline{3^{rd}}$. Using $F1_n$ results in a different ranking compared to F1, even resulting in a change in the top system in Task B (Talla instead of KeLP) and C (KeLP instead of bunji). It also produces a different ranking than MAP, resulting in a change of the best system for all three tasks (ECNU instead of KeLP for Task A, Talla instead of SimBow for Task B and KeLP instead of IIT-UHH for task C).

5.8 Conclusion

In this chapter, we investigated the role of lexical overlap bias in a range of semantic similarity datasets. We showed that there are apparent differences regarding word overlap between sentence pairs in existing datasets: In several popular datasets, word overlap provided direct clues for class membership (e.g. Quora and MSRP), while other datasets were more resilient to lexical overlap bias (such as SemEval and PAWS). Future work can build on our analysis to avoid lexical overlap bias, for example, by taking the distribution of lexical divergence and gold labels into account during dataset construction. Moreover, our simple lexical overlap baseline can be employed to ensure that new datasets cannot be solved solely by exploiting surface structure similarity.

Furthermore, we presented an automated criterion for distinguishing between easy and difficult items in text pair similarity prediction tasks based on lexical divergence and gold labels. Based on this distinction, we found that more than 50% of cases in current human-created semantic similarity datasets are relatively obvious. We demonstrated that recently proposed models perform significantly worse on non-obvious examples compared to obvious cases. In order to encourage the development of models that perform well on difficult items, we proposed using the non-obvious F1 score ($F1_n$) as a complementary ranking metric for model evaluation.

Part III

Models

CHAPTER 6

A Topic-Informed Approach to Semantic Similarity Detection

6.1 Introduction

Semantic Similarity Detection approaches Semantic Similarity Detection refers to a collection of sentence pair prediction tasks which aim at automatically recognising the presence of a certain semantic relationship (such as semantic equivalence or question-answer relation) between two given texts (refer to Chapter 3.1). A variety of models have been proposed for this task, including traditional feature-engineered techniques (Filice et al., 2017), hybrid approaches (Wu et al., 2017a; Feng et al., 2017; Koreeda et al., 2017) and purely neural architectures (Wang et al., 2017; Tan et al., 2018; Deriu and Cieliebak, 2017), see Chapters 3.2 to 3.6. Recent pre-trained contextualised representations such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have led to impressive performance gains across a variety of NLP tasks, including Semantic Similarity Detection. These models leverage large amounts of data to pre-train text encoders (in contrast to just individual word embeddings as in previous work) and have established a new pretrain-finetune paradigm. While improvements have been achieved with neural models on large-scale generic datasets (Tomar et al., 2017; Gong et al., 2018), Semantic Similarity Detection for smaller domain-specific datasets still remains a challenging problem.

Topics for Semantic Similarity Detection Earlier feature-engineered models have successfully incorporated topic-based features for Semantic Similarity Detection (Qin et al., 2009; Tran et al., 2015; Mihaylov and Nakov, 2016; Wu et al., 2017a). The rationale behind the usefulness of topics is that if two texts share similar topics, they are more likely to be semantically related (Mihaylov and Nakov, 2016). In particular, topics can provide additional clues for identifying semantically

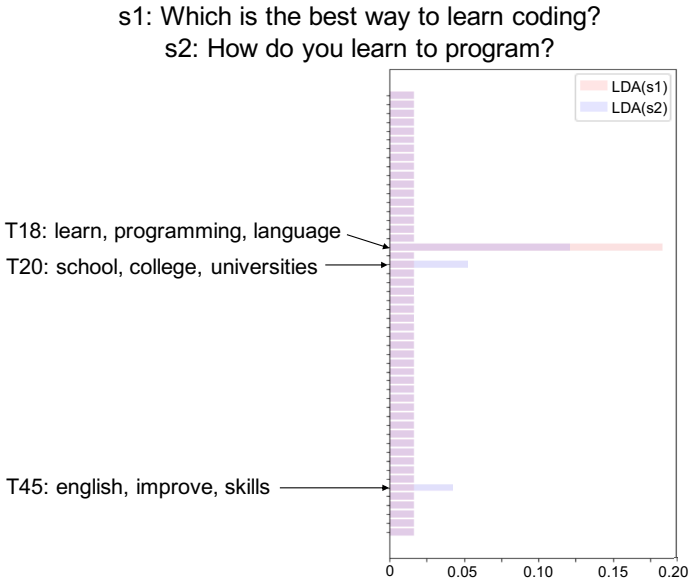


Figure 6.1: Topic distribution for example from the Quora development set with a positive label (is_paraphrase).

related text pairs with little or no direct word overlap as illustrated in Figure 6.1. Recent work on neural architectures has shown that the integration of topics can yield improvements in other tasks such as language modelling (Ghosh et al., 2016), machine translation (Chen et al., 2016), and summarisation (Narayan et al., 2018; Wang et al., 2018). In addition, the fact that topic models have been exploited for domain adaptation (Hu et al., 2014; Guo et al., 2009) suggests that they could also be useful for dealing with domain-specific language.

Research questions In this chapter, we study the following research questions (RQ3 and RQ4 from Chapter 1): *First, how can one develop a generic Semantic Similarity Detection model which performs well across a variety of tasks and datasets? Second, can alternative information sources be exploited to make semantic similarity models more resilient to the identified biases?* This chapter addresses these questions by proposing a novel and generic approach which combines topic models with neural architectures for Semantic Similarity Detection without relying on task-specific or dataset-specific features. The chapter is split into two parts: The first part (section 6.4) introduces and analyses a model which combines topics with an ELMo-enriched CNN-LSTM architecture. The second part of this chapter (section 6.5) builds on the findings of the preceding study and proposes an improved Semantic Similarity Detection framework for combining different topic models with

a pre-trained BERT model.

Contributions Topics have been useful for previous feature-engineered Semantic Similarity Detection models and neural models for other tasks, but there is currently no standard way of combining topics with neural architectures or pre-trained contextual representations. This chapter proposes novel topic-informed models for Semantic Similarity Detection, which do not require dataset-specific features and perform well across a range of challenging tasks. In the first part of this chapter (section 6.4), we introduce a novel topic-aware neural model based on a popular CNN-LSTM architecture, which is further enhanced with ELMo representations. This part is based on our preprint paper (provided as Appendix B) and makes the following contributions:

1. We propose two settings (early and late fusion) for incorporating topics in our neural Semantic Similarity Detection model (section 6.4.1).

2. Our topic-aware model improves over other neural models on multiple semantic similarity datasets, but not BERT, which has been pre-trained on text pair prediction tasks (section 6.4.4).

3. In an ablation study, we highlight the importance of topics and early topic-embedding fusion in our proposed architecture (section 6.4.4).

Based on these findings, the second part of this chapter (section 6.5) examines if adding topics to the state-of-the-art pre-trained neural model BERT can further improve performance for Semantic Similarity Detection. For this purpose, we introduce a novel framework for incorporating different topic models with BERT. This part is based on our ACL 2020 paper (provided as Appendix C) and makes the following contributions:

1. We propose tBERT — a simple architecture for combining topics with BERT for Semantic Similarity Detection (section 6.5.1).[1] Within this framework, we present experimental results for different topic models - LDA vs GSDMM - and topic types - word vs document topics (section 6.5.2).

2. We demonstrate that tBERT achieves improvements across multiple semantic similarity prediction datasets against a finetuned vanilla BERT and other neural models in both F1 and stricter evaluation metrics (section 6.5.7).

[1] Code is available at https://github.com/wuningxi/tBERT.

3. We highlight tBERT's increased resilience to lexical overlap bias in comparison to BERT on an adversarial dataset (section 6.5.7).

4. We show in our error analysis that tBERT's gains are prominent on domain-specific cases, such as those encountered in specialised CQA forums (section 6.5.7).

6.2 Related Work

Topics in feature-engineered models Earlier feature-engineered models have successfully incorporated topics to capture semantic similarity in tasks such as Paraphrase Detection and Answer Selection. For example, Qin et al. (2009) used topic models to measure the semantic distance between each candidate answer and the query in an answer selection task, demonstrating performance improvements over a purely keyword-based approach. A later study by Mihaylov and Nakov (2016) successfully used similarity measures between posts based on bag-of-topic distributions for CQA answer selection. Similarly, Wu et al. (2017a) reported various distance measures and kernel functions based on the topic distributions of two sentences to be helpful for CQA Answer Selection and Paraphrase Detection. Moreover, Tran et al. (2015) experimented with topic-based features obtained from Wikipedia in comparison to domain-specific topic models and found the latter to be beneficial for Answer Sentence Selection.

Topics in neural architectures Recent work on neural architectures has shown that the integration of topics can yield improvements in other tasks such as language modelling, machine translation, and summarisation. For language modelling, Ghosh et al. (2016) proposed a context-enriched LSTM model which incorporates topics of different text segments based on a hierarchical topic model. They modified LSTM cells to incorporate topic distributions of the previous sentence, the current sentence and the whole segment, demonstrating improved performance for next word prediction, next sentence selection and next sentence topic prediction. For e-commerce machine translation, Chen et al. (2016) suggested to enrich the decoder of their recurrent neural network with topics and experimented with human-labeled as well as LDA topics. For abstractive summarisation, Wang et al. (2018) introduced a sequence-to-sequence model which enriches CNN-based encoders and decoders with LDA topics. In a similar approach, Narayan et al. (2018) proposed a summarisation model with topic-enriched gated-linear units as encoders based on distributions obtained from an LDA topic model.

Traditional vs neural topic models The topic-informed neural approaches discussed above employ traditional topic models (see chapter 2.3.4) such as LDA (Blei et al., 2003) which are trained separately from the task-specific model, reducing the overall model complexity and the training time. In contrast, several recent studies (Srivastava and Sutton, 2017; Miao et al., 2018; Gui et al., 2019) have proposed neural topic models and demonstrated strong results in comparison to traditional topic models. Many neural topic modelling approaches focus on Variational Autoencoder (Kingma and Welling, 2014) based architectures or seek to extract topics from pre-trained word or contextual embeddings (Zhu et al., 2020). In contrast to traditional topic models, neural topic models offer the opportunity to be trained jointly with a downstream task to obtain task-specific topics. For example, Gui et al. (2020) proposed a multi-task learning approach for combining a neural topic model with a neural sentiment classification model and demonstrated that the resulting topics reflected semantic information and sentiment polarity. This was achieved by combining the loss of the two models with an additional loss term which encouraged the topic distribution of a word to be similar to the attention vector of the same word from the sentiment model RNN encoder. On the downside, joint neural modelling of topics and downstream tasks comes at the cost of the increased model complexity in comparison to training separate traditional topic models. In this chapter, we only focus on incorporating traditional topic models with neural and pre-trained model components for downstream tasks. However, this can serve as a first step to integrating more complex neural topic models with neural architectures in the future.

Topics for domain adaptation Topic models have been used for domain adaptation in multiple NLP tasks which suggests that topics can be helpful when dealing with domain-specific language. For example, Named Entity Recognition (NER) relies on specific terms and context clues which differ across domains (Guo et al., 2009). This makes it difficult to apply NER systems to a new target domain with different distributions than the source domain and frequently leads to decreased performance. Guo et al. (2009) proposed a topic-based NER system which successfully addresses this domain adaptation problem by leveraging topics to identify semantically similar words - rather than relying on lexical similarity - to overcome the distribution difference between the source and target domain. Another example is machine translation, where a system trained on one domain (e.g. social media) generally performs well for examples within the same domain, but may produce inappropriate translations for examples from a new domain (e.g. news). Hu et al.

(2014) address this problem by proposing a new polylingual tree-based topic model (ptLDA) which they use to capture domain information in their translation model.

6.3 Datasets and Tasks

Datasets This chapter trains and evaluates models on six well-known semantic similarity datasets featuring different sizes (small vs large), tasks (Answer Selection vs Paraphrase Detection) and sentence lengths (short vs long) which were described in detail in chapter 4. Among these are three popular Paraphrase Detection datasets: the general purpose paraphrase dataset **MSRP** (introduced in section 4.2), the duplicate question dataset released by the CQA platform **Quora** (discussed in section 4.3) and the adversarial counterpart to Quora **PAWS** (see section 4.4). We also include three domain-specific CQA datasets originating from the online forum *Qatar Living* which were featured in the SemEval 2017 shared task: the internal Answer Sentence Selection dataset **SemEval A**, the question Paraphrase Detection **SemEval B** dataset and the external Answer Sentence Selection dataset **SemEval C** (described in section 4.5).

Task All of these datasets compare two short texts (usually a sentence long but in some cases consisting of multiple sentences). From here onward we will use the term 'sentence' to refer to each short text. We frame the underlying task as predicting the semantic similarity between two sentences in a binary classification task (see Chapter 3.1 for more details regarding our approach). We use a binary classification setup (rather than ranking) as this is more generic and applies to all above datasets.

6.4 Part I: TAPA Model

The first part of this chapter examines whether we can successfully integrate topic model information in a neural architecture for Semantic Similarity Detection. We also explore how to best combine topics with word embeddings, which are the main information source in existing models (Deriu and Cieliebak, 2017; Pang et al., 2016; Gong et al., 2018).

6.4.1 Architecture

We propose a novel topic-informed neural architecture architecture dubbed **T**opic-**A**ware Text **P**air Prediction **A**rchitecture (TAPA) which is depicted in Figures 6.2 and 6.3. Our model consists of three layers: An encoding layer that obtains topic-aware representations for both sentences (section 6.4.1.1), a comparison layer for collating the obtained sentence representations (6.4.1.2) and an aggregation layer which extracts information for the final prediction (6.4.1.3). This approach is motivated by successful previous architectures (Wang and Jiang, 2016; Gong et al., 2018).

6.4.1.1 Encoding Layer

Word representations Given a sequence of tokens $(w_1, ..., w_N)$ representing a sentence with length N, we map each token to a dense representation, resulting in a sequence of embeddings $\mathbf{x} = (\mathbf{x_1}, ..., \mathbf{x_N})$ (step 1 in Figures 6.2 and 6.3). The representation for each word w_i consists of a combination of pre-trained static GloVe embeddings (Pennington et al. 2014, see Chapter 2.3.2.3) and contextual ELMo embeddings (Peters et al. 2018, see Chapter 2.3.3.1). More specifically, for each word w_i we concatenate the pre-trained static word embedding (indicated by Glove(w_i)) with the pre-trained contextual embedding (denoted by ELMo(w_i)) resulting in a combined word representation $\mathbf{x_i} \in \mathbb{R}^E$:

$$\mathbf{x_i} = [\text{Glove}(w_i); \text{ELMo}(w_i)] \tag{6.1}$$

where ; denotes concatenation and E is the resulting embedding dimension. The reasoning behind this dual representation is that static word embeddings such as GloVe are a proven and useful representation for many NLP tasks. However, as they are not context-sensitive, we further use ELMo representations which take into account the context when computing individual representations to provide the model with two alternative input representations.

Topic representations As topic model, we choose LDA (Blei et al. 2003, see Chapter 2.3.4) which is arguably the most popular and widely used topic model. We train one topic model with T number of topics on the training portion of each individual dataset. The trained topic model is then used to infer a topic distribution (step 2) for the whole sentence (referred to as document-level topics $\mathbf{t_D} \in \mathbb{R}^T$):

$$\mathbf{t_D} = \text{TopicModel}([w_1, ..., w_N]) \tag{6.2}$$

and to infer a topic distribution for every individual token w_i in the sentence (referred to as word-level topics $\mathbf{t'_i} \in \mathbb{R}^T$)

$$\mathbf{t'_i} = \text{TopicModel}(w_i). \tag{6.3}$$

We follow Narayan et al. (2018) in merging word-level and document-level topics to capture the interaction between both as

$$\mathbf{t_i} = [\mathbf{t'_i} \otimes \mathbf{t_D}] \tag{6.4}$$

where $\otimes$ denotes element-wise multiplication as 'word+doc' setting. Narayan et al. (2018)'s previous work focused on a different task (summarisation) with longer texts. In contrast, we reason that for our setting word-level topics may also be informative when used without document topics. Therefore, we further include only word-level topics as 'word' setting

$$\begin{aligned} \mathbf{t_i} &= [\mathbf{t'_i}] \\ \mathbf{t_i} &\in \mathbb{R}^T \end{aligned} \tag{6.5}$$

where the topic setting ('word' or 'word+doc') is treated as a hyperparameter.

Fusion of embeddings and topics We propose two different ways of combining topic distributions with word embeddings: early fusion and late fusion. Early fusion combines topic distributions with word embeddings before the encoder step (Figure 6.2), guiding the encoder in selecting relevant information when computing a representation for the sentence. In contrast, late fusion combines information derived from topics and word embeddings *after* computing separate affinity matrices for topics and word representations (Figure 6.3), therefore introducing the topic information more directly into the architecture as separate sentence interaction dimension. As a result, the encoding layer of both variations differs slightly. In early fusion TAPA, we obtain a sentence representation $\mathbf{e} = (\mathbf{e_1}, ..., \mathbf{e_n})$ by concatenating

topics and embeddings as follows:

$$\begin{aligned} \mathbf{e_i} &= [\mathbf{x_i}; \mathbf{t_i}] \\ \mathbf{e_i} &\in \mathbb{R}^{E+T} \end{aligned} \tag{6.6}$$

Late fusion TAPA only uses embeddings for $\mathbf{e}$

$$\begin{aligned} \mathbf{e_i} &= [\mathbf{x_i}] \\ \mathbf{e_i} &\in \mathbb{R}^{E} \end{aligned} \tag{6.7}$$

and obtains separate topic representations

$$\begin{aligned} \mathbf{T^L} &= [\mathbf{t_1}, ..., \mathbf{t_N}] \\ \mathbf{T^R} &= [\mathbf{t_1}, ..., \mathbf{t_M}] \\ \mathbf{T^L} \in \mathbb{R}^{N \times T}&, \mathbf{T^R} \in \mathbb{R}^{M \times T} \end{aligned} \tag{6.8}$$

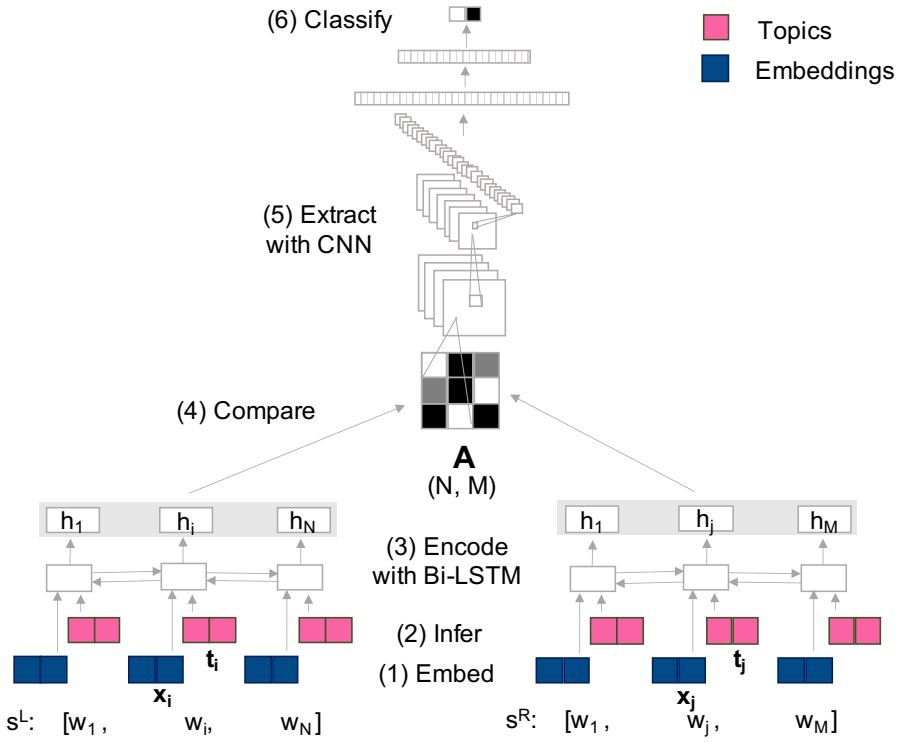


Figure 6.2: Proposed TAPA architecture with early topic-embedding fusion. Illustrated with toy examples consisting of three tokens.

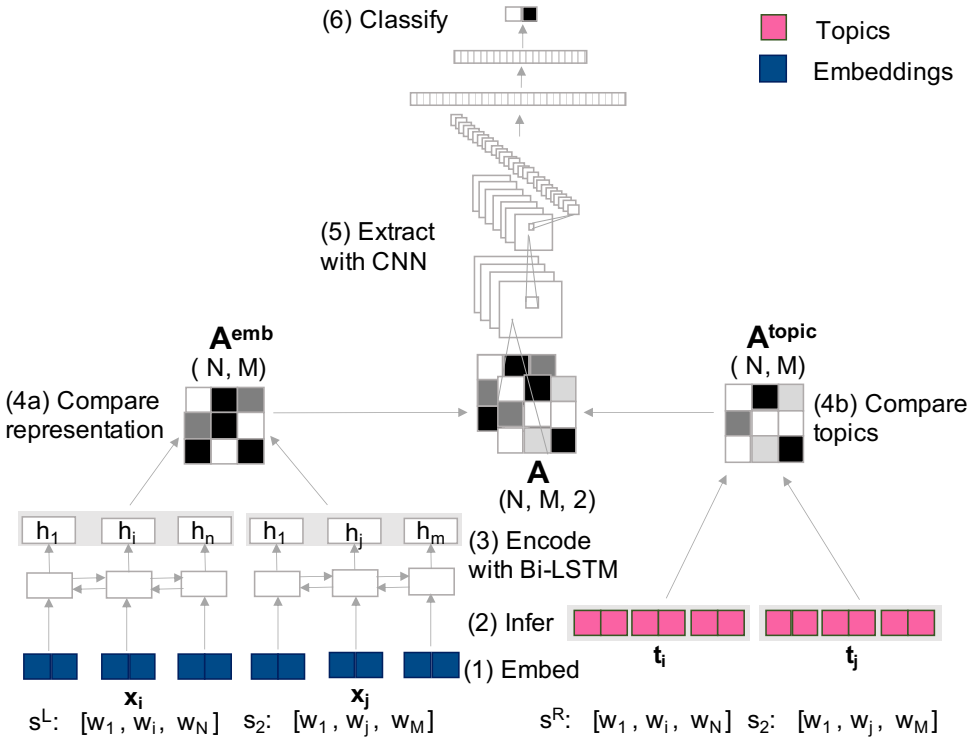


Figure 6.3: Proposed TAPA architecture with late topic-embedding fusion.

Encoder Convolutional Neural Networks (CNNs, see Chapter 2.4.3) and Bidirectional LSTMs (BiLSTMs, see Chapter 2.4.4.2) are both commonly used text encoders. In our preliminary experiments, BiLSTMs worked better than CNN encoders, presumably due to their ability to capture long-range dependencies. As a result, we encode the two sentences s^1 and s^2 represented by the sentence representation sequences $(\mathbf{e_1}, ..., \mathbf{e_N})$ and $(\mathbf{e_1}, ..., \mathbf{e_M})$ with two BiLSTMs (step 3 in Figures 6.2 and 6.3):

$$\begin{aligned} \mathbf{E^L} &= \text{BiLSTM}(\mathbf{e_1}, ..., \mathbf{e_N}) \\ \mathbf{E^R} &= \text{BiLSTM}(\mathbf{e_1}, ..., \mathbf{e_M}) \\ \mathbf{E^L} &\in \mathbb{R}^{N \times D}, \mathbf{E^R} \in \mathbb{R}^{M \times D} \end{aligned} \tag{6.9}$$

where D denotes the dimension of the BiLSTM's hidden layer. We decide to share weights between both BiLSTMs in a Siamese setting which reduces the number of required parameters and has been shown to work well for pairwise classification tasks (Deriu and Cieliebak, 2017; Feng et al., 2015; Mueller and Thyagarajan, 2016).

6.4.1.2 Comparison Layer

We model the similarity between the two encoded sentences by computing pairwise affinity scores between their words in affinity matrices similar to previous studies (Deriu and Cieliebak, 2017; Pang et al., 2016; Gong et al., 2018). In both early and late fusion TAPA, we calculate the affinity matrix $\mathbf{A^{emb}} \in \mathbb{R}^{n \times m}$ based on the encoded sentences as

$$\mathbf{A}^{emb}_{i,j} = s(\mathbf{E^L}_{i,:}, \mathbf{E^R}_{j,:}) \tag{6.10}$$

using a similarity function s (step 4 in Figure 6.2 and 4a in Figure 6.3). For the late fusion model, we also calculate an affinity matrix $\mathbf{A^{topic}} \in \mathbb{R}^{n \times m}$ based on the topic distributions in both sentences (4b in Figure 6.3):[2]

$$\mathbf{A}^{topic}_{i,j} = s(\mathbf{T^L}_{i,:}, \mathbf{T^R}_{j,:}) \tag{6.11}$$

Common choices for the similarity function s are Euclidean distance, dot product and cosine similarity (Yin et al., 2016; Deriu and Cieliebak, 2017). We use cosine similarity, as it worked best in preliminary experiments. The comparison layer output $\mathbf{A}$ is simply $\mathbf{A^{emb}}$ for early fusion TAPA, while we combine topic and embedding affinity matrices in the late fusion version:

$$\mathbf{A} = [\mathbf{A^{emb}}; \mathbf{A^{topic}}]. \tag{6.12}$$

6.4.1.3 Aggregation Layer

Similar to extracting information from a grey scale image in computer vision, we follow Gong et al. (2018) and Pang et al. (2016) in aggregating useful affinity patterns from $\mathbf{A}$ with a CNN (LeCun et al., 1998a) feature extractor (step 5 in in Figure 6.2 and 6.3). We use a two-layer CNN architecture (where one layer consists of a convolution and a pooling operation). The output of the last convolution layer is flattened into a vector. This is followed by multiple hidden layers of reducing size and a softmax layer for calculating class probabilities (step 6 in Figure 6.2 and 6.3). The model is trained using cross-entropy loss. We tune the hyperparameters with hyperopt (Bergstra et al., 2013). Tuned hyperparameters and further implementation details are reported in section 6.4.2.

[2] We experimented with additional topic encoders, but abandoned them as they did not consistently improve the results.

Dataset	# of filters	filter size	# of hidden layers	Batch size	Learning rate	Optimizer	Embd
Quora	(4,12)	(2,2)	2	64	.050	adadelta	Glove
PAWS	(4,12)	(2,2)	2	64	.050	adadelta	Glove
SemEval B	(0,0)	(0,0)	2	10	.100	adadelta	Deriu
SemEval C	(0,0)	(0,0)	0	10	.002	adam	Glove

Table 6.1: General model hyperparameters obtained after tuning on development set. 'Filter' refer to the filters of the CNN feature extractor. 'Deriu' embedding refers to the embeddings which were trained on the SemEval dataset and released by Deriu and Cieliebak (2017).

Dataset	Topic type	# of topics	Topic alpha	Topic update	Fusion
Quora	word	70	50	True	early
PAWS	word	70	50	True	early
SemEval B	word	90	0.1	True	early
SemEval C	word	80	10	True	late

Table 6.2: Topic-related hyperparameters obtained after tuning on the development set.

6.4.2 Implementation Details

All words were lower-cased during preprocessing. Embeddings are initialised with pre-trained 300 dim Glove embeddings (Pennington et al.,2014) which were updated during training. We trained LDA topic models (Blei et a.,2003) on the training set of each dataset using *ldamallet* from the Gensim package and experimented with static vs updated topic distributions, different alpha values (0.1 to 50) and number of topics (10 to 100) which are treated as hyperparameters of the proposed TAPA model. As PAWS is based on Quora, we used Quora topic models for both datasets. We tune hyperparameters with the tree of Parzen estimators algorithm implemented in Hyperopt (Bergstra et al., 2013) based on dev set F1, see Tables 6.1 and 6.2 for our best hyperparameters.

6.4.3 Baselines

Previous systems As baseline systems, we provide the top three SemEval 2017 models (based on F1 score of primary submission). For subtask B, these are KeLP (Filice et al., 2017), NLM NIH (Abacha and Demner-Fushman, 2017) and UinsuskaUiTech (Agustian and Takamura, 2017). For subtask C, the top systems are BUNJI (Koreeda et al., 2017), IIT-UHH (Nandi et al., 2017) and KeLP (Filice et al., 2017). All of these models use hand-crafted features, which provides an advantage

on this small dataset (see Chapter 4 for dataset sizes). Moreover, the systems are task-dependent as features and settings of the same system vary between subtasks B and C. For Quora, we provide Shen et al. (2018b)'s simple word embedding-based model, Pang et al. (2016)'s CNN-based MatchPyramid, as well as Gupta and Zhang (2018)'s TreeLSTM. We note that results for other systems (Gong et al., 2018; Tan et al., 2018; Tomar et al., 2017) have been reported on Quora with accuracies ranging between .75 and .89, but are not directly comparable to F1 score.

Topic baseline We also provide a simple topic baseline which calculates the Jensen-Shannon Divergence (JSD, Lin 1991) between two topic distributions to predict a negative label if JSD is larger than a certain threshold and a positive label otherwise. Threshold and topic hyperparameters are chosen based on development set F1.

Siamese network In addition to the above mentioned systems, we also compare our proposed model with a Siamese network, as this is a common neural baseline for sentence pair prediction tasks (Wang et al., 2017). The two questions are embedded with pre-trained 300 dimensional Glove embeddings (Pennington et al., 2014) and encoded by two weight-sharing BiLSTM encoders. This is followed by a max pooling layer and two hidden layers. For Siamese network +ELMo, we concatenate Glove embeddings with ELMo representations before encoding them with BiLSTMs.

BERT BERT (Devlin et al. 2019, see Chapter 2.3.3.2) was released after the work on our proposed model was already completed, but due to its wide-ranging success and importance it is provided as an additional baseline. We follow standard practice by encoding the sentence pair with BERT and using its **c** vector followed by a softmax layer to predict the class. We finetune all layers for three epochs with early stopping and tune learning rates on the development set of each dataset.

6.4.4 Results

We evaluate the model performance on the basis of F1 scores as this is more reliable for datasets with label imbalance than accuracy and we present the results in Table 6.3.

Overall performance TAPA performs better than the other neural baselines (Siamese networks, TreeLSTM and MatchPyramid) but is outperformed by BERT. We reason that BERT's advantage over TAPA is due to the fact that BERT is a

	Type	PAWS	Quora	SemEval	
				B	C
Published systems					
BUNJI (Koreeda et al., 2017)	feature-based	-	-	-	.197
IIT-UHH (Nandi et al., 2017)	feature-based	-	-	.390	.144
KeLP (Filice et al., 2017)	feature-based	-	-	**.506**	.121
NLM NIH (Abacha and Demner-Fushman, 2017)	feature-based	-	-	.473	-
UinsuskaTiTech (Agustian and Takamura, 2017)	feature-based	-	-	.467	-
SWEM (Shen et al., 2018b)	embedding-based	-	.830	-	-
MatchPyramid (Pang et al., 2016)	neural	-	.829	-	-
TreeLSTM (Gupta and Zhang, 2018)	neural	-	.719	-	-
BERT (Devlin et al., 2019)	pretrained	**.440**	**.902**	.473	**.268**
Our systems					
Topic baseline	feature-based	.370	.736	.436	.096
Siamese BiLSTM	neural	.173	.813	.349	.126
Siamese BiLSTM +ELMo	neural	.372	.832	.345	.149
TAPA	neural	**.422**	**.841**	**.464**	**.152**

Table 6.3: Performance (F1) of proposed TAPA model on test set. The first five rows are taken from the official SemEval 2017 results Nakov et al. (2017). Rows six to eight are taken from the corresponding publication and the rest are our own implementations.

larger and fully pre-trained architecture (except for the weights in the output layer) that has been optimised on sentence pair prediction tasks. In contrast, most parts of our architecture (encoder, comparison and aggregation layer) need to be trained from scratch and the ELMo representations have only been pre-trained on next word prediction. In comparison to the feature-engineered models, TAPA cannot directly compete with the highest ranked SemEval systems (KeLP for Task B and BUNJI for task C) due to the lack of sufficient training data (compare dataset sizes in Chapter 4). However, it outperforms the second ranked system on SemEval C (IIT-UHH) and gets within reach of the third placed system on SemEval B (Uinsuska TiTech). Our neural architecture may not outperform the best feature-engineered models, but it can generalise better across datasets, while the top SemEval systems require dataset-specific feature adjusting for each subtask.

Do topics help? Our topic-aware TAPA model improves over the Siamese baselines without topic information on all datasets (see Table 6.3). The largest gains of TAPA are observed on SemEval B - the dataset where the strong performance

Type	PAWS	Quora	SemEval B	SemEval C
full TAPA (early fusion)	.422	.841	.464	.152
-topics	.406	.839	.451	.094
-ELMO	.269	.845	.450	.124
-fusion type	.398	.839	.401	.112

Table 6.4: Ablation study for our TAPA model reporting F1 scores for model without certain components.

of the topic baseline indicated that topics are particularly helpful. Furthermore, our ablation study (Table 6.4) shows that removing topics consistently reduces F1 scores on all datasets. In comparison, the effect of ELMo representations is dataset dependent. Removing context-aware ELMo representations slightly improves performance on Quora, but leads to a large performance drop on PAWS. The large impact on PAWS can be explained by the fact that this dataset was automatically constructed to have high textual overlap between questions and differences between paraphrases are chiefly due to variations in syntax.

What is the best way to integrate topics in our neural architecture? Hyperparameter tuning (see section 6.4.2) selected word topics and early fusion for TAPA on all datasets which suggests that these are good settings for the model. Our full TAPA model therefore uses word topics with early fusion. When comparing the full (early fusion) model with a tuned[3] late fusion variant (last line in Table 6.4), we find that performance of the late fusion model drops to the same or even lower level than TAPA without topics. We conclude that topics contribute consistently to the performance of our proposed model, but that early word topic-embedding fusion is crucial.

[3]As late fusion may require slightly different hyperparameters, we compare with a tuned late fusion model to make a fair comparison between early and late fusion.

6.5 Part II: tBERT Model

The first part of this chapter indicated that the addition of topics is beneficial for neural Semantic Similarity Detection models. However, our proposed partially pre-trained model cannot compete with the recently released contextualised BERT (Devlin et al., 2019) model, which has been fully pre-trained with text pair prediction tasks. In the second part of this chapter, we therefore investigate whether topic models can also improve BERT's performance for Semantic Similarity Detection tasks. We propose a generic framework for combining topics with BERT, compare the effectiveness of different model settings and experiment with a wider range of datasets and topic models.

6.5.1 Architecture

Our proposed **t**opic-informed **BERT**-based model (tBERT) is illustrated in Figures 6.4 and 6.5. The framework consists of three steps: obtaining a BERT-based text pair representation (section 6.5.1.1), obtaining a topic representation for each text (section 6.5.1.2) and combining both representations to predict the class (section 6.5.1.3).

6.5.1.1 Text pair representation

BERT Given two sentences in the form of token sequences $s^1 = (t_1, ..., t_N)$ with length N and $s^2 = (t'_1, ..., t'_M)$ with length M, we obtain a context-aware sentence pair representation by encoding both sentences with the uncased version of $\text{BERT}_{\text{BASE}}$ (Devlin et al. 2019, see Chapter 2.3.3.2). We follow standard practice by feeding the sentences (concatenated with '[CLS]' and '[SEP]' tokens) and using the $\mathbf{c} \in \mathbb{R}^D$ vector from BERT's final layer which corresponds to the '[CLS]' token in the input:

$$\mathbf{c} = \text{BERT}^{final}([[\text{CLS}]; s^1; [\text{SEP}]; s^2]) \tag{6.13}$$

where D denotes the internal hidden size of BERT (768 for $\text{BERT}_{\text{BASE}}$).

6.5.1.2 Topic representation

Topic model While other topic models can be used within our framework, we experiment with two well-known traditional topic models in this work: LDA (Blei

et al., 2003) and GSDMM (Yin and Wang, 2014).[4] We chose traditional topic models (see Chapter 2.3.4), rather than neural topic models as they are well-studied, lightweight and fast to train. Although neural topic models offer the opportunity to obtain task-specific topics through joint training, we focus on illustrating the effectiveness of our general framework based on separately trained traditional topic models as a starting point and leave more sophisticated topic models for future work.

Topic types Previous research (Narayan et al., 2018) has successfully combined word- and document-level topics with neural architectures for summarisation by combining both topic types through an element-wise multiplication operation. In Chapter 6.4.4 we experimented with this fusion of word and document topics for Semantic Similarity Detection, but found that mere word topics worked better than a combination of both. In this work, we therefore experiment with word and document topics separately (instead of combining them both).

tBERT with document topics When using document topics $\mathbf{d^1}$ and $\mathbf{d^2}$ (illustrated in Figure 6.4), all tokens in a sentence are passed to the topic model to infer

[4]Refer to section 6.5.2 for more details and topic model hyper-parameters. We use the same topic models as in the topic baseline.

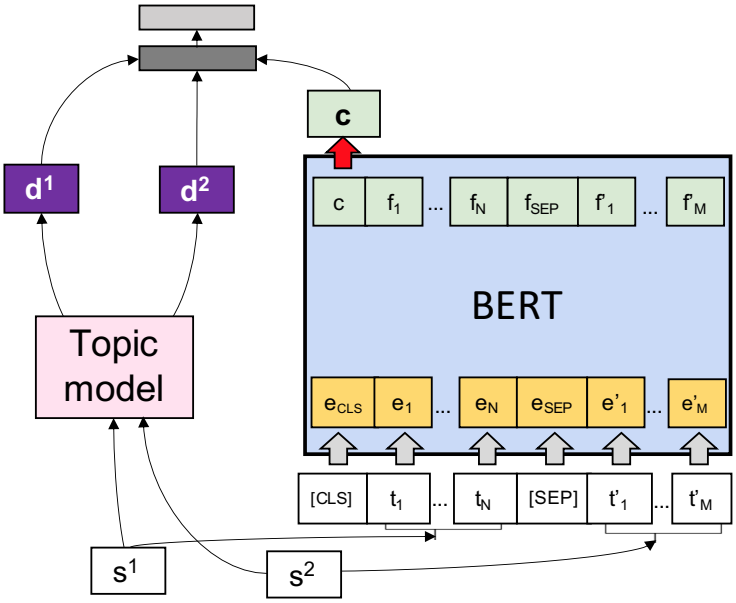


Figure 6.4: Proposed tBERT architecture with document topics ($\mathbf{d^1}$ and $\mathbf{d^2}$).

one topic distribution per sentence:

$$\begin{aligned} \mathbf{d^1} &= \text{TopicModel}([t_1, ..., t_N]) \\ \mathbf{d^2} &= \text{TopicModel}([t'_1, ..., t'_M]) \\ \mathbf{d^1}, \mathbf{d^2} &\in \mathbb{R}^T \end{aligned} \tag{6.14}$$

where T indicates the number of topics.

tBERT with word topics Alternatively, we can use word topics instead of document topics. When using word topics $\mathbf{w^1}$ and $\mathbf{w^2}$ (illustrated in Figure 6.5), one topic distribution $\mathbf{w_i} \in R^T$ is inferred per token t_i

$$\mathbf{w_i} = \text{TopicModel}(t_i). \tag{6.15}$$

Then, we average all word topics in the sentence to obtain a fixed-length topic representation on the sentence level:

$$\begin{aligned} \mathbf{w^1} &= \frac{\sum_{i=1}^{N} \mathbf{w_i}}{N} \\ \mathbf{w^2} &= \frac{\sum_{i=1}^{M} \mathbf{w'_i}}{M}. \\ \mathbf{w^1}, \mathbf{w^2} &\in \mathbb{R}^T \end{aligned} \tag{6.16}$$

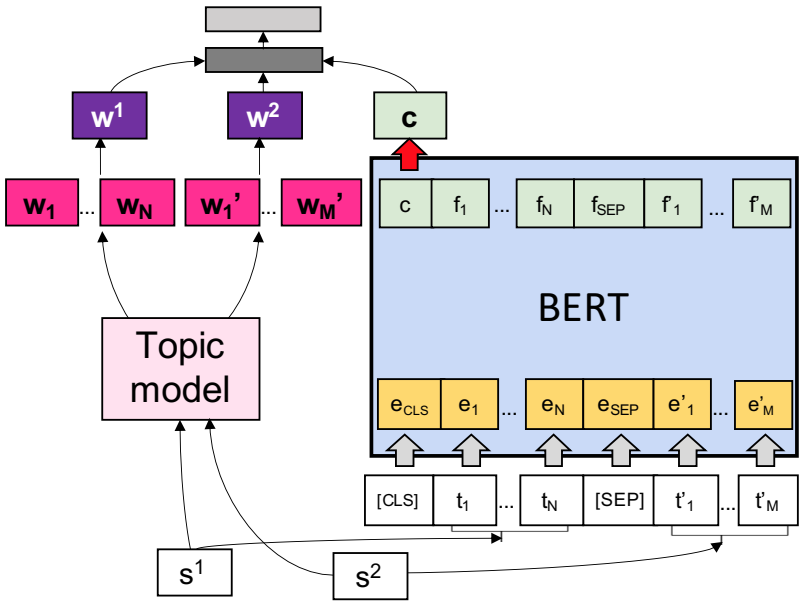


Figure 6.5: Proposed tBERT architecture with word topics $\mathbf{w^1}$ and $\mathbf{w^2}$.

6.5.1.3 Representation combination and prediction

Motivated by Ostendorff et al. (2019)'s combination method for Knowledge Graph Embeddings with BERT, we combine the sentence pair vector with the sentence-level topic representations as

$$\begin{aligned} \mathbf{f} &= [\mathbf{c}; \mathbf{d^1}; \mathbf{d^2}] \\ \mathbf{f} &\in \mathbb{R}^{D+2T} \end{aligned} \tag{6.17}$$

for document topics and as

$$\begin{aligned} \mathbf{f} &= [\mathbf{c}; \mathbf{w^1}; \mathbf{w^2}] \\ \mathbf{f} &\in \mathbb{R}^{D+2T} \end{aligned} \tag{6.18}$$

for word topics (where ; denotes concatenation). This is followed by a hidden layer $\mathbf{h} \in \mathbb{R}^{(D+2T)/2}$ and a softmax classification layer to compute class probabilities. We train the model with cross-entropy loss for three epochs with small learning rates (2e-5, 3e-5 or 5e-5) as recommended by Devlin et al. (2019) and use early stopping based on development set F1 score to prevent overfitting. Learning rates are tuned per dataset and random seed based on development set performance.

6.5.2 Topic Model Settings

We train topic models separately from the neural model components on the training portion of the individual datasets with the exception of the SemEval tasks, where we combine the training data from all three subtasks. We use minimal text preprocessing, employing only tokenisation and filtering of common stopwords based on the NLTK toolkit (Bird and Loper, 2004).

Topic model and topic type We use LDA (Blei et al. 2003, see Chapter 2.3.4.1) because it is a well-known and widely used topic model. However, as LDA has been reported to be less suitable for short text (Hong and Davison, 2010; Vayansky and Kumar, 2020), we also experiment with the popular short text topic model GSDMM (Yin and Wang 2014, see Chapter 2.3.4.2) which assumes only one topic per document. Although tBERT with GSDMM topic models performed well on the development set, GSDMM topics appear to be less consistent (compare topic key words in Tables 6.7-6.12). To select the best setting for our final model (in Table 6.14), we evaluated different combinations of tBERT with LDA vs GSDMM and word ($\mathbf{w^1}$ and $\mathbf{w^2}$) vs document topics ($\mathbf{d^1}$ and $\mathbf{d^2}$) on the development partition of

	MSRP	Quora	SemEval		
			A	B	C
BERT	.906	.906	.714	.754	.414
tBERT with GSDMM					
+ word topics	**.918**	.908	**.752**	**.760**	**.447**
+ doc topics	.915	**.909**	.751	.760	.424
tBERT with LDA					
+ word topics	.905	**.911**	.744	**.766**	**.439**
+ doc topics	**.907**	.909	**.748**	.761	.419

Table 6.5: Performance (F1) of proposed tBERT models with different topic settings on development set. We report average performance for two different random seeds. Bold indicates the selected setting for our final model.

the datasets (Table 6.5). The tBERT settings generally scored higher than BERT, with word topics ($\mathbf{w^1}$ and $\mathbf{w^2}$) usually outperforming document topics.

Topic number and alpha value The number of topics and alpha values are important topic model hyper-parameters and dataset dependent. In contrast to the first part of this chapter (section 6.4.2), we tune the number of topics and alpha value separately from the main model based on the simple topic baseline (section 7.5.2) as a fast proxy (on average 12 seconds per experiment on CPU). This has the benefit of identifying useful topic models for each dataset without expensive hyper-parameter tuning on the full tBERT model. In our experiments, 70 to 90 topics with alpha values of 1 or 10 worked well. Topic model hyper-parameters were chosen based on development set F1 scores of the topic baseline. For each dataset we tried number of topics: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 and alpha values: 0.1, 1, 10, 50 (Figure 6.6 illustrates the influence of different alpha values on topic distributions for four Quora development set examples). For GSDMM, we used the same number of topics as LDA for better comparison and set the beta value to 0.1 based on the original settings in Yin and Wang (2014). The topic baselines and tBERT use topic models with the same hyper-parameters as listed in Table 6.6.

	MSRP	Quora	SemEval		
			A	B	C
# of topics	80	90	70	80	70
LDA alpha values	1	1	50	10	10
GSDMM alpha values	0.1	0.1	0.1	0.1	0.1

Table 6.6: Tuned hyper-parameters of topic models.

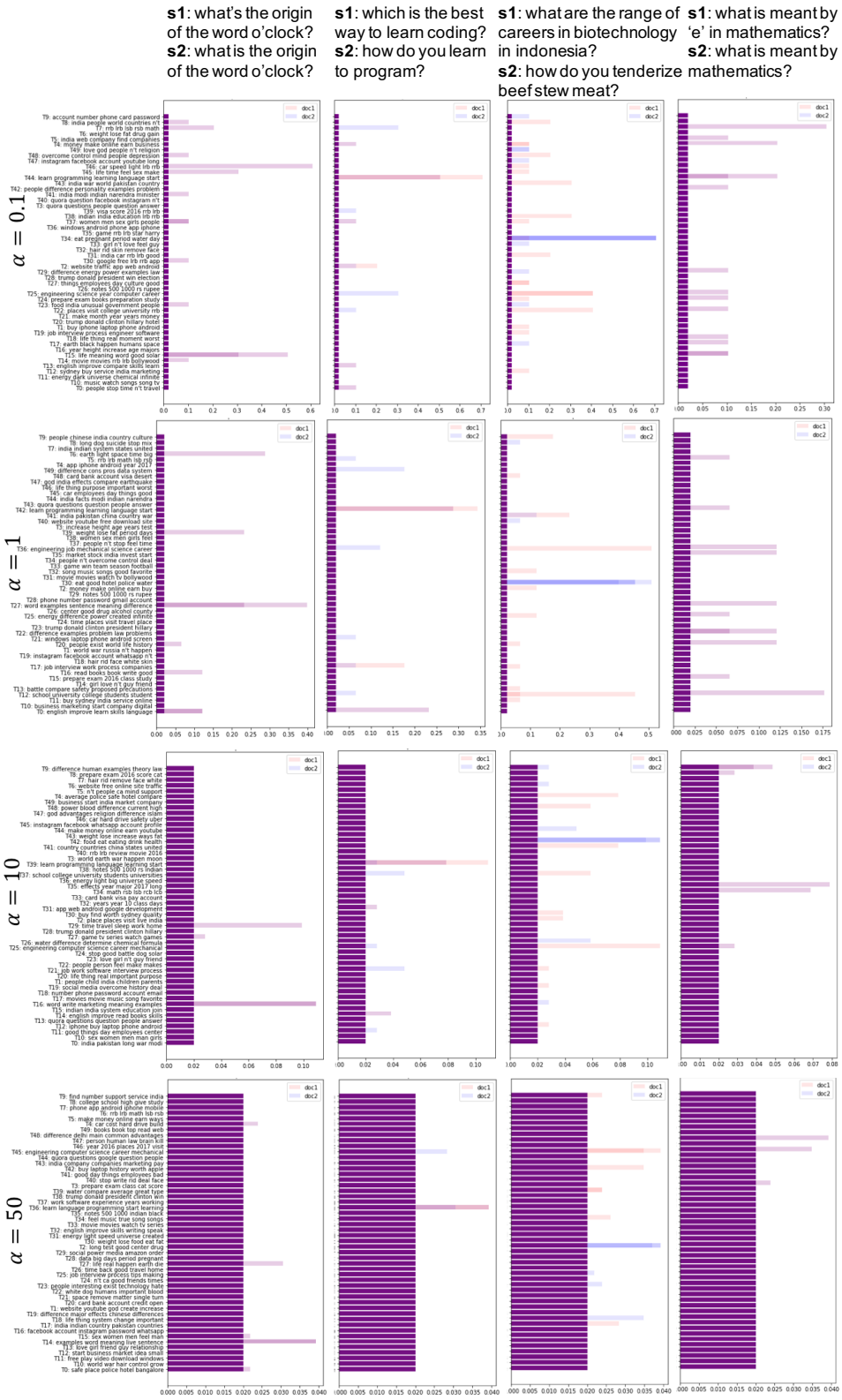


Figure 6.6: LDA word topic distributions for four examples from the Quora development set using different alpha values.

6.5.3 GSDMM Topic Model Examples

T1:	difference examples law social science
T2:	effects earthquake major compare cambodia
T3:	arbitration court cards australia world
T4:	panel solar provider installation california
T5:	get best rid skin remove

Table 6.7: Top key words for example topics learned by GSDMM model on Quora. Inconsistent key words are highlighted in red.

T1:	cases said number year reported sales meeting
T2:	states united wrong sense deal
T3:	two killed united states people government
T4:	condition hospital center taken medical county
T5:	charges commission arrested exchange

Table 6.8: Top key words for example topics learned by GSDMM model on MSRP. Inconsistent key words are highlighted in red.

T1:	know qatar years many indian qatari
T2:	good qatar live doha know dog
T3:	arabic doha best people time english
T4:	month like 000 car compound villa
T5:	time find visa working company study

Table 6.9: Top key words for example topics learned by GSDMM model on the training set of all three SemEval tasks combined. Inconsistent key words are highlighted in red.

6.5.4 LDA Topic Model Examples

T1:	life purpose important thing real biggest
T2:	drink water coffee tea drinking good
T3:	flight car ticket train fly
T4:	school university college high students student
T5:	chemical determine formula acid determined

Table 6.10: Top key words for example topics learned by LDA model on Quora.

T1:	regiment cavalry north 3rd passenger fort
T2:	court judge federal district supreme file
T3:	windows server software microsoft 2003 system
T4:	president bush time presidential report george
T5:	hospital condition study risk cancer women

Table 6.11: Top key words for example topics learned by LDA model on MSRP.

T1:	gym club pool fitness gyms swimming
T2:	drink good club music night alcohol
T3:	husband sponsorship wife company sponsor work
T4:	day eid holidays days ramadan hours
T5:	time doha bus area morning early

Table 6.12: Top key words for example topics learned by LDA model on the training set of all three SemEval tasks combined.

6.5.5 Baselines

Topic baselines As a simple baseline, we train a topic model (LDA or GSDMM) on the training portion of each dataset (combining training sets for SemEval subtasks) and calculate the Jensen-Shannon divergence (Lin 1991, JSD) between the topic distributions of the two sentences. The model predicts a negative label if JSD is larger than a threshold and a positive label otherwise. We tune the threshold, the number of topics and the alpha value based on development set F1. Refer to section 6.5.2 for more details regarding the topic model settings.

Previous systems For the SemEval CQA dataset, we compare against the highest performing system for each subtask based on F1 score of primary submission: ECNU (Wu et al., 2017a), KeLP (Filice et al., 2017) and BUNJI (Koreeda et al., 2017). As these models predominantly rely on hand-crafted dataset-specific features which provide an advantage on the small datasets, we also include the only neural system without manual features (Deriu and Cieliebak, 2017). For MSRP, we show a CNN-based text-matching architecture (Pang et al., 2016). For Quora, we provide Gupta and Zhang (2018)'s attention-enriched Tree-LSTM which utilises syntactic constituent trees. We further compare with the popular Densely Interactive Inference Network (Gong et al., 2018) using accuracy, as no F1 has been reported. In addition, we provide Shen et al. (2018b)'s simple word embedding-based model on both datasets.

Siamese BiLSTM Siamese networks are a common neural baseline for sentence pair classification tasks (Yih et al., 2011; Wang et al., 2017). For our Siamese network implementation, we embed both sentences with pre-trained GloVe embeddings (concatenated with ELMo for BiLSTM + ELMo) and encode them with two weight-sharing BiLSTMs, followed by max pooling and hidden layers.

BERT We encode the sentence pair with BERT's **c** vector (as in tBERT) followed by a softmax layer and finetune all layers for three epochs with early stopping. Following Devlin et al. (2019), we tune the learning rates on the development set of each dataset. Refer to section 6.5.6 for detailed hyperparameter settings.

6.5.6 Hyperparameters

Table 6.13 reports tuned hyper-parameters used in the tBERT models and the BERT baseline. The learning rate was tuned based on development set F1 score per seed and model using grid search (2e-5, 3e-5 or 5e-5). The batch size was generally set to 32, but had to be reduced for SemEval A and C to fit data on a single GPU due to longer sentences.

	MSRP	Quora	SemEval		
			A	B	C
batch size	32	32	16	32	16
BERT					
learning rate (1st seed)	5e-5	2e-5	3e-5	2e-5	2e-5
learning rate(2nd seed)	5e-5	2e-5	2e-5	2e-5	3e-5
tBERT					
learning rate (1st seed)	3e-5	3e-5	2e-5	2e-5	3e-5
learning rate (2nd seed)	5e-5	2e-5	2e-5	3e-5	2e-5

Table 6.13: Tuned hyper-parameters for BERT-based models.

6.5.7 Results

Evaluation setup We evaluate the systems based on F1 scores (Table 6.14) as this is more reliable for datasets with imbalanced labels (e.g. SemEval C, see information regarding label distributions in Chapter 4) than accuracy. We further report performance on difficult cases with non-obvious F1 score which identifies challenging instances in the datasets based on lexical overlap and gold labels (see Chapter 5). Dodge et al. (2020) recently showed that early stopping and random seeds can have considerable impact on the performance of finetuned BERT models.

	Type	F1					non-obvious F1				
		MSRP	Quora	SemEval			MSRP	Quora	SemEval		
				A	B	C			A	B	C
Previous systems											
KeLP (Filice et al., 2017)	feature-based	-	-	.698	.506	.121	-	-	.628	.199	.054
ECNU (Wu et al., 2017a)	feature-based	-	-	**.777**	.503	.136	-	-	.707	.235	.022
BUNJI (Koreeda et al., 2017)	feature-based	-	-	.725	-	.197	-	-	.672	-	.028
SwissAlps (Deriu and Cieliebak, 2017)	neural	-	-	.433	-	-	-	-	.352	-	-
SWEM (Shen et al., 2018b)	embedding-based	.813	.830	-	-	-	-	-	-	-	-
MatchPyramid (Pang et al., 2016)	neural	.829	-	-	-	-	-	-	-	-	-
TreeLSTM (Gupta and Zhang, 2018)	neural		.719	-	-	-	-	-	-	-	-
DIIN (Gong et al., 2018) (accuracy)	neural	-	(.891)	-	-	-	-	-	-	-	-
Our implementation											
GSDMM topic baseline	feature-based	.796	.679	.663	.403	.102	.769	.448	.488	.130	.015
LDA topic baseline	feature-based	.799	.736	.684	.436	.096	.780	.606	.684	.172	.019
Siamese BiLSTM	neural	.763	.813	.671	.349	.126	.781	.740	.597	.168	.049
Siamese BiLSTM + ELMo	neural	.765	.832	.661	.345	.149	.775	.754	.599	.180	.073
BERT (Devlin et al., 2019)	neural	.876	.902	.704	.473	.268	.827	**.860**	.656	.243	.085
tBERT with GSDMM topics	neural	.883	**.905**	.766	.518	.233	.844	.856	**.714**	**.266**	.081
tBERT with LDA topics	neural	**.884**	**.905**	.768	**.524**	**.273**	**.866**	.859	.708	.258	**.100**

Table 6.14: Performance (F1 and F1n) of proposed tBERT model on test set. The first 8 rows are taken from the cited papers. Bold font highlights the best system overall and our best implementation is underlined.

We therefore use early stopping during finetuning and report average model performance across two runs with different seeds for the BERT and tBERT models.

Overall trends The simple topic baselines perform surprisingly well compared to much more sophisticated models, indicating the usefulness of topics for the tasks. Contrary to our initial expectation, the LDA topic baseline performs better than the GSDMM baseline. The GSDMM topic model seems to have learned less consistent topics (see section 6.5.2) and may require additional hyperparameter tuning for better performance. The BERT-based models outperform the other neural systems as well as feature-engineered SemEval systems on SemEval B and C, while closely competing on the relatively small SemEval A dataset. It is worth pointing out that the feature-based systems KeLP, ECNU and BUNJI exploit highly dataset-specific metadata features (e.g. the post's position in the thread or the author of the post), while the BERT-based systems only rely on generic information sources which are applicable across all datasets.

Do topics improve BERT's performance? Adding LDA topics to BERT consistently improves F1 performance across all datasets. Moreover, it improves the performance on non-obvious cases over BERT on all datasets (except for Quora, which contains many generic examples and few domain-specific cases, see Table 6.15). The addition of GSDMM topics to BERT is slightly less stable: improving

	MSRP	Quora	SemEval		
			A	B	C
F1 on cases with **named entities** (total: 230/500)					
BERT	.20	**.54**	.50	**.53**	.32
tBERT	**.35**	.49	**.52**	.21	**.56**
(# of cases)	(23)	(31)	(58)	(60)	(58)
F1 on cases with **domain-specific words** (total: 159/500)					
BERT	.18	.00	.36	.36	.26
tBERT	**.67**	**.50**	**.62**	**.40**	**.58**
(# of cases)	(14)	(7)	(36)	(41)	(61)
F1 on cases with **non-standard spelling** (total: 53/500)					
BERT	.00	N/A	.20	**.71**	.43
tBERT	.00	N/A	**.80**	.00	**.62**
(# of cases)	(1)	(0)	(20)	(19)	(13)

Table 6.15: Performance (F1) for BERT and tBERT on annotated development set examples (100 cases per dataset) by manually annotated properties. Number of cases in parenthesis.

s1	Are there good beaches in the Northern part of Qatar?
s2	Fuwairit is very clean !
gold label	True
predictions	BERT:False, BERT+topics:True
manual annotation	domain-specific word:True, named entity:True, non-standard spelling:False

Table 6.16: Predictions and annotation for an example from SemEval.

performance on MSRP, SemEval A and B, while dropping on SemEval C. However, the higher performance of tBERT with LDA topics in comparison to GSDMM topics is in line with the trend of the LDA and GSDMM topic baselines. The largest performance gains regardless of the chosen topic model are observed in the internal question-answering task (SemEval A).

Where can topics help? We randomly sampled 100 examples (half only correct by BERT, half only correct by LDA-tBERT) from the development set of each dataset and manually annotated them (500 in total for all datasets) with binary labels regarding three properties that may be associated with topic-related gains or losses (Table 6.15). Named entities (e.g. *iPhone*) and domain-specific words (e.g. *murabaha*) occurred frequently in the datasets, while there were too few examples with non-standard spelling (e.g. *thanx*) for meaningful comparisons. tBERT generally performed better than BERT on examples with domain-specific cases. Overall patterns were less clear for named entities. Based on manual inspection, BERT dealt better with common named entities likely to have occurred in pre-training (such as well-known brands), while tBERT improved on dataset-specific named entities. We reason that for domain-specific words which are unlikely to have occurred in pre-training (e.g. *Fuwairit* in Table 6.16), BERT may not have learned a good representation (even after finetuning) and hence cannot make a correct prediction. Here, topic models could serve as an additional source for dataset-specific information. The usefulness of topics for such cases is also supported by previous work, which successfully leveraged topics for domain adaptation (Hu et al., 2014; Guo et al., 2009), see section 6.2.

Could we just finetune BERT longer? Based on our observation that tBERT performs better on dataset-specific cases, one could assume that BERT may simply need to be finetuned longer than the usual three epochs to pick up more domain-specific information. In an additional experiment, we finetuned BERT and tBERT (with LDA topics) for 9 epochs (see Figure 6.7). On most datasets, BERT reached

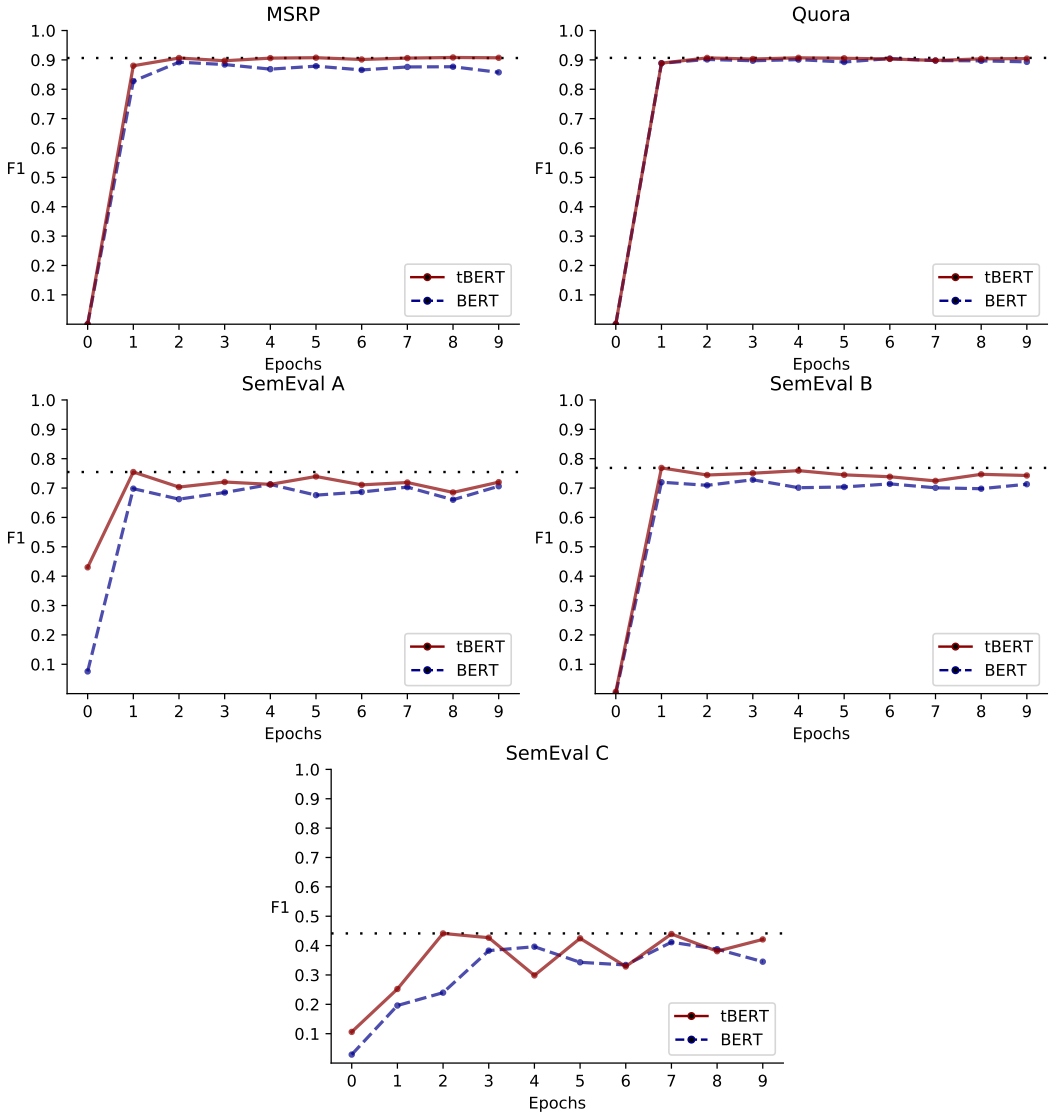


Figure 6.7: Performance of BERT and tBERT on dev set when trained for up to 9 epochs. The dotted line indicates tBERT's best performance within the first three epochs.

	MSRP	Quora	SemEval		
			A	B	C
BERT					
3 epochs	13	839	223	26	340
9 epochs	44	2710	638	75	1047
tBERT					
3 epochs	13	885	211	24	348
9 epochs	42	2916	658	75	1082

Table 6.17: Average training time on one NVIDIA Tesla K80 GPU in minutes.

Model	Quora → Quora	Quora →PAWS	Performance drop
BERT	.902	.440	51.22%
tBERT with GSDMM topics	**.905**	.442	51.16%
tBERT with LDA topics	**.905**	**.462**	**48.95%**

Table 6.18: Robustness analysis of proposed tBERT models. Quora →PAWS indicates that models were trained on Quora and evaluated on PAWS.

peak performance within the first three epochs. Although training for 4 or 7 epochs achieved marginal gains on SemEval A and C, longer finetuning of BERT could not exceed tBERT's best performance from the first three epochs (dotted line) on any dataset. We therefore conclude that longer finetuning does not considerably boost BERT's performance. Adding topics instead is more effective while avoiding the burden of greatly increased training time (compare Table 6.17).

Robustness analysis We previously highlighted the prevalence of lexical overlap bias in existing semantic similarity datasets (see Chapter 5): Example pairs with high lexical overlap tend to have a positive label, while low lexical overlap is associated with a negative label. In order to explore to what extent BERT and our best tBERT model (with LDA word topics) rely on lexical overlap, we conduct a robustness analysis by evaluating models which were trained on Quora on the adversarial PAWS dataset. PAWS was designed as an adversarial counterpart to Quora and created through the permutation and back-translation of examples from Quora, resulting in text pairs with particularly high lexical overlap. Therefore, models cannot rely on word overlap to identify paraphrases as is possible - to a certain extent - in Quora. Table 6.18 shows the results of the robustness experiment: the performance of BERT and tBERT models on PAWS drops noteably in comparison to Quora - as would be expected since the models have not encountered adversarial examples in training. However, both tBERT models perform better than BERT on PAWS, with the LDA-enriched tBERT model showing the smallest performance drop. This indicates that topics can contribute to reduced reliance on direct lexical overlap and increased model resilience to adversarial examples.

6.6 Conclusion

In this chapter we proposed two topic-informed approaches to Semantic Similarity Detection. The first part of the chapter introduced a novel topic-enriched CNN-LSTM architecture. Our model successfully combined topics with a neural architecture and improved over several neural baselines on multiple Semantic Similarity Detection datasets. We demonstrated that topics contributed consistently to our model's performance and that an early fusion of word embeddings with topic distributions is preferable over integration at a later stage. However, our results also showed that the proposed model was clearly outperformed by the more recent BERT model which has been fully pre-trained and been exposed to a sentence pair prediction task. This finding raised the questions if topics could be successfully combined with BERT and whether other topic models (e.g. short text topic models) might be beneficial.

To address these questions, the second part of this chapter proposed a flexible framework for combining topic models with BERT. Within this framework, we conducted experiments for combining BERT with LDA and GSDMM topics. We demonstrated that adding LDA topics to BERT consistently improved performance across a range of Semantic Similarity Detection datasets. Adding GSDMM topics achieved strong results on 3 of the datasets, but was less successful than the combination of LDA topics with BERT. Furthermore, our robustness analysis on an adversarial dataset indicated that incorporating topics as an additional source of information resulted in models which were more resilient to lexical overlap bias. In our qualitative analysis, we showed that improvements were mainly achieved on examples involving domain-specific words.

Future work may focus on whether combining topics with other pre-trained contextual models can lead to similar gains. Subsequent research could also address how to directly induce topic information directly into BERT without losing pre-trained information through catastrophic forgetting. Another research direction is to investigate if introducing more sophisticated topic models into the proposed framework can further improve results. One possibility is incorporating named entity promoting topic models (Krasnashchok and Jouili, 2018) which increase the importance of named entities in LDA, as named entities play a crucial role in domain-specific language. Another option is neural topic models (Ding et al., 2018), which could be trained jointly to obtain more task-specific topics. However, this requires more research since BERT is only finetuned for few epochs and with a small learning rate to prevent overfitting, while a neural topic model is likely to require more

training to learn meaningful topics.

CHAPTER 7

A Linguistically-Informed Approach to Semantic Similarity Detection

7.1 Introduction

Pre-trained contextual models Semantic Similarity Detection refers to a collection of sentence pair prediction tasks which aim at automatically recognising the presence of certain semantic relationships (see Chapter 3.1). Recently proposed pre-trained contextualised language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have achieved state-of-the-art performance across many areas of NLP including Semantic Similarity Detection (see Chapter 2.3.3). Consequently, there is an increased interest in exploring methods which build on and further improve these architectures. One line of work is model compression, which aims at developing smaller models that are more accessible while preserving most of BERT's performance (Xu et al., 2020; Goyal et al., 2020; Sanh et al., 2019; Aguilar et al., 2020; Lan et al., 2020; Chen et al., 2020; Sajjad et al., 2020). Another direction is to further enhance performance through increasing model parameters, e.g. by duplicating existing layers (Kao et al., 2020). A third line of work seeks performance gains by introducing complementary external information into BERT, such as information from knowledge bases (Peters et al., 2019; Wang et al., 2020a) or multi-modal information (Lu et al., 2019; Lin et al., 2020).

Linguistically-enriched embeddings Before the advent of contextualised models, transfer of pre-trained information between datasets and tasks in NLP was mostly based on word embeddings (refer to Chapter 2.3.2). Over multiple years, substantial effort was put into the creation and refinement of such embeddings.

While originally capturing mainly collocation patterns (Mikolov et al., 2013; Pennington et al., 2014), subsequent work enriched dense word representations with additional linguistic information, such as dependencies (Levy and Goldberg, 2014), subword information (Luong et al., 2013; Bojanowski et al., 2017), and semantic lexicons (Faruqui et al., 2015). As a result, there exists a wealth of pre-trained embedding resources for many languages in a unified format which could be useful for Semantic Similarity Detection.

Research questions In this chapter, we study the following two research questions (RQ3 and RQ4 from chapter 1): *First, how can one develop a generic Semantic Similarity Detection model which performs well across a variety of tasks and datasets? Second, can alternative information sources be exploited to make semantic similarity models more resilient to the identified biases?* This chapter addresses these questions by proposing an approach which combines a pre-trained BERT model with linguistically-enriched embeddings. Semantic and syntactic information plays a crucial role in Semantic Similarity Detection tasks and has been successfully exploited in earlier feature-engineered models (Filice et al., 2017). While recent large pre-trained language models such as BERT have achieved state-of-the-art performance in Semantic Similarity Detection, BERT is only trained to predict missing words - either behind masks or in the next sentence - and has no knowledge of syntax or semantics beyond what it picks up through this unsupervised pre-training. Therefore, we propose a novel method that explicitly injects additional linguistic knowledge in the form of pre-trained word embeddings in a pre-trained BERT model.

Contributions In this work, we propose a novel method which injects pre-trained embeddings into BERT's internal representation. Our work differs from previous approaches by introducing linguistically-enriched embeddings (as opposed to visual or knowledge base information) directly into BERT through a novel gated injection method which requires only 14% of the parameters in a standard multi-head attention mechanism. We apply our method to a range of Semantic Similarity Detection datasets and show that injecting pre-trained dependency-based and counter-fitted embeddings can further enhance BERT's performance. More specifically, we make the following contributions:

1. We propose GiBERT - a generic lightweight method for injecting externally pre-trained embeddings into BERT (section 7.4).

2. We provide ablation studies and detailed analysis for core model components (sections 7.4.4 and 7.7.2).

3. We demonstrate that our model improves over a vanilla BERT model and other baselines on semantic similarity detection datasets (section 7.6.1).

4. We show that injecting linguistically-enriched embeddings increases model resilience on an adversarial dataset (section 7.6.1).

5. Through detailed error analysis, we highlight examples where GiBERT provides improved performance, such as in cases of sentence pairs involving synonym pairs (section 7.7.1).

This chapter is based on our current NAACL submission, which is provided as Appendix D.

7.2 Related Work

BERT modifications Due to BERT's (Devlin et al. 2019, see Chapter 2.3.3.2) widespread success in NLP, many recent studies have focused on further improving BERT by introducing external information. Studies differ regarding the type of provided external information, the application area and their technical approach. We broadly categorise existing approaches based on their modification method into input-related, external and internal. *Input modifications* (Zhao et al., 2020; Singh et al., 2020; Lai et al., 2020; Ruan et al., 2020) adapt the information that is fed to BERT - e.g. feeding text triples separated by [SEP] tokens instead of sentence pairs as in Lai et al. (2020) - while leaving the architecture unchanged. *Output modifications* (Xuan et al., 2020; Zhang et al., 2020; Peinelt et al., 2020b) build on BERT's pre-trained representation by adding external information after the encoding step - e.g. combining it with additional semantic information as in Zhang et al. (2020) - without changing BERT itself.

In contrast, our approach belongs to the category of *internal modifications* which introduce new information directly into BERT by adapting its internal architecture. This approach is technically more difficult and might increase the risk of so-called catastrophic forgetting - completely forgetting previous knowledge when learning new tasks (French, 1999; Wen et al., 2018), but also offers the opportunity to directly harness BERT's powerful architecture to process the external information alongside the pre-trained one. Previous work on internal modifications has primarily attempted to combine BERT's internal representation with visual and knowledge

base information: Lu et al. (2019) modified BERT's transformer block with co-attention to integrate visual and textual information, while Lin et al. (2020) introduced a multimodal model which employs multi-head attention to integrate encoded image and text information between each transformer block. Peters et al. (2019) suggested a word-to-entity attention mechanism to incorporate external knowledge into BERT and Wang et al. (2020a) proposed to inject factual and linguistic knowledge through separate adapter modules. Our approach differs from previous research as we propose to introduce external information with a simple addition-based mechanism which requires fewer parameters than existing attention-based techniques (Lu et al., 2019; Lin et al., 2020; Peters et al., 2019). We further incorporate a gating mechanism to scale injected information in an attempt to reduce the risk of catastrophic forgetting. Moreover, our work focuses on injecting pre-trained word embeddings, rather than multimodal or knowledge base information as in previous studies.

Semantic Similarity Detection Detecting paraphrases and semantically related posts in Community Question Answering requires modelling the semantic relationship between a text pair. This is a fundamental and well known NLP problem for which many methods have been proposed. Early work has focused on feature-engineering techniques and successfully harnessed various syntactic (Filice et al., 2017), semantic (Charlet and Damnati, 2017) and lexical features (Tran et al., 2015; Almarwani and Diab, 2017). Subsequent work attempted to model text pair relationships solely based on increasingly complex neural architectures (Deriu and Cieliebak, 2017; Wang et al., 2017; Tan et al., 2018) or by combining both approaches through hybrid techniques (Wu et al., 2017a; Feng et al., 2017; Koreeda et al., 2017). Most recently, contextual models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have reached state-of-the-art performance through pre-training large context-aware language models on vast amounts of textual data. Our proposed approach joins up earlier lines of work with current state-of-the-art contextual representations by combining BERT with previously useful information in the form of dependency-based and counter-fitted word embeddings.

7.3 Datasets and Tasks

Datasets In this chapter, we perform experiments on six well-known semantic similarity datasets featuring different sizes (small vs large), tasks (answer selection vs Paraphrase Detection) and sentence lengths (short vs long) which were described in

detail in Chapter 4. Among these are three popular Paraphrase Detection datasets: the general purpose paraphrase dataset **MSRP** (introduced in section 4.2), the duplicate question dataset from the CQA platform **Quora** (discussed in section 4.3) and the adversarial counterpart to Quora **PAWS** (see section 4.4). We also include three domain-specific CQA datasets originating from the online forum *Qatar Living* which were featured in the SemEval 2017 shared task: the internal Answer Selection Dataset **SemEval A**, the question Paraphrase Detection dataset **SemEval B** and the external Answer Selection Dataset **SemEval C** (described in section 4.5).

Task All of the above datasets provide two short texts (usually a sentence long but in some cases consisting of multiple sentences). From here onward, we will use the term 'sentence' to refer to each short text. We frame the underlying task as predicting the semantic similarity between two sentences in a binary classification task (see Chapters 3.1 and 4). We use a binary classification setup (rather than ranking) as this is more generic and applies to all above datasets.

7.4 GiBERT Model

Feature-engineered Semantic Similarity Detection models have benefitted from semantic and syntactic information which is not explicitly provided to current state-of-the-art pre-trained architectures such as BERT. We therefore propose GiBERT - a **G**ated **I**njection **M**ethod for **BERT**. GiBERT directly injects linguistically-enriched word embeddings into BERT's internal representations, allowing the model to use this external knowledge to improve its performance further. Due to computational restrictions, our approach focuses on injecting information into a pre-trained BERT model during finetuning, rather than during the pre-training phase.[1]

7.4.1 Architecture

Overview Our proposed architecture is illustrated with a toy example in Figure 7.1. It comprises the following phases: obtaining BERT's intermediate representation from Transformer block i (step 1-2 in Figure 7.1), obtaining an alternative input representation based on linguistically-enriched word embeddings (step 3-4), combining both representations (steps 5-7) and passing on the injected information to the following BERT layers to make a final prediction (steps 8-9).

[1] Devlin et al. (2019) report that pre-training one BERT model took four days on 16 TPU chips.

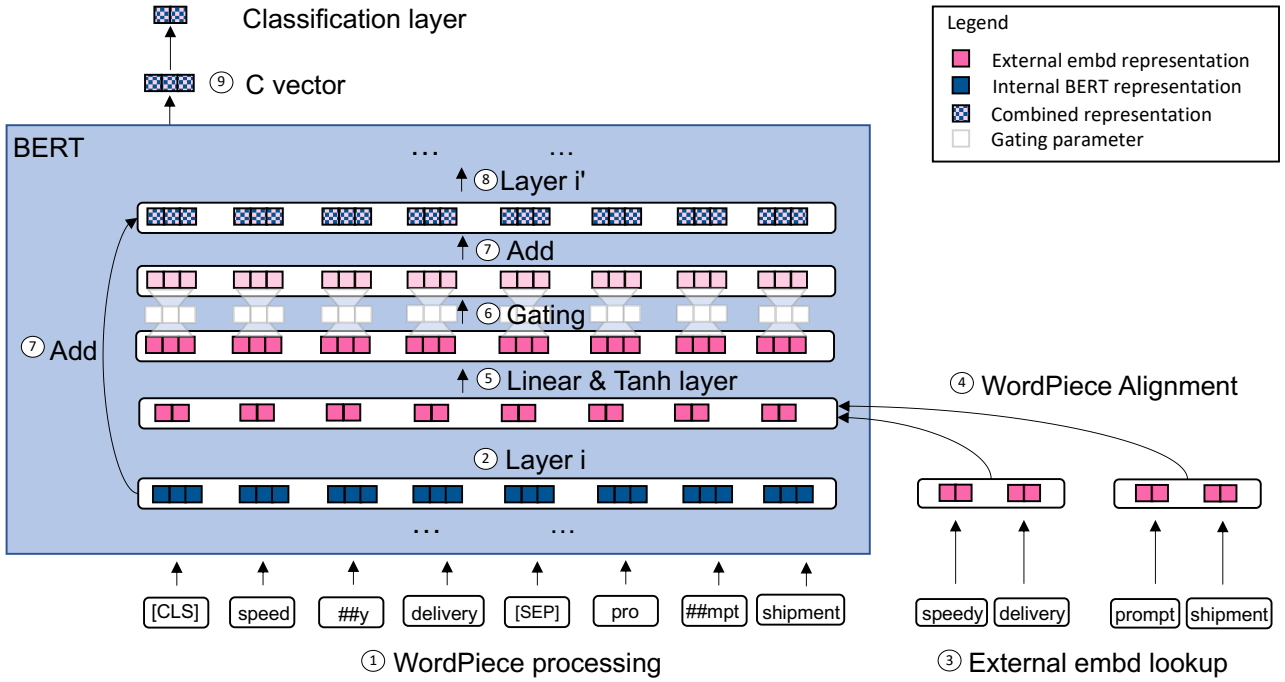


Figure 7.1: Our proposed GiBERT architecture illustrated with a toy example (where internal BERT dimension $D = 3$ and embedding dimension $E = 2$). The model input consists of a sentence pair which is processed with a word piece tokeniser (step 1) and encoded with BERT up to layer i (step 2). We obtain an alternative representation for the sentence pair based on pre-trained word embeddings (step 3), while ensuring that external word embeddings are aligned with BERT's word pieces by repeating embeddings for tokens which have been broken down into several word pieces (step 4). The aligned word embedding sequence is passed through a linear and tanh layer to match BERT's embedding dimension (step 5). We apply a gating mechanism (step 6) before adding the injected information to BERT's representation from layer i (step 7). The combined representation is passed on to the next layer (step 8). At the final layer, the C vector is used as the sentence pair representation, followed by a classification layer (step 9).

BERT representation We encode a sentence pair with a pre-trained BERT model (Devlin et al. 2019, see Chapter 2.3.3.2) and obtain BERT's internal representation at different layers (see section 7.4.4 for injection layer choices).[2] Following standard practice, we process the two input sentences S_1 and S_2 with a word piece tokenizer (Wu et al., 2017b) and combine them using '[CLS]' and '[SEP]' tokens which indicate sentence boundaries. Then, the word pieces are mapped to ids, resulting in a sequence of word piece ids $\mathbf{E^W} = [w_1, ..., w_N]$ where N indicates the number of word pieces in the sequence (step 1 in Figure 7.1). In the case of embedding layer injection, we use BERT's embedding layer output denoted with $\mathbf{H^0}$ which results from summming the word piece embeddings $\mathbf{E^W}$, positional embeddings $\mathbf{E^P}$ and segment embeddings $\mathbf{E^S}$ (step 2):

$$\begin{aligned} \mathbf{H^0} &= \text{LayerNorm}(\mathbf{E^W} + \mathbf{E^P} + \mathbf{E^S}) \\ \mathbf{E^W}, \mathbf{E^P}, \mathbf{E^S}, \mathbf{H^0} &\in \mathbb{R}^{N \times D} \end{aligned} \tag{7.1}$$

where D is the internal hidden size of BERT ($D = 768$ for $\text{BERT}_{\text{BASE}}$). For injecting information at later layers, we obtain BERT's internal representation $\mathbf{H^i} \in \mathbb{R}^{N \times D}$ after transformer block i with $1 \leq i \leq L$ (step 2):

$$\begin{aligned} \mathbf{M^i} &= \text{LayerNorm}(\mathbf{H^{i-1}} + \text{MultiHeadAttention}(\mathbf{H^{i-1}}, \mathbf{H^{i-1}}, \mathbf{H^{i-1}})) \\ \mathbf{H^i} &= \text{LayerNorm}(\mathbf{M^i} + \text{FeedForward}(\mathbf{M^i})) \end{aligned} \tag{7.2}$$

where L is the number of layers ($L = 12$ for $\text{BERT}_{\text{BASE}}$).

External embedding representation To enrich this representation, we obtain alternative representations for S_1 and S_2 by looking up word embeddings in a matrix of pre-trained embeddings $\mathbf{E} \in \mathbb{R}^{|V| \times E}$ where $|V|$ indicates the vocabulary size and E is the dimensionality of the pre-trained embeddings (step 3, refer to section 7.4.3 for details on our choice of pre-trained embeddings). To ensure alignment between BERT's representation at word piece level and the word embedding representation at token level, an alignment function copies embeddings of tokens that were separated into multiple word pieces and adds BERT's special '[CLS]' and '[SEP]' tokens, resulting in an injection sequence $\mathbf{I} \in \mathbb{R}^{P \times E}$ (step 4). For example, we copy the pre-trained embedding of the word 'prompt' to match the two corresponding word pieces 'pro' and '##mpt' (see Figure 7.1).

[2] We use the the uncased version of $\text{BERT}_{\text{BASE}}$ available through Tensorflow Hub.

Multihead Attention Injection Multihead attention was proposed by Vaswani et al. (2017):

$$\begin{aligned}\text{MultiHeadAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= [head_1;...;head_h]\mathbf{W^O} \\ \text{where } head_i &= \text{Attention}(\mathbf{QW_i^Q},\mathbf{KW_i^K},\mathbf{VW_i^V}).\end{aligned} \tag{7.3}$$

and is employed in Transformer networks in the form of self-attention (where queries, keys and values come from the previous layer) or encoder-decoder attention (queries come from the decoder; keys and values from the encoder). Previous work has successfully employed multi-head attention to combine BERT with external information (see section 7.2). For example, in their multimodel VilBERT model, Lu et al. (2019) combined textual and visual representations by passing the keys and values from each modality as input to the other modality's multi-head attention block. Similarly, Peters et al. (2019) used multi-head attention to combine projected BERT representations (as queries) with entity-span representations (as keys and values) in their knowledge-enrichment method for BERT. For our case of combining BERT with the injection sequence, it is therefore intuitive to try to use the following multi-head attention injection method:

$$\mathbf{H^{i'}} = \mathbf{H^i} + \text{MultiHeadAttention}(\mathbf{H^i},\mathbf{I},\mathbf{I}) \tag{7.4}$$

where queries are provided by BERT's internal representation, while keys and values come from the injected embeddings. The output of the attention mechanism is then added to the previous layer.

Gated Injection However, the above multi-head attention injection is rather complex and requires many parameters. We therefore propose an alternative way of combining external embeddings with BERT which requires fewer parameters (see section 7.4.2). First, we add a feed-forward layer - consisting of a linear layer with $\mathbf{W^P} \in \mathbb{R}^{D\times E}$ and $\mathbf{b^P} \in \mathbb{R}^D$ with a tanh activation function - to project the aligned embedding sequence to BERT's internal dimensions and squash the output into a space between -1 and 1 (step 5):

$$\mathbf{P} = \text{FeedForward}(\mathbf{I}) \in \mathbf{R}^{N\times D} \tag{7.5}$$

Then, we inject the projected external information into BERT's representation from Transformer block i (see section 7.4.4 for injection at different locations) and obtain

a new enriched representation $\mathbf{H^{i'}} \in \mathbb{R}^{N \times D}$ as follows:

$$\mathbf{H^{i'}} = \mathbf{H^{i}} + \mathbf{P} \tag{7.6}$$

However, as injection values can get rather large (between -1 and 1) in comparison to BERT's internal representation (based on our observation usually ranging around -0.1 to 0.1), a downside of directly injecting external information in this way is that BERT's pre-trained information can be easily overwritten by the injection, resulting in catastrophic forgetting. To address this potential pitfall, we further propose a gating mechanism which uses a gating vector $\mathbf{g} \in \mathbb{R}^{D}$ to scale the injected information before combining it with BERT's internal representation as follows:

$$\mathbf{H^{i'}} = \mathbf{H^{i}} + \mathbf{g} \odot \mathbf{P} \tag{7.7}$$

where $\odot$ denotes element-wise multiplication using broadcasting (step 6 & 7). The gating parameters are initialised with zeros and updated during training.[3] This has the benefit of starting finetuning from representations which are equivalent to vanilla BERT and gradually introducing the injected information during finetuning along certain dimensions. In case the external representations are not beneficial for the task, it is easy for the model to ignore them by keeping the gating parameters at zero.

Output layer The combined representation $\mathbf{H^{i'}}$ is then fed as input to BERT's next Transformer block $i+1$ (step 8). At the final Transformer block $\mathbf{L}$, we use the $\mathbf{c} \in \mathbb{R}^{D}$ vector which corresponds to the '[CLS]' token in the input and is typically used as the sentence pair representation (step 9). As proposed by Devlin et al. (2019), this is followed by a softmax classification layer (with weights $\mathbf{W^{L}} \in \mathbb{R}^{C \times D}$ and $\mathbf{b^{L}} \in \mathbb{R}^{C}$) to calculate class probablilities where C indicates the number of classes. During finetuning, we train the entire model for three epochs with early stopping and cross-entropy loss. Learning rates are tuned for each seed and dataset based on development set performance (reported in section 7.5.1).

Overall, our proposed method offers a lightweight, intuitive and generic way of injecting pre-trained word embeddings into a pre-trained BERT architecture. As information can be injected directly into BERT (between any of the pre-trained transformer blocks), the final layer preserves the same format as vanilla BERT,

[3] We continue by using the gated addition from equation 7.7, but present a comparison with the non-gated variant from equation 7.6 in section 7.4.4.

which offers the opportunity to directly switch out vanilla BERT for our model in existing BERT-based architectures.

7.4.2 Required Injection Parameters

This section compares the number of required parameters in the two alternative injection methods discussed in section 7.4: a multi-head attention injection mechanism which is based on previous methods for combining external knowledge with BERT and a novel lightweight gated injection mechanism.

Multihead attention injection In multi-head attention injection (equations 7.3 to 7.4), the keys are provided by BERT's representation from the injection layer $\mathbf{H^i}$ and the queries are the injected information $\mathbf{I}$. Multihead attention requires the following weight matrices $\mathbf{W}$ and biases $\mathbf{b}$ to transform queries, keys and values (indicated by Q, K and V) and transform the attention output (indicated by O):

$$\begin{aligned}\text{params(MultiHeadAttentionInjection)} =& \text{params}(\mathbf{W^K}, \mathbf{W^Q}, \mathbf{W^V}, \mathbf{b^K}, \mathbf{b^Q}, \mathbf{b^V}) \\ &+ \text{params}(\mathbf{W^O}, \mathbf{b^O}) \\ =& D(2D + 2E + 4D) \\ \mathbf{W^Q}, \mathbf{W^O} \in \mathbb{R}^{D\times D}, \mathbf{W^K}, \mathbf{W^V} &\in \mathbb{R}^{E\times D}, \mathbf{b^K}, \mathbf{b^Q}, \mathbf{b^V}, \mathbf{b^O} \in \mathbb{R}^D\end{aligned} \tag{7.8}$$

where D indicates BERT's hidden dimension and E indicates the dimensionality of the injected embeddings. When injecting embeddings with $D = 300$ (see section 7.4.3) into $\text{BERT}_{\text{BASE}}$ with $E = 768$, this amounts to $\approx 1.6M$ new parameters.

Gated injection The proposed gated injection method (equations 7.5 to 7.7) only introduces the weights and biases from the projection layer, as well as the gating vector:

$$\begin{aligned}\text{params(GatedInjection)} =& \text{params}(\mathbf{W^P}, \mathbf{b^P}, \mathbf{g}) \\ =& D(E + 2). \\ \mathbf{W^P} \in \mathbb{R}^{D\times E}, &\mathbf{b^P} \in \mathbb{D}, \mathbf{g} \in \mathbb{D}\end{aligned} \tag{7.9}$$

Therefore, injecting embeddings with $D = 300$ into $\text{BERT}_{\text{BASE}}$ requires $\approx 0.2M$ new parameters. Our proposed gated injection mechanism only requires 14% of the parameters used in a multihead attention injection mechanism. Using fewer parameters results in a smaller model which is especially beneficial for injecting

information during finetuning, where small learning rates and few epochs make it difficult to learn large amounts of new parameters.

7.4.3 Injected Embeddings

While other kinds of information could be injected, we focus on two types of linguistically-enriched embeddings: dependency-based (Levy and Goldberg, 2014) and counter-fitted embeddings (Mrkšić et al., 2016). Our choice is motivated by previous research which found syntactic features useful for Semantic Similarity Detection (Filice et al., 2017; Feng et al., 2017) and counter-fitted embeddings helpful in several other tasks (Alzantot et al., 2018; Jin et al., 2020). We discuss both types of pre-trained embeddings in more detail below.

Dependency-based embeddings The dependency-based embeddings proposed by Levy and Goldberg (2014) modify the popular SkipGram embedding algorithm (Mikolov et al. 2013, see section 2.3.2.2) by replacing linear bag-of-word contexts with syntactic contexts. In SkipGram, bag-of-word contexts are used to train embeddings by predicting the surrounding words of a given word within a certain context window size. However, especially for small window sizes this approach may miss important words due to long-range dependencies. Dependency-based contexts alleviate this problem because they are not limited by linear token distance, but based on syntactic dependencies which offer a more focused and inclusive context. Levy and Goldberg (2014) extract dependency contexts from parsed English Wikipedia sentences and report that the resulting embeddings show more functional similarity than vanilla Skipgram embeddings. As BERT has not been explicitly exposed to syntactic information during pre-training and previous studies have found that BERT's knowledge of syntax is only partial (Rogers et al., 2020), we reason that these embeddings could provide helpful complementary information.

Counter-fitted embeddings The counter-fitted embeddings released by Mrkšić et al. (2016) are obtained by finetuning existing word embeddings based on linguistic constraints. Mrkšić et al. (2016) integrate antonymy and synonymy relations into word embeddings based on an objective function which combines three principles: repelling antonyms, attracting synonymy and preserving the vector space. For training, they obtain synonymy and antonymy pairs from the Paraphrase Database and WordNet, demonstrating an increased performance on SimLex-999. We use their highest-scoring vectors which they obtained by applying their counter-fitting method to Paragram vectors from Wieting et al. (2015). We reason that the antonym and

the synonym relations contained in the word embeddings could be especially useful for Paraphrase Detection by explicitly capturing these semantic relations.

Settings Both dependency-based and counter-fitted embeddings have a dimensionality of $D = 300$. Although the injected embeddings could be updated during finetuning, we choose to keep them static so that the experimental results reflect the effect of injecting the pre-trained embeddings as originally provided. Out of vocabulary words are assigned a randomly initialised embedding.

7.4.4 Injection Settings

Gating Mechanism Catastrophic forgetting is a potential problem when introducing external information into a pre-trained model as the injected information could disturb or completely overwrite existing knowledge (Wang et al., 2020a). In our proposed model, a gating mechanism is used to scale injected embeddings before adding them to the pre-trained internal BERT representation (see section 7.4.1). To understand the importance of this mechanism, we contrast development set performance for injecting information after the embedding layer with gating - as defined in equation 7.7 - and without - as in equation 7.6 - (Table 7.1).

For dependency embedding injection without gating, performance only improves on 2 out of 5 datasets over the baseline and in some cases even drops under BERT's performance, while it outperforms the baseline on all datasets when using the gating mechanism. Counter-fitted embedding injection without gating improves on four out of five datasets, with further improvements when adding gating, out-

	MSRP	Quora	SemEval		
			A	B	C
BERT	.906	.906	.714	.754	.414
GiBERT with dependency embeddings					
- without gating mechanism	.906	.905	.732	.751	.424
- with gating mechanism	**.913**	**.908**	**.755**	**.778**	**.433**
GiBERT with counter-fitted embeddings					
- without gating mechanism	**.907**	.906	.733	.763	.435
- with gating mechanism	**.907**	**.908**	**.751**	**.767**	**.451**

Table 7.1: Performance (F1) of proposed iBERT models injecting pretrained embeddings after the embedding layer with vs. without gating mechanism on the development set.

	MSRP	Quora	SemEval		
			A	B	C
BERT	.906	.906	.714	.754	.414
GiBERT with dependency embeddings					
- injection after embedding layer	.913	.908	.755	**.778**	.433
- injection after Transformer block 6	.911	.908	.755	.776	.438
- injection after Transformer block 11	**.914**	**.910**	**.760**	.773	**.444**
GiBERT with counter-fitted embeddings					
- injection after embedding layer	.907	.908	.751	.767	.451
- injection after Transformer block 6	**.917**	**.909**	**.760**	**.771**	**.464**
- injection after Transformer block 11	.910	.907	.755	**.771**	.450

Table 7.2: Performance of proposed GiBERT model for embedding injection at different BERT locations on development set.

performing the vanilla BERT model across all datasets. In addition, gating makes model training more stable and reduces failed runs (where the model predicted only the majority class) from 30% to 0% on the particularly imbalanced SemEval C dataset. These results highlight the importance of the gating mechanism in our proposed architecture.

Injection Location In our proposed model, information can be injected between any of BERT's pre-trained transformer blocks. We reason that different locations may be more appropriate for certain kinds of embeddings as previous research has found that different types of information tend to be encoded and processed at specific BERT layers (Rogers et al., 2020). We experiment with injecting embeddings at three possible locations: after the embedding layer (using $\mathbf{H^0}$), after the middle layer (using $\mathbf{H^6}$ in $\text{BERT}_{\text{BASE}}$) and after the penultimate layer (using $\mathbf{H^{11}}$ in $\text{BERT}_{\text{BASE}}$). Table 7.2 shows that midlayer injection is ideal for counter-fitted embeddings, while late injection appears to work best for dependency embeddings (Table 7.2). This is in line with previous work which found that BERT tends to processes syntactic information at later layers than linear word-level information (Rogers et al., 2020). We consequently use these injection locations in our final model (Table 7.6).

7.5 Evaluation

7.5.1 Experimental Setup

Metrics Our main evaluation metric is F1 score as this is more meaningful for datasets with imbalanced label distributions (such as SemEval C, see label ratios

	MSRP	Quora	SemEval		
			A	B	C
batch size	32	32	16	32	16
BERT					
Learning rate (1st seed)	5e-5	2e-5	3e-5	2e-5	2e-5
Learning rate (2nd seed)	5e-5	2e-5	2e-5	2e-5	3e-5
AiBERT with dependency-based embeddings					
Learning rate (1st seed)	3e-5	3e-5	2e-5	3e-5	2e-5
Learning rate (2nd seed)	5e-5	2e-5	2e-5	5e-5	2e-5
AiBERT with counter-fitted embeddings					
Learning rate (1st seed)	5e-5	2e-5	2e-5	3e-5	2e-5
Learning rate (2nd seed)	5e-5	3e-5	5e-5	3e-5	2e-5
GiBERT with dependency-based embeddings					
Learning rate (1st seed)	2e-5	3e-5	2e-5	3e-5	2e-5
Learning rate (2nd seed)	3e-5	2e-5	2e-5	5e-5	3e-5
GiBERT with counter-fitted embeddings					
Learning rate (1st seed)	5e-5	2e-5	2e-5	5e-5	2e-5
Learning rate (2nd seed)	5e-5	3e-5	3e-5	5e-5	3e-5

Table 7.3: Tuned hyper-parameters for BERT-based models.

reported in Chapter 4.1) than accuracy. We also report performance on difficult cases using the non-obvious F1 score (refer to Chapter 5). This metric distinguishes non-obvious instances in a dataset from obvious ones based on lexical overlap and gold labels, calculating a separate F1 score for challenging cases. It therefore tends to be lower than the normal F1 score.

Tuning Dodge et al. (2020) recently showed that early stopping and random seeds can have considerable impact on the performance of finetuned BERT models. In this work, we finetune all models for three epochs with early stopping. Our reported scores average model performance across two different seeds for BERT-based models. Tuned hyperparameters for all BERT-based methods are shown in Table 7.3.

7.5.2 Baselines

Previous systems We compare the performance of our proposed models with several previously published models: For SemEval, we compare against the best participating SemEval 2017 systems based on F1 score. For MSRP, we show a neural matching architecture (Pang et al., 2016). For Quora, we compare against the Interactive Inference Network (Gong et al., 2018) using accuracy, as no F1 has been reported. We further provide Shen et al. (2018b)'s simple word embedding-

based model which employs GloVe embeddings.

BERT We further provide a vanilla BERT baseline which follows standard practice by encoding the sentence pair with BERT's **c** vector from the final layer (corresponding to the '[CLS]' token in the input) which is followed by a softmax layer. We follow the same finetuning procedure as our main model by finetuning all layers for three epochs with early stopping and tuning learning rates on the development set of each dataset as recommended by Devlin et al. (2019).

tBERT Moreover, we attempt to combine embeddings with BERT using the averaging and concatenation method from the tBERT model (see Chapter 6). Instead of the word topics in the original system, we use pretrained counter-fitted and dependency embeddings for a direct comparison with our methods.

AiBERT We provide an alternative **A**ttention-based embedding **I**njection method for **BERT** (AiBERT) based on the multihead attention injection mechanism described in equations 7.3 to 7.4. We follow the same finetuning procedure as our primary model by finetuning all layers for three epochs with early stopping and tuning learning rates as well as injection location (after transformer block six for both dependency and counter-fitted embeddings, see Table 7.4) on the development set.

	MSRP	Quora	SemEval		
			A	B	C
AiBERT with dependency embeddings					
- injection after embedding layer	.9028	.9058	.7306	**.7512**	.4214
- injection after Transformer block 6	**.9040**	**.9073**	**.7342**	.7493	.4177
- injection after Transformer block 11	.8993	.9070	.7265	.7506	**.4296**
AiBERT with counter-fitted embeddings					
- injection after embedding layer	.9008	**.9078**	.7181	.7472	.4186
- injection after Transformer block 6	**.9066**	.9070	**.7269**	.7500	**.4267**
- injection after Transformer block 11	.9010	.9068	.7171	**.7505**	.4206

Table 7.4: Performance of AiBERT model for embedding injection at different BERT locations on development set.

7.6 Experimental Results

7.6.1 Full Model

Overall results Table 7.6 shows the performance of our linguistically enriched models in comparison with previous work. Our results demonstrate that GiBERT with counter-fitted embeddings outperforms all other systems in both average F1 and average non-obvious F1 score. This shows that the model improves on challenging dataset instances, rather than merely leveraging shallow surface patterns. It is worth noting that GiBERT has the fewest parameters of all BERT-enhancing models (see Table 7.5) and does not require any additional preprocessing tools (such as the neural SRL tagger required by SemBERT), making it more efficient than SemBERT, tBERT and AiBERT. The largest improvement of GiBERT over BERT is observed with counter-fitted embeddings, especially on the internal CQA datasets SemEval A and B (the datasets with the highest proportion of examples involving synonym pairs, see section 7.7.1). GiBERT with dependency embeddings still improves over vanilla BERT, but performance gains tend to be smaller and roughly similar to the more complex AiBERT injection method. We reason that the injection of dependency embeddings may be less effective because semantic information is more important for the tasks at hand or because syntactic information needs to be introduced more directly.

Injection method This section contrasts our gated injection method (GiBERT) with the alternative attention injection method (AiBERT). We find that both injection methods generally improve over the performance of the vanilla BERT baseline. In a direct comparison between the two injection methods, we find that injecting embeddings with our lightweight gated method achieves comparable results to the complex multihead attention injection method for introducing dependency embeddings, while for the injection of counter-fitted embeddings, GiBERT even outperforms AiBERT.

	Total parameters
BERT	110.1M
GiBERT	110.3M
tBERT	111.0M
AiBERT	111.7M
SemBERT	111.9M

Table 7.5: BERT-based models ordered by size (using $\text{BERT}_{\text{BASE}}$).

	F1						non-obvious F1					
	MSRP	Quora	SemEval			avg	MSRP	Quora	SemEval			avg
			A	B	C				A	B	C	
Previous systems												
KeLP⋄	-	-	-	.506	-	-	-	-	-	.199	-	-
ECNU⋄	-	-	.777	-	-	-	-	-	.707	-	-	-
Bunji⋄	-	-	-	-	.197	-	-	-	-	-	.028	-
SWEM (Shen et al., 2018b)	.813	.830	-	-	-	-	-	-	-	-	-	-
DIIN (Gong et al., 2018)	-	(.891)	-	-	-	-	-	-	-	-	-	-
MatchPyramid (Pang et al., 2016)	.829	-	-	-	-	-	-	-	-	-	-	-
BERT⋆	.876	.902	.704	.473	.268	.645	.827	.860	.656	.243	.085	.534
SemBERT⋆	.876	.901	✗	✗	✗	-	.820	**.874**	✗	✗	✗	-
$\text{tBERT}_{\text{dependency}}$⋆	.882	.906	.780	.510	.242	.664	.827	.858	**.728**	.262	.090	.553
$\text{tBERT}_{\text{counter-fitted}}$⋆	.879	.906	.756	.500	.215	.651	.824	.857	.699	.258	.064	.540
Proposed systems												
$\text{AiBERT}_{\text{dependency}}$	.863	.903	.738	.498	**.282**	.657	.792	.866	.681	**.268**	**.090**	.539
$\text{AiBERT}_{\text{counter-fitted}}$	.877	.904	.724	.496	.263	.653	.835	.867	.662	.264	.076	.541
$\text{GiBERT}_{\text{dependency}}$	.883	.904	.768	.474	.238	.653	.849	.864	.704	.231	.087	.547
$\text{GiBERT}_{\text{counter-fitted}}$	**.884**	**.907**	**.780**	**.511**	.256	**.668**	**.858**	.862	.719	.248	**.090**	**.555**

Table 7.6: Model performance (F1) on the test set. All BERT-based methods use $\text{BERT}_{\text{BASE}}$. avg = average performance across all datasets, ⋄ = results from publication, ⋆ = official code run on our data, ✗= run failed due to insufficient GPU memory, $_{\text{dependency}}$ = dependency embeddings, $_{\text{counter-fitted}}$ = counter-fitted embeddings.

Robustness Analysis We previously showed that lexical overlap in existing semantic similarity datasets provides direct clues for class membership (refer to Chapter 5): Example pairs with high lexical overlap tend to have a positive label, while low lexical overlap is associated with a negative label. In order to explore to what extent BERT and GiBERT are prone to lexical overlap bias, we conduct a robustness analysis in which we train models on Quora and evaluate them on the adversarial PAWS dataset. PAWS was designed as an adversarial counterpart to Quora. The dataset was created through the permutation and back-translation of examples from Quora, resulting in text pairs with particularly high lexical overlap. Therefore, models cannot rely on word overlap to identify paraphrases (in contrast to Quora). For comparison purposes, we also provide the models' performance on Quora. Table 7.7 shows that performance of all models drops notably on PAWS in comparison to Quora. This behaviour is expected because the models have not encountered any adversarial examples in training. However, both GiBERT models perform better than BERT in the adversarial setup, which indicates that injecting pre-trained embedding into BERT increases model resilience slightly. The dependency injected GiBERT model shows the smallest performance drop as dependency embeddings appear to be more helpful in this syntactically adversarial dataset.

7.7 Qualitative Analysis

7.7.1 Error Analysis

Counter-fitted embeddings are designed to explicitly encode synonym and antonym relationships between words. To better understand how the injection of counter-fitted embeddings affects the ability of our model to deal with instances involving such semantic relations, we use synonym and antonym pairs from the PPDB and wordnet (provided by Mrkšić et al. 2016) and search the development partition of the datasets for sentence pairs where the first sentence contains one word of the synonym/antonym pair and the second sentence the other word. Table 7.8 reports F1

Model	Quora→Quora	Quora→PAWS	Performance drop
BERT	.902	.440	51.22%
$\text{GiBERT}_{\text{dependency}}$	.904	**.450**	**50.22%**
$\text{GiBERT}_{\text{counter-fitted}}$	**.907**	.446	50.83%

Table 7.7: Robustness analysis of proposed GiBERT models. Quora →PAWS indicates that models were trained on Quora and evaluated on PAWS.

	MSRP	Quora	SemEval A	SemEval B	SemEval C
Instances with **antonym** pairs	(4%)	(4%)	(21%)	(28%)	(20%)
- BERT	**.81**	**.87**	**.77**	**.75**	**.46**
- GiBERT with counter-fitted embd	**.81**	.86	**.77**	**.75**	**.46**
Instances with **synonym** pairs	(11%)	(9%)	(22%)	(31%)	(17%)
- BERT	.87	.90	.81	.78	**.54**
- GiBERT with counter-fitted embd	**.90**	**.91**	**.82**	**.83**	**.54**
Instances **without** synonym/antonym pairs	(85%)	(87%)	(64%)	(51%)	(68%)
- BERT	.91	**.91**	.71	.72	.36
- GiBERT with counter-fitted embd	**.92**	**.91**	**.73**	**.73**	**.41**

Table 7.8: Performance (F1) of proposed GiBERT model on instances containing synonymy pairs, antonymy pairs or no pairs across datasets. Note that one instance can contain synonym and antonym pairs at the same time, therefore the percentages of the three categories do not necessarily add up to 100.

	Sentence 1	Sentence 2	Gold label	BERT prediction	GiBERT prediction
(1)	it took me more than 10 people; over the course of the whole day to convince my point at qatar **airways** ... as to how my points needs to be redeemed ... at long last my point was made ... dont seem know what they are doing ??? appalling to say the least	this isn't the first time. so many rants by irate customers on so many diverse situations signals a very serious problem. so called first class **airlines** and no basic customer care. over confidence much?	is related	not related	is related
(2)	hi; my **wife** was on a visit visa; today; her residency visa was issued; so i went to immigration and paid 500 so there is no need to leave the country and enter again on the residency visa. she has done her medical before for the visit visa extension; do we need to do the medical again for the residency visa? thanks	dear all ; please let me know how many days taking for approve family visa nw; am last wednesday (12/09/2012) apply family visa for my **husband** and daughter; but still now showing in moi website itz under review; itz usual reply? why delayed like this? please help me regards divya	is related	is related	not related

Table 7.9: Examples from the SemEval development set. Synonym and antonym pairs are highlighted.

performance of our model on cases with synonym pairs, antonym pairs and neither one. We find that our model's F1 performance particularly improves over BERT on instances containing synonym pairs, as illustrated in example (1) in Table 7.9. In contrast, the performance on cases with antonym pairs stays roughly the same but slightly decreases on Quora. This can be understood with the help of example (2) in Table 7.9, as word pairs can be antonyms in isolation (e.g. husband - wife), but not in the specific context of a given example (e.g. here it is not important if the wife or the husband applied for the visa). In such cases, the injection of distant antonym pair embeddings could deter the model from detecting related sentence pairs. We also observe a slight performance boost for cases that do not contain synonym or antonym pairs. This could be because of improved representations for individual words which occurred in examples without their synonym or antonym counterpart and are therefore not captured under the antonym or synonym categories in Table 7.8.

7.7.2 Gating Parameter Analysis

As described in section 7.4, the gating parameters $\mathbf{g}$ in our proposed model are initialised as a vector of zeros. During training, the model can learn to gradually inject external information by adjusting gating parameters to > 0 for adding, or < 0 for subtracting injected information along certain dimensions. Alternatively, injection stays turned off if all parameters remain at zero. Figure 7.2 shows a histogram of learned gating vectors for our best GiBERT models with counter-fitted (top) and dependency embedding injection (bottom). On most datasets, the majority of parameters have been updated to small non-zero values, letting through controlled amounts of injected information without completely overwriting BERT's internal representation. Only on SemEval B (with 4K instances the smallest of the datasets, see section 5.2), more than 500 of the 768 dimensions of the injected information stay blocked out for both model variants. The gating parameters also filter out many dimensions of the dependency-based embeddings on MSRP (the second smallest dataset). This suggests that models trained on smaller datasets may benefit from slightly longer finetuning or a different gating parameter initialisation to make full use of the injected information.[4]

[4]Note that we train models for the same number of epochs, but one epoch uses all training examples contained in the dataset. This gives models trained on larger datasets more opportunity to update their parameters.

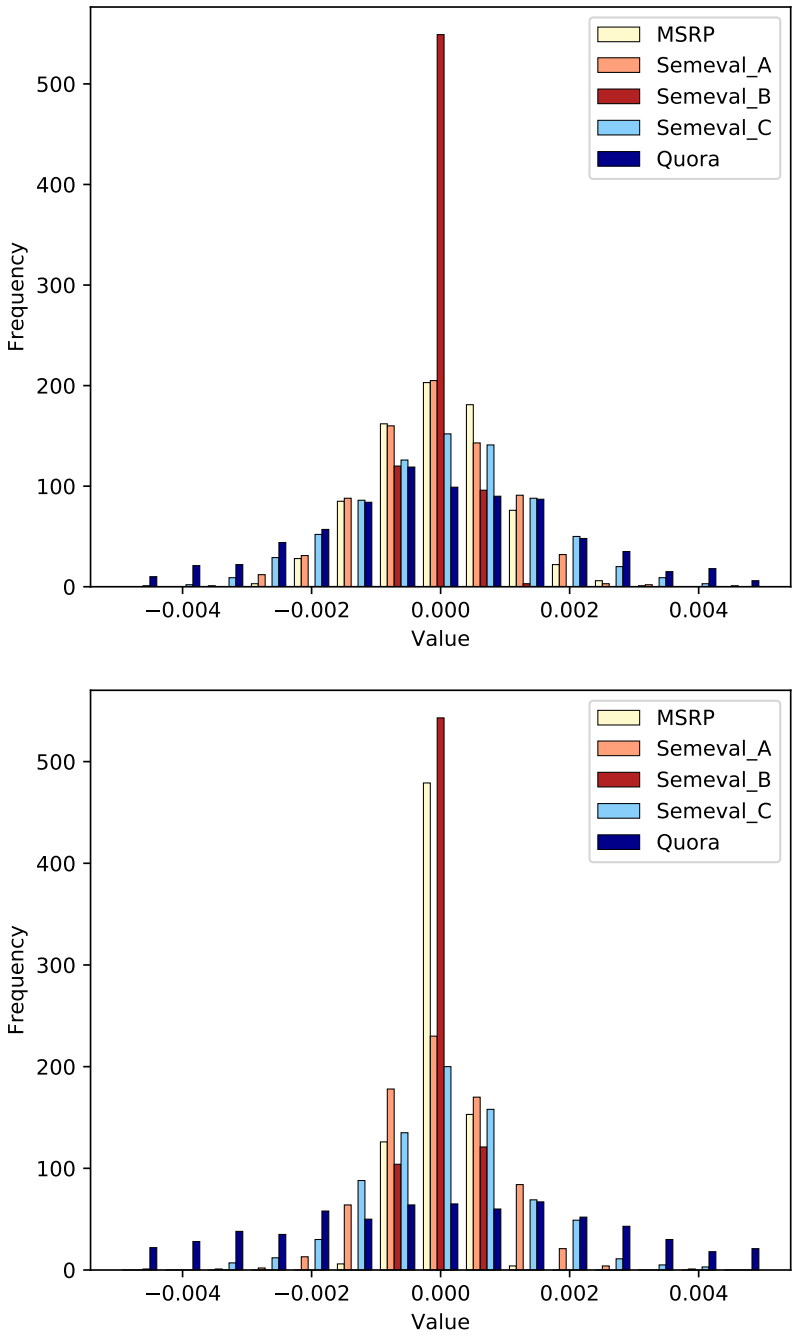


Figure 7.2: Histogram of the 768-dimensional gating vector $\mathbf{g}$ across datasets for GiBERT with counter-fitted embeddings (top) and GiBERT with dependency embeddings (bottom).

7.8 Future Work: Topic Injection

This chapter focused on the injection of linguistically-enriched embeddings into BERT. However, the success of the proposed embedding injection method in combination with our earlier finding that topics can improve BERT's performance for Semantic Similarity Detection (refer to Chapter 6) leads to the question whether gated injection could be used for topics. In our earlier topic-informed neural model TAPA (see Chapter 6.4), an early introduction of topics was critical for good results, but topics are introduced relatively late in our successful tBERT model (see Chapter 6.5). This suggested that an earlier introduction of topics into BERT could be promising. We therefore experimented with injecting LDA topics in the embedding layer (Table 7.10). While this topic injection improved over the BERT baseline on three of the five datasets, it did not outperform the simpler tBERT model introduced in Chapter 6. Therefore, more experiments may be necessary to determine the best injection layer (trying injection in a different early layer, e.g. 1-5). Alternatively, the injection mechanism may need to be adjusted for topics. We leave this question to be explored in future research.

Model	Type	MSRP	Quora	SemEval		
				A	B	C
LDA topic baseline	feature-based	.799	.736	.684	.436	.096
BERT (Devlin et al., 2019)	neural	.876	.902	.704	.473	.268
tBERT with LDA topics	neural	**.884**	**.905**	.768	**.524**	**.273**
GiBERT with LDA topics	neural	.881	.904	**.780**	.473	.254

Table 7.10: Test set performance (F1) of proposed GiBERT model with topic injection in the embedding layer. Bold font highlights the best system.

7.9 Conclusion

In this chapter, we introduced a new approach to Semantic Similarity Detection which introduced external word embeddings into a pre-trained BERT model. Our proposed architecture combines linguistically-enriched embeddings with the representation at an intermediate BERT layer based on a novel gated injection mechanism which requires significantly fewer parameters than a multi-head attention injection method. Evaluating our gated injection method on a range of Semantic Similarity Detection datasets, we demonstrated that injecting counter-fitted embeddings clearly improved performance over vanilla BERT, while dependency embeddings

achieved slightly smaller gains for these tasks. In comparison to the multi-head attention injection mechanism, we found our gated method at least equally effective with a comparable performance for dependency embedding and improved results for counter-fitted embeddings. In ablations studies, we showed that the choice of injection location and the use of the proposed gating mechanism are crucial for our architecture. Our qualitative analysis highlighted that counter-fitted injection was particularly helpful for instances involving synonym pairs. Moreover, the evaluation on an adversarial dataset showed that injecting linguistically-informed embeddings increased model robustness.

This work has evaluated our proposed architecture for Semantic Similarity Detection tasks. However, the model is very generic and could be applied to other tasks in the future. Future work could also explore combining multiple embedding sources or injecting other types of information (e.g. knowledge base embeddings). Another direction is to investigate the usefulness of embedding injection for other tasks or distilled BERT models. Our work focused on injecting linguistic information during finetuning, but future work could focus on introducing linguistic knowledge during pre-training.

Part IV

Conclusion

CHAPTER 8

Conclusions

8.1 Main Findings

This chapter summarises and discusses the main findings of this thesis with respect to the research questions posed in Chapter 1. Following the structure of the thesis, we start by discussing questions related to evaluation methods for Semantic Similarity Detection and then proceed to questions concerned with modelling techniques for Semantic Similarity Detection.

8.1.1 Evaluation for Semantic Similarity Detection

RQ1 Which biases exist in current semantic similarity datasets?

- **OBJ 1.1** Select a range of well-known and commonly used Semantic Similarity Detection datasets.
- **OBJ 1.2** Analyse semantic similarity dataset properties and identify bias patterns.
- **OBJ 1.3** Demonstrate how models can exploit the identified biases.
- **OBJ 1.4** Quantify the prevalence of identified biases across well-known semantic similarity datasets.

Chapter 5 of this thesis investigated the role of lexical overlap across a range of semantic similarity datasets. We demonstrated that sentence pairs with high lexical overlap tend to be associated with a positive label, while sentence pairs with low lexical overlap are likely to have a negative label (Section 5.4). Therefore, lexical overlap can provide direct clues for class membership. We highlighted this lexical overlap bias by visualising the relationship between lexical divergence distribution patterns and gold labels and quantified the prevalence of lexical overlap bias based

on a simple rule-based baseline, which assigned labels solely based on lexical overlap (Section 5.5). We found that lexical overlap bias was present to varying degrees across the examined datasets: The highest levels were observed in two very popular and widely used benchmark datasets (Quora and MSRP), where the simple lexical overlap baseline as able to reach high performance (0.7 F1). In contrast, several other datasets (STS, SemEval A and B) showed weaker links between lexical overlap and gold labels (0.4-0.6 F1), while the lexical overlap baseline could not predict semantic similarity (< 0.1 F1) in SemEval C and the automatically created adversarial PAWS dataset. We argued that the presence of high degrees of lexical overlap bias is problematic both for training and evaluating Semantic Similarity Detection models. Using biased datasets during training risks producing models which focus only on shallow patterns and are not reliable for real-world applications. Using biased datasets during evaluation risks producing inflated scores and overestimating true model capabilities. However, these problems can be addressed by taking lexical overlap bias into account during dataset construction and model evaluation.

RQ2 What are ways to account for identified dataset biases during model evaluation?

- **OBJ 2.1** Understand currently used evaluation metrics and their limitations.
- **OBJ 2.2** Demonstrate how identified dataset biases impact evaluation.
- **OBJ 2.3** Propose novel evaluation techniques which account for identified dataset biases.
- **OBJ 2.4** Demonstrate the effectiveness of proposed evaluation techniques.

Current evaluation metrics give equal weight to all examples in a dataset, regardless of their difficulty or existing biases. One approach to deal with lexical overlap bias is by simply measuring bias prevalence for a selection of available datasets and only working with bias-free datasets. However, this greatly limits the choice of available datasets and in practise it is not always feasible to obtain unbiased datasets. A compromise could be achieved in measuring the robustness of models which were trained on biased datasets against lexical overlap bias by evaluating them on an adversarial bias-free counterpart, which can be automatically generated, e.g. based on the word permutation and back-translation method as proposed by Zhang et al. (2019c). However, this method only focuses on instances with high word overlap and requires human verification to ensure that the automatically produced dataset instances are correct. Therefore, our work further investigated metrics which

account for lexical overlap during model evaluation. We presented a criterion for automatically distinguishing between easy and difficult items in text pair similarity prediction tasks based on lexical overlap and gold labels (Section 5.6). Based on this distinction, we found that more than 50% of cases in all human-created datasets under investigation are relatively obvious. We further proposed evaluation metrics based on this criterion and demonstrated that recently proposed models perform significantly worse on non-obvious examples compared to obvious cases (Section 5.7). In order to capture model performance on difficult items and encourage the development of models that perform well on such cases, we proposed the non-obvious F1 score as a novel complementary ranking metric for model evaluation.

8.1.2 Models for Semantic Similarity Detection

RQ3 Can alternative information sources be exploited to make Semantic Similarity Detection models more resilient to the identified biases?

- **OBJ 3.1** Attempt to combine topics with neural models as an additional semantic similarity signal.
- **OBJ 3.2** Attempt to incorporate additional linguistic information into pre-trained models.
- **OBJ 3.3** Propose appropriate mechanisms for combining external information with existing models.
- **OBJ 3.4** Evaluate if the incorporation of additional information into pre-trained models improves model performance and robustness.
- **OBJ 3.5** Understand the properties of cases in which proposed models improve.

Based on our findings regarding the prevalence of lexical overlap in existing datasets and alternative evaluation methods, the second part of this thesis aimed at using these methods to develop more resilient models for Semantic Similarity Detection. To develop models that are less reliant on lexical overlap, we investigated two additional information sources for their potential to serve as complementary indicators of semantic similarity. Our first approach in Chapter 6 investigated the incorporation of topic models with successful neural models. Based on experiments with an earlier topic-enhanced neural model which indicated the usefulness of topics for Semantic Similarity Detection (Section 6.4), we developed a simple framework dubbed tBERT which successfully combined LDA topics with BERT (Section 6.5).

We demonstrated that this model increased performance on non-obvious cases of semantic similarity and performed better on the adversarial PAWS dataset compared to its topic-unaware counterpart. Through manual error analysis, we found that topics particularly improved model predictions for cases involving domain-specific words which are frequently encountered in specialised online question answering communities and are unlikely to have occurred during pre-training. Our second approach in Chapter 7 explored combining pre-trained contextual language models with pre-trained word embeddings for Semantic Similarity Detection. The results of our GiBERT model showed that injecting linguistically-enriched embeddings into BERT improved model performance and increased robustness against lexical overlap bias (Section 7.6). Our study indicated that injecting counter-fitted embeddings was particularly effective and increased the model's ability to deal with synonyms (Section 7.7). For each information source, we investigated multiple mechanisms for combining the additional information with neural models (Sections 6.4.1, 6.5.1 and 7.4.1). Based on our findings, there was no single best mechanism. Rather, successful methods for combining additional information with existing models were dependent on the type of information provided: While topics were best combined with BERT through a simple external model modification, the integration of embeddings with BERT benefited more from a lightweight gated injection mechanism.

RQ4 How can one develop a generic Semantic Similarity Detection model which performs well across a variety of tasks and datasets?

- **OBJ 4.1** Identify well-known datasets covering a wide range of Semantic Similarity Detection tasks.
- **OBJ 4.2** Investigate the existing literature in the area of Semantic Similarity Detection to understand common approaches and promising directions.
- **OBJ 4.3** Develop robust and accurate neural models for Semantic Similarity Detection which do not require feature engineering or dataset-specific information.
- **OBJ 4.4** Evaluate developed models against dataset-specific feature-engineered systems and other neural baselines.

Many techniques have been proposed for semantic similarity tasks, including statistical, feature-engineered and neural approaches. While earlier work has focused primarily on feature-engineered models, this thesis contributes to research on more recent neural methods. Neural methods provide the benefit of not requiring manual

feature-engineering which is time-consuming and does not generalise well. In order to develop generic and transferable architectures which work well across a variety of datasets and related tasks, we focused on the development of three distinct neural models for Semantic Similarity Detection which do not require any task- or dataset-specific information. Although our initial attempts were not able to compete with feature-engineered models on the smaller datasets due to the scarcity of training data (Section 6.4.4), our subsequent models built on pre-trained models such as BERT to overcome this problem (Section 6.5.7). Our proposed architectures enhanced BERT with external information sources which were chosen in a way that is relevant and applicable across multiple datasets and tasks (Chapters 6.5 and 7). We leveraged externally pre-trained embeddings which are generally available and topics which can be easily obtained by training a topic model for each respective dataset. In our evaluation, we demonstrated that our proposed models performed well on a range of datasets and outperformed dataset-specific feature-engineered models, as well as generic neural baselines.

8.2 Directions for Future Work

There are multiple directions in which the work presented in this thesis can be further extended. This section discusses applications of the presented research and highlights the main opportunities for future contributions to the area of evaluation methods and modelling techniques for Semantic Similarity Detection.

8.2.1 Evaluation for Semantic Similarity Detection

Dataset construction In this thesis, we demonstrated that the majority of examined Semantic Similarity Detection sets contained noticeable levels of lexical overlap bias, while only one human-generated dataset was bias-free. This highlights the need for taking lexical overlap bias into account during dataset construction and creating more challenging datasets which cannot be solved by merely exploiting superficial surface structure similarity. The presented methods in this thesis can be employed to visualise and quantify lexical overlap bias as a quality control measure for new semantic similarity datasets. For example, the simple lexical overlap baseline can be used to gauge to what extent surface similarity can be exploited to solve a new dataset. If lexical overlap bias is detected, debiasing procedures should be applied before releasing the dataset. Automatic methods for removing lexical overlap bias could employ overlap-aware sampling strategies and are a promising direction for future work.

Evaluating non-obvious cases of semantic similarity We presented a simple novel metric (non-obvious F1) for evaluating non-obvious cases of semantic similarity. Future work could further refine this metric, for example by harnessing word order and part-of-speech information. The current version of our metric only captures word overlap between individual tokens and is therefore not sensitive to word order permutations. This could be improved by measuring lexical similarity based on n-gram counts. Moreover, our metric currently does not distinguish between content and function words, although overlap of the former is more important than of the latter when it comes to predicting semantic similarity. This can be addressed by filtering out or giving lower weight to certain words depending on their POS or a stop word list. However, these decisions require careful consideration as certain function words (e.g. phrases involved in negation) can be crucial for detecting semantic similarity between sentences.

Identifying and addressing other biases This work only focused on highlighting the presence of lexical overlap bias. However, it has been demonstrated that there exist other kinds of sentence pair prediction biases such as sentence length bias (Gururangan et al., 2018) and selection bias (Zhang et al., 2019a). More work is needed to develop resilient evaluation methods and appropriate procedures for debiasing one or ideally multiple bias patterns at a time (Mahabadi et al., 2020). Furthermore, semantic similarity datasets are likely to contain other kinds of biases which remain yet to be explored.

8.2.2 Models for Semantic Similarity Detection

More recent pre-trained models While our presented BERT-based models have been successful for Semantic Similarity Detection, it is likely that they can be further improved by incorporating more recently released pre-trained models. Many current pre-trained models are based on the Transformer architecture, which makes it easy to replace BERT with them. The performance of our models is likely to be further increased through improved BERT-like models such as RoBERTa (Liu et al., 2019). Moreover, recent research in model compression techniques for reducing the size of pre-trained models can be applied to our models by replacing BERT with distilled or pruned BERT alternatives, such as DistilBERT (Sanh et al., 2019) or through layer dropping strategies (Sajjad et al., 2020).

Topic-enriched Semantic Similarity Detection Our proposed topic-enriched BERT-based architecture performed well with simple LDA topics. Although the

model was not improved with short-text topic models in our experiments, a promising research direction is to investigate if introducing more sophisticated topic models into the framework can further boost results. One possibility is the incorporation of named-entity promoting topic models (Krasnashchok and Jouili, 2018) which increase the importance of named entities in LDA, as named entities play a crucial role in domain-specific language. Another option are neural topic models (Ding et al., 2018), which could be trained jointly with the Semantic Similarity Detection model to obtain more task-specific topics. However, this still requires more research as BERT is only finetuned for few epochs and with a small learning rate to prevent overfitting, while a neural topic model is likely to require more training to learn meaningful topics.

Linguistically-enriched Semantic Similarity Detection Our linguistically-enriched BERT-based model benefited from the injection of dependency-based embeddings, although they were generally less effective than the addition of counter-fitted embeddings. We reason that leveraging dependencies for the training of word embeddings is a rather indirect way to capture syntactic knowledge and see great potential in more direct ways of combining syntactic information with pre-trained models, for example by directly encoding dependency relations between words as in Zhang et al. (2019d).

Evaluation for automatic text generation With slight modifications, the presented Semantic Similarity Detection models in this thesis could also be applied for evaluating automatically generated text. Commonly used methods for automatic evaluation of generated text are primarily based on comparing the generated text with a reference text based on lexical overlap measures. However, recently proposed methods such as BERTScore (Zhang et al., 2019b) and MoverScore (Zhao et al., 2019) have successfully incorporated contextual representations. However, both methods are based on BERT, which is not pre-trained or finetuned on paraphrase detection tasks. Our proposed GiBERT model may be particularly useful for such purposes as it is finetuned on semantic similarity tasks and showed an improved ability to identify paraphrases.

Bibliography

Asma Ben Abacha and Dina Demner-Fushman. NLM NIH at SemEval-2017 Task 3: From Question Entailment to Question Similarity for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 349–352, Vancouver, Canada, 2017. Association for Computational Linguistics.

Rod Adams, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. Textual entailment through extended lexical overlap and lexico-semantic matching. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing - RTE '07*, page 119, Prague, Czech Republic, 2007. Association for Computational Linguistics. doi: 10.3115/1654536.1654560.

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. Neural Models for Detecting Binary Semantic Textual Similarity for Algerian and MSA. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 78–87, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4609.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012*, pages 385–393, Montreal, Canada, 2012. Association for Computer Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *SEM 2013 shared task: Semantic Textual Similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013*, pages 32–43, Atlanta, USA, 2013. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015), pages 252–263, Denver, Colorado, 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2045.

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge Distillation from Internal Representations. *arXiv:1910.03723 [cs]*, January 2020.

Surya Agustian and Hiroya Takamura. UINSUSKA-TiTech at SemEval-2017 Task 3: Exploiting Word Importance Levels for Similarity Features for CQA. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 370–374, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2061.

Nada Almarwani and Mona Diab. GW_QA at SemEval-2017 Task 3- Question Answer Re-ranking on Arabic Fora. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 344–348, Vancouver, Canada, 2017. Association for Computational Linguistics.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating Natural Language Adversarial Examples. *arXiv:1804.07998 [cs]*, September 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivý. Sentence Pair Scoring: Towards Unified Framework for Text Comprehension. *arXiv preprint: 1603.06127*, 2016.

Anton Benz and Katja Jasinskaja. Questions Under Discussion: From Sentence to Discourse. *Discourse Processes*, 54(3):177–186, April 2017. ISSN 0163-853X, 1532-6950. doi: 10.1080/0163853X.2017.1316038.

James Bergstra, Dan Yamins, and David D. Cox. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20, 2013.

Steven Bird and Edward Loper. NLTK: The Natural Language Toolkit. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*,

pages 214–217, Barcelona, Spain, 2004. Association for Computational Linguistics.

David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4): 77–84, 2012. ISSN 00010782. doi: 10.1145/2133806.2133826.

David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*, pages 113–120, Pittsburgh, Pennsylvania, 2006. ACM Press. ISBN 978-1-59593-383-6. doi: 10.1145/1143844.1143859.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning research*, 3:993–1022, 2003.

Mohan John Blooma and Jayan Chirayath Kurian. Research Issues In Community Based Question Answering. In *Pacific Asia Conference on Information Systems, PACIS 2011: Quality Research in Pacific Asia*, pages 1–13, Brisbane, Australia, 2011.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146, 2017.

Daniele Bonadiman, Antonio Uva, and Alessandro Moschitti. Effective Shared Representations with Multitask Learning for Community Question Answering. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 726–732, Valencia, Spain, 2017. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint: 1508.05326*, 2015.

Jordan Boyd-Graber, Yuening Hu, and David Mimno. Applications of Topic Models. *Foundations and Trends in Information Retrieval*, 2-3(11):143–296, 2017.

René Brauer, Mirek Dymitrow, and Mats Fridlund. The digital shaping of humanities research: The emergence of Topic Modeling within historical studies. *Enacting Futures: DASTS 2014*, 2014.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on*

Semantic Evaluation (SemEval-2017), pages 1–14, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001.

Delphine Charlet and Geraldine Damnati. SimBow at SemEval-2017 Task 3- Soft-Cosine Semantic Similarity between Questions for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 315–319, Vancouver, Canada, 2017. Association for Computational Linguistics.

Daoyuan Chen, Yaliang Li, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, and Jingren Zhou. AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search. *arXiv:2001.04246 [cs]*, January 2020.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1152.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. Guided Alignment Training for Topic-Aware Neural Machine Translation. In *Proceedings of AMTA*, pages 121–134, Austin, USA, July 2016.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint: 1406.1078*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555 [cs]*, December 2014.

Daniel Cohen and W. Bruce Croft. End to End Long Short Term Memory Networks for Non-Factoid Question Answering. pages 143–146. ACM Press, 2016. ISBN 978-1-4503-4497-5. doi: 10.1145/2970398.2970438.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1070.

Ido Dagan and Oren Glickman. Probabilistic textual entailment: Generic applied modeling of language variability. In *Learning Methods for Text Understanding and Mining 2004*, pages 26–29, 2004.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii, October 2009. ISSN 1351-3249, 1469-8110. doi: 10.1017/S1351324909990209.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd International Conference on World Wide Web - WWW '13*, pages 307–318, Rio de Janeiro, Brazil, 2013. ACM Press. ISBN 978-1-4503-2035-1. doi: 10.1145/2488388.2488416.

Jan Milan Deriu and Mark Cieliebak. SwissAlps at SemEval-2017 Task 3: Attention-based Convolutional Neural Network for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation.*, volume 17, pages 334–338, Vancouver, Canada, 2017. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 4171—4186, Minneapolis, USA, 2019. Association for Computational Linguistics.

Lee R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302, July 1945. ISSN 00129658. doi: 10.2307/1932409.

Ran Ding, Ramesh Nallapati, and Bing Xiang. Coherence-Aware Neural Topic Modeling. *arXiv:1809.02687 [cs]*, September 2018.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*, February 2020.

William B. Dolan and Chris Brockett. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP@IJCNLP)*, pages 9–16, Jeju Island, Korea, 2005. Asian Federation of Natural Language Processing.

Andrei Dulceanu, Thang Le Dinh, Walter Chang, Trung Bui, Doo Soon Kim, Manh Chien Vu, and Seokhwan Kim. PhotoshopQuiA: A Corpus of Non-Factoid Questions and Answers for Why-Question Answering. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018*, pages 2763–2770, Miyazaki, Japan, 2018. European Language Resources Association.

Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, March 1990. ISSN 03640213. doi: 10.1207/s15516709cog1402_1.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting Word Vectors to Semantic Lexicons. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2015)*, pages 1606–1615, Denver, USA, March 2015. Association for Computational Linguistics.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820, Scottsdale, USA, December 2015. IEEE. ISBN 978-1-4799-7291-3. doi: 10.1109/ASRU.2015.7404872.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. Beihang-MSRA at SemEval-2017 Task 3- A Ranking System with Neural Matching Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 280–286, Vancouver, Canada, 2017. Association for Computational Linguistics.

Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52, 2008.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *SemEval@ NAACL-HLT*, pages 1116–1123, 2016.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. KeLP at SemEval-2017 Task 3- Learning Pairwise Patterns in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 326–333, Vancouver, Canada, 2017. Association for Computational Linguistics.

John R. Firth. A Synopsis of Linguistic Theory, 1930–1955. In *Selected Papers of J. R. Firth (1952-59)*, pages 168–205, 1957.

Robert French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, April 1999. ISSN 13646613. doi: 10.1016/S1364-6613(99)01294-2.

Byron Galbraith, Bhanu Pratap, and Daniel Shank. Talla at SemEval-2017 Task 3: Identifying Similar Questions Through Paraphrase Detection. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 375–379, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2062.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. TANDA: Transfer and Adapt Pre-Trained Transformer Models for Answer Sentence Selection. *arXiv:1911.04118 [cs]*, November 2019.

Benjamin Ross George. *Question Embedding and the Semantics of Answers*. University of California, Los Angeles, 2011. ISBN 1-267-14505-6.

Martin Gerlach and Francesc Font-Clos. A Standardized Project Gutenberg Corpus for Statistical Analysis of Natural Language and Quantitative Linguistics. *Entropy*, 22(1):126, January 2020. ISSN 1099-4300. doi: 10.3390/e22010126.

Reza Ghaeini, Xiaoli Z. Fern, and Prasad Tadepalli. Interpreting Recurrent and Attention-Based Neural Models: A Case Study on Natural Language Inference. *arXiv:1808.03894 [cs]*, August 2018.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. Contextual LSTM (CLSTM) Models for Large Scale NLP Tasks. *arXiv:1602.06291 [cs]*, February 2016.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Aistats*, volume 15, page 275, 2011.

Yoav Goldberg. A Primer on Neural Network Models for Natural Language Processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420, 2016.

Yoav Goldberg. Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, April 2017. ISSN 1947-4040, 1947-4059. doi: 10.2200/S00762ED1V01Y201703HLT037.

Yichen Gong, Heng Luo, and Jian Zhang. Natural Language Inference over Interaction Space. In *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

Saurabh Goyal, Anamitra Roy Choudhary, Venkatesan Chakaravarthy, Saurabh ManishRaje, Yogish Sabharwal, and Ashish Verma. PoWER-BERT: Accelerating BERT inference for Classification Tasks. *arXiv:2001.08950 [cs, stat]*, January 2020.

Bela Gipp, Norman Meuschke, Corinna Breitinger, Jim Pitman, and Andreas Nürnberger. Web-based Demonstration of Semantic Similarity Detection Using Citation Pattern Visualization for a Cross Language Plagiarism Case:. In *Proceedings of the 16th International Conference on Enterprise Information Systems*, pages 677–683, Lisbon, Portugal, 2014. SCITEPRESS - Science and and Technology Publications. ISBN 978-989-758-028-4. doi: 10.5220/0004985406770683.

Lin Gui, Jia Leng, Gabriele Pergola, Yu Zhou, Ruifeng Xu, and Yulan He. Neural Topic Model with Reinforcement Learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3476–3481, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1350.

Lin Gui, Leng Jia, Jiyun Zhou, Ruifeng Xu, and Yulan He. Multi-Task Learning with Mutual Learning for Joint Sentiment Classification and Topic Detection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2020.2999489.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. Domain Adaptation with Latent Semantic Association for Named Entity Recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09*, pages 281–289, Boulder, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1. doi: 10.3115/1620754.1620795.

Amulya Gupta and Zhu Zhang. To Attend or not to Attend: A Case Study on Syntactic Structures for Semantic Relatedness. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2116–2125, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1197.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation Artifacts in Natural Language Inference Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2017.

Zellig S. Harris. Distributional Structure. WORD, 10(2-3):146–162, August 1954. ISSN 0043-7956, 2373-5112. doi: 10.1080/00437956.1954.11659520.

Hua He, Kevin Gimpel, and Jimmy Lin. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1576–1586, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1181.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

Liangjie Hong and Brian D. Davison. Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics - SOMA '10*, pages 80–88, Washington D.C., USA, 2010. ACM Press. ISBN 978-1-4503-0217-3. doi: 10.1145/1964858.1964870.

Yuening Hu, Ke Zhai, Vladimir Eidelman, and Jordan Boyd-Graber. Polylingual Tree-Based Topic Models for Translation Domain Adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1166–1176, Baltimore, Maryland, 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1110.

Chieh-Yang Huang, Nicole Peinelt, and Lun-Wei Ku. Automatically Suggesting Example Sentences of Near-Synonyms for Language Learners. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 302–306, Osaka, Japan, 2016.

Adrian Iftene. UAIC Participation at RTE4. In *Proceedings of the First Text Analysis Conference, TAC 2008*, pages 1–10, Gaithersburg, USA, 2008. NIST.

Yangfeng Ji and Jacob Eisenstein. Discriminative Improvements to Distributional Sentence Similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), A Meeting of SIGDAT, a Special Interest Group of the ACL*, pages 891–896, Seattle, USA, 2013. Association for Computational Linguistics.

Valentin Jijkoun and Maarten de Rijke. Recognizing Textual Entailment Using Lexical Similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 73–76, 2005.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *arXiv:1907.11932 [cs]*, January 2020.

Qiang Jipeng, Qian Zhenyu, Li Yun, Yuan Yunhao, and Wu Xindong. Short Text Topic Modeling Techniques, Applications, and Performance: A Survey. *arXiv:1904.07695 [cs]*, April 2019.

Wei-Tsung Kao, Tsung-Han Wu, Po-Han Chi, Chun-Cheng Hsieh, and Hung-Yi Lee. Further Boosting BERT-based Models by Duplicating Existing Layers: Some Intriguing Phenomena inside BERT. *arXiv:2001.09309 [cs]*, January 2020.

Divyansh Kaushik and Zachary C Lipton. How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium, 2018. Association for Computational Linguistics.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, 2014. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014.

Yuta Koreeda, Takuya Hashito, Yoshiki Niwa, Misa Sato, Toshihiko Yanase, Kenzo Kurotsuchi, and Kohsuke Yanai. Bunji at SemEval-2017 Task 3: Combination of Neural Similarity Features and Comment Plausibility Features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 353–359, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2058.

Milen Kouylekov and Bernardo Magnini. Tree edit distance for textual entailment. In Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *Current Issues in Linguistic Theory*, volume 292, pages 167–176. John Benjamins Publishing Company, Amsterdam, 2007. ISBN 978-90-272-4807-7 978-90-272-9128-8. doi: 10.1075/cilt.292.22kou.

Olga Kovaleva, Anna Rumshisky, and Alexey Romanov. Similarity-Based Reconstruction Loss for Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4875–4880, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1525.

Katsiaryna Krasnashchok and Salim Jouili. Improving Topic Quality by Promoting Named Entities in Topic Modeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 247–253, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2040.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466, March 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00276.

Tuan Manh Lai, Trung Bui, and Sheng Li. A Review on Deep Learning Techniques Applied to Answer Selection. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 2132–2144, Santa Fe, USA, 2018. Association for Computational Linguistics.

Tuan Manh Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. A Simple but Effective BERT Model for Dialog State Tracking on Resource-Limited Systems. *arXiv:1910.12995 [cs]*, January 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv:1909.11942 [cs]*, January 2020.

Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, JMLR Workshop and Conference Proceedings}, pages 1188–1196, Beijing, China, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998a.

Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998b.

Omer Levy and Yoav Goldberg. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, USA, 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-2050.

Jianhua Lin. Divergence Measures based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Junyang Lin, An Yang, Yichang Zhang, Jie Liu, Jingren Zhou, and Hongxia Yang. InterBERT: Vision-and-Language Interaction for Multi-modal Pretraining. *arXiv:2003.13198 [cs]*, March 2020.

Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, 5(1), December 2016. ISSN 2193-1801. doi: 10.1186/s40064-016-3252-8.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019.

Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Representation Learning for Natural Language Processing*. Springer Singapore, Singapore, 2020. ISBN 9789811555725 9789811555732. doi: 10.1007/978-981-15-5573-2.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *arXiv:1908.02265 [cs]*, August 2019.

Thang Luong, Richard Socher, and Christopher Manning. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013*, pages 104–113, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. Re-examining Machine Translation Metrics for Paraphrase Identification. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 182–190, Montreal, Canada, 2012. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. End-to-End Bias Mitigation by Modelling Biases in Corpora. *arXiv:1909.06321 [cs]*, April 2020.

Goutam Majumder, Partha Pakray, Alexander Gelbukh, and David Pinto. Semantic Textual Similarity Methods, Tools, and Applications: A Survey. *Computación y Sistemas*, 20(4), December 2016. ISSN 2007-9737, 1405-5546. doi: 10.13053/cys-20-4-2506.

Erwin Marsi and Emiel Krahmer. Annotating a parallel monolingual treebank with semantic similarity relations. In *The Sixth International Workshop on Treebanks and Linguistic Theories (TLT'07)*, volume 1, pages 85–96, 2007.

Jon D Mcauliffe and David M Blei. Supervised Topic Models. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, pages 121–128, Vancouver, Canada, 2008. Curran Associates.

Philip M. McCarthy, Rebekah H. Guess, and Danielle S. McNamara. The components of paraphrase evaluations. *Behavior Research Methods*, 41(3):682–690, August 2009. ISSN 1554-351X, 1554-3528. doi: 10.3758/BRM.41.3.682.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1334.

Yashar Mehdad, Alessandro Moschitti, and Fabio Massiomo Zanzotto. SemKer: Syntactic/Semantic Kernels for Recognizing Textual Entailment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1020–1028. Association for Computational Linguistics, 2010.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. Discovering Discrete Latent Topics with Neural Variational Inference. *arXiv:1706.00359 [cs]*, May 2018.

Todor Mihaylov and Preslav Nakov. SemanticZ at SemEval-2016 Task 3: Ranking Relevant Answers in Community Question Answering Using Semantic Similarity Based on Fine-tuned Word Embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2016)*, pages 879–886, San Diego, USA, 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1136.

Tsvetomila Mihaylova, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, Mitra Mohtarami, Georgi Karadzhov, and James Glass. Fact Checking in Community Forums. *arXiv preprint: 1803.03178*, 2018.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations (ICLR 2013)*, Scottsdale, USA, 2013.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pages 752–762, Portland, USA, 2011. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting Word Vectors to Linguistic Constraints. *arXiv:1603.00892 [cs]*, March 2016.

Jonas Mueller and Aditya Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 2786–2792, Phoenix, USA, 2016. AAAI Press.

Preslav Nakov, Lluis Marquez, Magdy Walid, Alessandro Moschitti, James Glass, and Bilal Randeree. SemEval-2015 task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic*

Evaluation (SemEval@NAACL-HLT 2015), pages 269–281, Denver, USA, 2015. Association for Computational Linguistics.

Preslav Nakov, Lluis Marquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, James Glass, and Bilal Randeree. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2016)*, pages 525–545, San Diego, USA, 2016. Association for Computational Linguistics.

Preslav Nakov, Doris Hoogeveen, Llúis Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 27–48, Vancouver, Canada, 2017. Association for Computational Linguistics.

Titas Nandi, Chris Biemann, Seid Muhie Yimam, Deepak Gupta, Sarah Kohail, Asif Ekbal, and Pushpak Bhattacharyya. IIT-UHH at SemEval-2017 Task 3: Exploring Multiple Features for Community Question Answering and Implicit Dialogue Identification. 2017.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807, Brussels, Belgium, 2018. Association for Computational Linguistics.

Dong Nguyen and Carolyn P Rose. Language use as a reflection of socialization in online communities. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 76–85, 2011.

Hoang-Quoc Nguyen-Son, Yusuke Miyao, and Isao Echizen. Paraphrase Detection Based on Identical Phrase and Similar Word Matching. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC 29)*, pages 504–512, Shanghai, China, 2015. Association for Computational Linguistics.

Christopher Olah. Understanding LSTM Networks – colah's blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

Malte Ostendorff, Peter Bourgonje, Maria Berger, Julian Moreno-Schneider, Georg Rehm, and Bela Gipp. Enriching BERT with Knowledge Graph Embeddings

for Document Classification. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS)*, Erlangen, Germany, October 2019.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text Matching as Image Recognition. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 2793–2799, Phoenix, USA, 2016. AAAI Press.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2249–2255, Austin, USA, 2016. Association for Computational Linguistics.

Nicole Peinelt, Maria Liakata, and Shu-Kai Hsieh. ClassifierGuesser: A Context-based Classifier Prediction System for Chinese Language Learners. *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 41–44, 2017.

Nicole Peinelt, Maria Liakata, and Dong Nguyen. Aiming beyond the Obvious: Identifying Non-Obvious Cases in Semantic Similarity Datasets. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2792–2798, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1268.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. Better Early than Late: Fusing Topics with Word Embeddings for Neural Question Paraphrase Identification. *arXiv:2007.11314 [cs]*, July 2020a.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7047–7055, Seattle, USA, 2020b. Association for Computational Linguistics.

Nicole Peinelt, Marek Rei, and Maria Liakata. GiBERT: Introducing Linguistic Knowledge into BERT through a Lightweight Gated Injection Method. *arXiv:2010.12532 [cs]*, October 2020c.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162.

Diana Perez and Enrique Alfonseca. Application of the Bleu algorithm for recognising textual entailments. In *Proceedings of the First Challenge Workshop Recognising Textual Entailment*, pages 9–12, 2005.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, New Orleans, USA, February 2018. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge Enhanced Contextual Word Representations. *arXiv:1909.04164 [cs]*, October 2019.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Mapping Dependencies Trees: An Application to Question Answering. *Computational Linguistics 6*, (9), 2004.

Le Qi, Yu Zhang, and Ting Liu. SCIR-QA at SemEval-2017 Task 3: CNN Model Based on Similar and Dissimilar Information between Keywords for Question Similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 305–309, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2049.

Zengchang Qin, Marcus Thint, and Zhiheng Huang. Ranking Answers by Hierarchical Topic Models. In *Next-Generation Applied Intelligence, 22nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2009)*, volume 5579, pages 103–112, Tainan, Taiwan, 2009. Springer. ISBN 978-3-642-02567-9 978-3-642-02568-6. doi: 10.1007/978-3-642-02568-6_11.

Dragomir Radev. A Common Theory of Information Fusion from Multiple Text Sources Step One: Cross-Document Structure. In *Proceedings of the SIGDIAL 2000 Workshop, The 1st Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 74–83, Hong Kong, China, 2000. Association for Computational Linguistics.

Dragomir Radev, Jahna Otterbacher, and Zhu Zhang. CST bank: A corpus for the study of cross-document structural relationships. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

Rajat Raina, Aria Haghighi, Christopher Cox, Jenny Finkel, Jeff Michels, Kristina Toutanova, Bill MacCartney, Marie-Catherine de Marneffe, Christopher D. Manning, and Andrew Y. Ng. Robust textual inference using diverse knowledge sources. In *Proc. of the 1st. PASCAL Recognision Textual Entailment Challenge Workshop, Southampton, UK*, pages 57–60, 2005.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics.

Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. Bridging the Gap between Relevance Matching and Semantic Matching for Short Text Similarity Modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5369–5380, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1540.

Radim Rehurek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 46–50, Valletta, Malta, 2010. European Language Resources Association.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3980–3990, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. In *4th International Conference on Learning Representations (ICLR 2016)*, San Juan, Puerto Rico, 2016.

Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the Space of Topic Coherence Measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*, pages 399–408, Shanghai, China, 2015. ACM Press. ISBN 978-1-4503-3317-7. doi: 10.1145/2684822.2685324.

Joao Rodrigues, Chakaveh Saedi, Antonio Branco, and Joao Silva. Semantic Equivalence Detection: Are Interrogatives Harder than Declaratives? In *Proceedings of the 11th International Conference on Language Resources and Evaluation*, pages 3248–3253, Miyazaki, Japan, 2018. European Language Resources Association.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A Primer in BERTology: What we know about how BERT works. *arXiv:2002.12327 [cs]*, February 2020.

Yu-Ping Ruan, Zhen-Hua Ling, Jia-Chen Gu, and Quan Liu. Fine-Tuning BERT for Schema-Guided Zero-Shot Dialogue State Tracking. *arXiv:2002.00181 [cs]*, February 2020.

Vasile Rus. Opportunities and Challenges in Semantic Similarity. In *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference (FLAIRS 2014)*, pages 244–249, Pensacola Beach, USA, 2014. AAAI Press.

Vasile Rus, Mihai Lintean, Rajendra Banjade, Nobal Niraula, and Dan Stefanescu. SEMILAR: The Semantic Similarity Toolkit. In *51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 163–168, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Vasile Rus, Rajendra Banjade, and Mihai Lintean. On Paraphrase Identification Corpora. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 14)*, pages 2422–2429, Reykjavik, Iceland, 2014. European Language Resources Association (ELRA).

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, volume SIGDAT, pages 379–389, Lisbon, Portugal, 2015. Association for Computational Linguistics.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor Man's BERT: Smaller and Faster Transformer Models. *arXiv:2004.03844 [cs]*, April 2020.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*, October 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162.

Aliaksei Severyn and Alessandro Moschitti. Automatic Feature Engineering for Answer Selection and Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 458–467, Seattle, USA, 2013. Association for Computational Linguistics.

Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. Learning Context-Sensitive Convolutional Filters for Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1839–1848, Brussels, Belgium, 2018a. Association for Computational Linguistics. doi: 10.18653/v1/D18-1210.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Melbourne, Australia, May 2018b. Association for Computational Linguistics.

Gaurav Singh, Zahra Sabet, John Shawe-Taylor, and James Thomas. Constructing Artificial Data for Fine-tuning for Low-Resource Biomedical Text Tagging with Applications in PICO Annotation. *arXiv:1910.09255 [cs, stat]*, January 2020.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 801–809, Granada, Spain, 2011.

Ivan Srba and Maria Bielikova. A Comprehensive Survey and Classification of Approaches for Community Question Answering. *ACM Transactions on the Web*, 10(3):1–63, August 2016. ISSN 15591131. doi: 10.1145/2934687.

Akash Srivastava and Charles Sutton. Autoencoding Variational Inference For Topic Models. In *5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France, March 2017.

Pavel Stefanovič, Olga Kurasova, and Rokas Štrimaitis. The N-Grams Based Text Similarity Detection Approach Using Self-Organizing Maps and Similarity Measures. *Applied Sciences*, 9(9):1870, May 2019. ISSN 2076-3417. doi: 10.3390/app9091870.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. Multiway Attention Networks for Modeling Sentence Pairs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4411–4417, Stockholm, Sweden, 2018.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. LSTM-Based Deep Learning Models for Non-Factoid Answer Selection. *Proceedings of ICLR 2016*, 2016.

Doina Ta˘tar, Gabriela Serban, Andreea Mihis, and Rada Mihalcea. Textual Entailment as a Directional Relation. *Journal of Research and Practice in Information Technology*, 41(1):12, 2009.

Yi Tay, Aston Zhang, Anh Tuan Luu, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1494–1503, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1145.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December 2006. ISSN 0162-1459, 1537-274X. doi: 10.1198/016214506000000302.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. Neural Paraphrase Identification of Questions with Noisy Pretraining. In *Proceedings of the First Workshop on Subword and Character Level Models*

in NLP, pages 142–147, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4121.

Zhou Tong and Haiyi Zhang. A Text Mining Research Based on LDA Topic Modelling. In *Computer Science & Information Technology (CS & IT)*, pages 201–210. Academy & Industry Research Collaboration Center (AIRCC), May 2016. ISBN 978-1-921987-51-9. doi: 10.5121/csit.2016.60616.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. JAIST: Combining Multiple Features for Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 215–219, Denver, USA, 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2038.

Jun'ichi Tsujii. Computational Linguistics and Natural Language Processing. In Alexander F. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 6608, pages 52–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-19399-6 978-3-642-19400-9. doi: 10.1007/978-3-642-19400-9_5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008, Long Beach, USA, 2017.

Ike Vayansky and Sathish A.P. Kumar. A review of topic modeling methods. *Information Systems*, 94:101582, December 2020. ISSN 03064379. doi: 10.1016/j.is.2020.101582.

Tedo Vrbanec and Ana Meštrović. Corpus-Based Paraphrase Detection Experiments and Review. *Information*, 11(5):241, April 2020. ISSN 2078-2489. doi: 10.3390/info11050241.

Soumya Wadhwa, Khyathi Raghavi Chandu, and Eric Nyberg. Comparative Analysis of Neural QA models on SQuAD. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 89–97, Melbourne, Australia, July 2018a. Association for Computational Linguistics.

Soumya Wadhwa, Varsha Embar, Matthias Grabmair, and Eric Nyberg. Towards Inference-Oriented Reading Comprehension: ParallelQA. In *Proceedings of the*

Workshop on Generalization in the Age of Deep Learning, pages 1–7, New Orleans, Louisiana, 2018b. Association for Computational Linguistics. doi: 10.18653/v1/W18-1001.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. A Reinforced Topic-Aware Convolutional Sequence-to-Sequence Model for Abstractive Text Summarization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4453–4460, Stockholm, Sweden, May 2018.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 22–32, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu ji, Cuihong Cao, Daxin Jiang, and Ming Zhou. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. *arXiv:2002.01808 [cs]*, February 2020a.

Shuohang Wang and Jing Jiang. A Compare-Aggregate Model for Matching Text Sequences. In *5th International Conference on Learning Representations (ICLR 2017)*, November 2016.

Shuohang Wang, Yunshi Lan, Yi Tay, Jing Jiang, and Jingjing Liu. Multi-level Head-wise Match and Aggregation in Transformer for Textual Sequence Matching. *arXiv:2001.07234 [cs]*, January 2020b.

Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4144–4150, Melbourne, Australia, 2017. ISBN 0-9992411-0-9.

Christian Wartena. Distributional Similarity of Words with Different Frequencies. In *Proceedings of the 13th Dutch-Belgian Workshop on Information Retrieval*, volume 986 of *CEUR Workshop Proceedings*, pages 8–11, Delft, The Netherlands, 2013. CEUR-WS.org.

Julie Weeds, David Weir, and Bill Keller. The distributional similarity of subparses. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment - EMSEE '05*, pages 7–12, Ann Arbor, Michigan, 2005. Association for Computational Linguistics. doi: 10.3115/1631862.1631864.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making Neural QA as Simple as Possible but not Simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1028.

Junfeng Wen, Yanshuai Cao, and Ruitong Huang. Few-Shot Self Reminder to Overcome Catastrophic Forgetting. *arXiv:1812.00543 [cs, stat]*, December 2018.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the Association for Computational Linguistics*, 3:345–358, December 2015. ISSN 2307-387X. doi: 10.1162/tacl_a_00143.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards Universal Paraphrastic Sentence Embeddings. In *4th International Conference on Learning Representations (ICLR 2016)*, page 19, San Juan, Puerto Rico, 2016.

Guoshun Wu, Yixuan Sheng, Man Lan, and Yuanbin Wu. ECNU at SemEval-2017 Task 3- Using Traditional and Deep Learning Methods to Address Community Question Answering Task. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 356–360, Vancouver, Canada, 2017a. Association for Computational Linguistics.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. Phrase-level Self-Attention Networks for Universal Sentence Encoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3729–3738, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1408.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *Transactions of the Association for Computational Linguistics*, pages 339–351, 2017b.

Yufei Xie, Maoquan Wang, Jian Jiang, and Zhao Lu. EICA Team at SemEval-2017 Task 3: Semantic and Metadata-based Features for Community Question Answer-

ing. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 356–360, 2017.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. BERT-of-Theseus: Compressing BERT by Progressive Module Replacing. *arXiv:2002.02925 [cs]*, February 2020.

Zhenyu Xuan, Chuyu Ma, and Shengyi Jiang. FGN: Fusion Glyph Network for Chinese Named Entity Recognition. *arXiv preprint: 2001.05272*, 2020.

Mohammad-Ali Yaghoub-Zadeh-Fard, Boualem Benatallah, Moshe Chai Barukh, and Shayan Zamanirad. A Study of Incorrect Paraphrases in Crowdsourced User Utterances. In *Proceedings of NAACL-HLT 2019*, pages 295–306. Association for Computational Linguistics, 2019.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *EMNLP*, pages 2013–2018, 2015.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning Discriminative Projections for Text Similarity Measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, pages 247–256, Portland, USA, 2011. Association for Computational Linguistics.

Jianhua Yin and Jianyong Wang. A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14*, pages 233–242, New York, USA, 2014. ACM Press. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623715.

Wenpeng Yin and Hinrich Schütze. Convolutional Neural Network for Paraphrase Identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado, 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1091.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *arXiv preprint: 1412.1632*, 2014.

Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*, December 2012.

Guanhua Zhang, Bing Bai, Jian Liang, Kun Bai, Shiyu Chang, Mo Yu, Conghui Zhu, and Tiejun Zhao. Selection Bias Explorations and Debias Methods for Natural Language Sentence Matching Datasets. *arXiv:1905.06221 [cs]*, May 2019a.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations (ICLR 2020)*, Addis Ababa, Ethiopia, April 2019b.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint: 1510.03820*, 2015.

Yitao Zhang and Jon Patrick. Paraphrase Identification by Text Canonicalization. In *Proceedings of the Australasian Language Technology Workshop, (ALTA 2005)*, pages 160–166, Sydney, Australia, 2005. Australasian Language Technology Association.

Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, USA, April 2019c. Association for Computational Linguistics.

Yue Zhang, Rui Wang, and Luo Si. Syntax-Enhanced Self-Attention-Based Semantic Role Labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 616–626, Hong Kong, China, 2019d. Association for Computational Linguistics. doi: 10.18653/v1/D19-1057.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware BERT for Language Understanding. *arXiv:1909.02209 [cs]*, February 2020.

Lingyun Zhao, Lin Li, and Xinhao Zheng. A BERT based Sentiment Analysis and Key Entity Detection Approach for Online Financial Texts. *arXiv:2001.05326 [cs]*, 2020.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1053.

Lixing Zhu, Yulan He, and Deyu Zhou. A Neural Generative Model for Joint Learning Topics and Topic-Specific Word Embeddings. *arXiv preprint:2008.04702 [cs]*, August 2020.

Appendix

A Aiming beyond the Obvious - Identifying Non-Obvious Cases in Semantic Similarity Datasets

Aiming beyond the Obvious: Identifying Non-Obvious Cases in Semantic Similarity Datasets

Nicole Peinelt[1,2] and **Maria Liakata**[1,2] and **Dong Nguyen**[1,3]
[1]The Alan Turing Institute, London, UK
[2]University of Warwick, Coventry, UK
[3]Utrecht University, Utrecht, The Netherlands
{[redacted]

Abstract

Existing datasets for scoring text pairs in terms of semantic similarity contain instances whose resolution differs according to the degree of difficulty. This paper proposes to distinguish obvious from non-obvious text pairs based on superficial lexical overlap and ground-truth labels. We characterise existing datasets in terms of containing difficult cases and find that recently proposed models struggle to capture the non-obvious cases of semantic similarity. We describe metrics that emphasise cases of similarity which require more complex inference and propose that these are used for evaluating systems for semantic similarity.

1 Introduction

Modelling semantic similarity between a pair of texts is a fundamental task in NLP with a wide range of applications (Baudiš et al., 2016). One area of active research is Community Question Answering (CQA) (Nakov et al., 2017; Bonadiman et al., 2017), which is concerned with the automatic answering of questions based on user generated content from Q&A websites (e.g. StackExchange) and requires modelling the semantic similarity between question and answer pairs. Another well-studied task is paraphrase detection (Socher et al., 2011; He et al., 2015; Tomar et al., 2017), which models the semantic equivalence between a pair of sentences.

Evaluation for such tasks has primarily focused on metrics, such as mean average precision (MAP), F1 or accuracy, which give equal weights to all examples, regardless of their difficulty. However, as illustrated by the examples in Table 1, not all items within text pair similarity datasets are equally difficult to resolve.

Recent work has shown the need to better understand limitations of current models and datasets in natural language understanding (Wadhwa et al., 2018a; Rajpurkar et al., 2018). For example, Kaushik and Lipton (2018) showed that models sometimes exploit dataset properties to achieve high performance even when crucial task information is withheld, and Gururangan et al. (2018) demonstrated that model performance is inflated by annotation artefacts in natural language inference tasks.

id	case	documents
160174	P_o	what's the origin of the word o'clock? what is the origin of the word o'clock?
115695	P_n	which is the best way to learn coding? how do you learn to program?
193190	N_o	what are the range of careers in biotechnology in indonesia? how do you tenderize beef stew meat?
268368	N_n	what is meant by 'e' in mathematics? what is meant by mathematics?

Table 1: Examples for difficulty cases from the development set of the Quora dataset. o=obvious, n=non-obvious, N=negative label, P=positive label

In this paper, we analyse current datasets and recently proposed models by focusing on item difficulty based on shallow lexical overlap. Rodrigues et al. (2018) found declarative CQA sentence pairs to be more difficult to resolve than interrogative pairs as the latter contain more cases of superficial overlap. In addition, Wadhwa et al. (2018b) showed that competitive neural reading comprehension models are susceptible to shallow patterns (e.g. lexical overlap). Our study digs deeper into these findings to investigate the properties of current text pair similarity datasets with respect to different levels of difficulty and evaluates models based on how well they can resolve difficult cases.

We make the following contributions:

1. We propose a criterion to distinguish between obvious and non-obvious examples in text

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2792–2798
Florence, Italy, July 28 - August 2, 2019.

pair similarity datasets (section 4).

2. We characterise current datasets in terms of the extent to which they contain obvious vs. non-obvious items (section 4).

3. We propose alternative evaluation metrics based on example difficulty (section 5) and provide a reference implementation at https://github.com/wuningxi/LexSim.

2 Datasets and Tasks

We selected well-known benchmark datasets differing in size (small vs. large), document length (single sentence vs. multi-sentence), document types (declarative vs. interrogative) and tasks (answer ranking vs. paraphrase detection vs. similarity scoring), see Table 2.

SemEval The SemEval Community Question Answering (CQA) dataset (Nakov et al., 2015, 2016, 2017) contains posts from the online forum Qatar Living. The task is to rank relevant posts above non-relevant ones. Each subtask involves an initial post and 10 possibly relevant posts with binary annotations. Task A contains questions and comments from the same thread, task B involves question paraphrases, and task C is similar to A but contains comments from an external thread.

MSRP The Microsoft Research Paraphrase corpus (MSRP) is a popular paraphrase detection dataset, consisting of pairs of sentences with binary judgments (Dolan and Brockett, 2005).

Name	Task	Type	Size
SemEval	(A) answer ranking	rank	26K
	(B) paraphrase ranking	rank	4K
	(C) answer ranking	rank	47K
Quora	paraphrase detection	class	404K
MSRP	paraphrase detection	class	5K
STS	similarity scoring	regr	8K

Table 2: Selected text pair similarity data sets. Size as number of text pairs. rank=ranking task, class=classification task, regr=regression task.

Quora The Quora duplicate questions dataset contains a large number of question pairs with binary labels[1]. The task is to predict whether two questions are paraphrases, similar to Task B of SemEval, but it is framed as a classification rather than a ranking problem. We use the same training / development / test set partition as Wang et al. (2017).

STS The Semantic Textual Similarity Benchmark (STS) dataset (Cer et al., 2017) consists of a selection of STS SemEval shared tasks (2012-2017). It contains sentence pairs annotated with continuous semantic relatedness scores on a scale from 0 (low similarity) to 5 (high similarity).

In this paper, we focus on predicting the semantic similarity between two text snippets in a binary classification scenario, as the ranking scenario is only applicable to some of the datasets. Binary labels are already provided for all tasks except for

[1]https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning

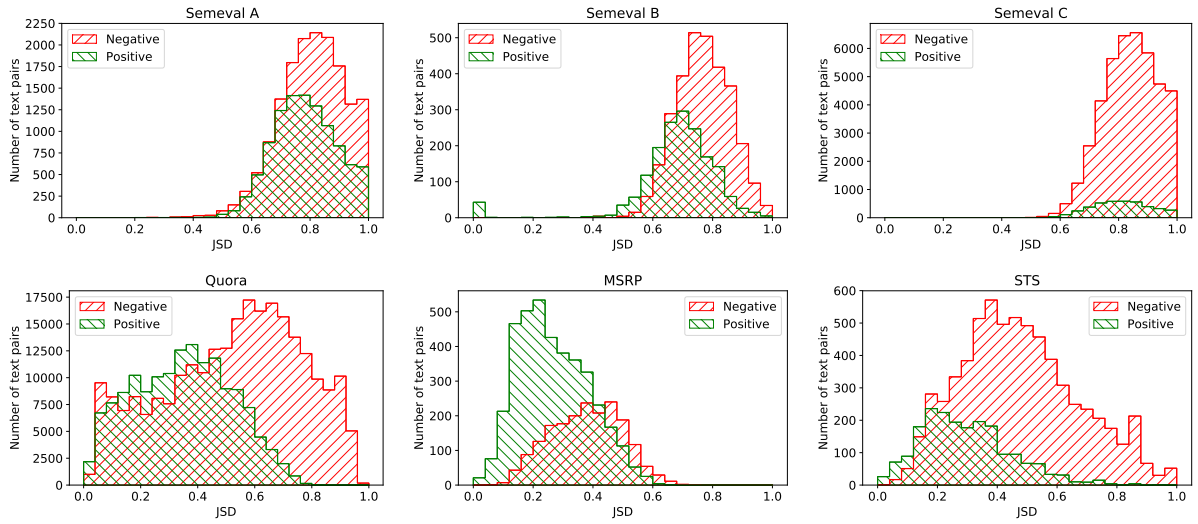


Figure 1: Lexical divergence distribution by labels across datasets. JSD=Jensen-Shannon divergence.

STS. In the case of STS, we convert the scores into binary labels. Based on the description of the relatedness scores in Cer et al. (2017), we assign a positive label if relatedness ≥ 4 and a negative one otherwise to use a similar criterion as in the other datasets.

3 Lexical divergence in current datasets

To characterise the datasets, we represent the text pairs as two distributions over words and measure their lexical divergence using Jensen-Shannon divergence (JSD) (Lin, 1991).[2] Figure 1 shows the entire JSD distribution by label for each dataset.

The datasets differ with respect to the degree of lexical divergence they contain: The three SemEval CQA datasets show a high degree of lexical divergence (majority > 0.5), especially in the external QA scenario (task C). Text pairs in MSRP tend to have low-medium JSD scores (majority < 0.6), while items in Quora and STS show the widest range of lexical divergence (see also Appendix A). Overall, pairs with negative labels tend to have higher JSD scores than pairs with positive labels. Especially in Quora, MSRP and STS, distinct distributions emerge for positive vs. negative labels, providing direct clues for label assignment.

4 Distinguishing between obvious and non-obvious examples

As shown, pairs with high lexical divergence tend to have a negative label in the above datasets (e.g. N_o in Table 1), while low lexical divergence is associated with a positive label (e.g. P_o in Table 1). Intuitively, these are cases which should be relatively easy to identify. More difficult are text pairs with a positive label but high lexical divergence (e.g. P_n in Table 1), or a negative label despite low lexical divergence (e.g. N_n in Table 1). We use Table 3 to categorise cases in terms of their difficulty level.

	positive label	negative label
low div	obvious pos (P_o)	non-obvious neg (N_n)
high div	non-obvious pos (P_n)	obvious neg (N_o)

Table 3: Defining obvious and non-obvious similarity cases based on labels and lexical overlap.

[2]We also calculated set-based similarity metrics (Jaccard Index and Dice Coefficient) and found consistent results with JSD, but give preference to the distribution-based metric which is more natural for text. Due to space restrictions, we only report JSD in this paper.

	Fleiss' Kappa	Avg. time per pair	Instances
P_o	0.6429	11.58s	35
P_n	0.0878	11.68s	15
N_o	0.3886	12.50s	34
N_n	0.0892	13.83s	16
total	0.6267	12.27s	100

Table 4: Statistics for manual annotation on Quora. o=obvious, n=non-obvious, N=negative, P=positive

	SemEval			Quora	MSRP	STS
	A	B	C			
P_o	5893	1162	2492	107612	2398	1597
P_n	4428	531	1590	41691	1502	409
N_o	8842	1843	22155	160410	1398	3900
N_n	7377	1213	21253	94632	503	2719
o	56	63	52	66	65	64
m	0.80	0.79	0.82	0.53	0.52	0.52

Table 5: Difficulty case splits across datasets (train, dev and test combined). o=obvious, m=median JSD.

Pairs are categorised into high and low lexical divergence categories by comparing their JSD score to the median of the entire JSD distribution in order to account for differences between datasets (>median: high div, ≤median: low div). To verify if this automatic difficulty distinction corresponds with real-world difficulty, the authors of the study annotated the semantic relatedness of 100 random pairs from the Quora development set and measured inter-annotator agreement based on Fleiss' Kappa. The agreement for non-obvious cases (P_n and N_n) is significantly lower (p-value<0.01 with permutation test) than for obvious cases (P_o and N_o) and the average annotation time per item is longer for non-obvious cases (Table 4), confirming the validity of this distinction.

Table 5 shows the number of instances in the four cases across datasets. In all of the analysed datasets, there are more obvious positives (P_o) than non-obvious positives (P_n) and more obvious negatives (N_o) than non-obvious negatives (N_n). All obvious cases combined (P_o+N_o) make up more than 50% of pairs across all datasets.

5 Evaluating model predictions based on difficulty

We now use this categorisation for the purpose of model evaluation (Tables 6-8).[3] We calculate the

[3]Due to the lack of openly available model prediction files, we only present our analysis for the Se-

	KeLP	Beihang MSRA	IIT UHH	ECNU	bunji	EICA	Swiss Alps	FuRong Wang	FA3L	Snow Man	ran-dom
TPR_o	0.652	**1.000**	0.800	0.790	0.681	0.328	0.333	0.562	0.691	0.677	0.501
TPR_n	0.496	**1.000**	0.676	0.636	0.575	0.269	0.223	0.399	0.478	0.469	0.499
TNR_o	0.909	0.000	0.731	0.877	0.894	0.959	**0.984**	0.913	0.787	0.900	0.515
TNR_n	0.908	0.000	0.676	0.820	0.851	**0.953**	0.950	0.892	0.751	0.757	0.536
$F1_o$	0.751	0.682	0.781	**0.829**	0.765	0.480	0.494	0.684	0.731	0.765	0.513
$F1_n$	0.628	0.686	0.686	**0.707**	0.672	0.410	0.352	0.533	0.560	0.555	0.519
F1	0.698	0.684	0.739	**0.777**	0.725	0.450	0.433	0.621	0.659	0.673	0.516
MAP	**0.884**	0.882	0.869	0.867	0.866	0.865	0.862	0.843	0.834	0.818	0.623

Table 6: Proposed evaluation metrics for top 10 primary submissions on SemEval Task A. The systems are ordered in columns according to their MAP ranking. Bold indicates the highest value for each metric. We indicate the 2^{nd} and 3^{rd} systems based on $F1_n$ and F1.

	Sim Bow	LearningTo Question	KeLP	Talla	Beihang MSRA	NLM NIH	Uin-suska TiTech	IIT UHH	SCIR QA	FA3L	ran-dom
TPR_o	0.976	**1.000**	0.920	0.760	**1.000**	0.880	0.752	0.704	0.912	0.448	0.552
TPR_n	0.842	**1.000**	0.632	0.763	**1.000**	0.500	0.421	0.737	0.842	0.263	0.395
TNR_o	0.609	0.000	0.831	0.684	0.000	0.841	0.858	0.682	0.709	**0.861**	0.495
TNR_n	0.197	0.000	0.432	0.467	0.000	0.397	0.552	0.403	0.352	**0.756**	0.521
$F1_o$	0.604	0.383	**0.746**	0.548	0.383	0.736	0.681	0.516	0.641	0.473	0.348
$F1_n$	0.198	0.195	0.199	**0.247**	0.195	0.154	0.164	0.221	0.234	0.160	0.147
F1	0.424	0.312	**0.506**	0.426	0.312	0.473	0.467	0.390	0.464	0.365	0.280
MAP	**0.472**	0.469	0.467	0.457	0.448	0.446	0.434	0.431	0.427	0.422	0.298

Table 7: Proposed evaluation metrics for top 10 primary submissions on SemEval Task B.

true positive rate TPR (for P_o and P_n) and true negative rate TNR (for N_o and N_n) to analyse model performance within each difficulty category. In the three SemEval 2017 CQA tasks, all systems perform worse on the hard cases compared to the obvious cases ($TPR_n < TPR_o$ and $TNR_n < TNR_o$), while there are only minor changes in the random baseline which predicts all classes with equal probability. To compare how well models do on obvious vs. non-obvious cases overall, we compute F1 scores for obvious cases (P_o and N_o) as $F1_o$ and non-obvious cases (P_n and N_n) as $F1_n$ separately. This is necessary as the high percentage of obvious cases (observed in section 4) can inflate the overall F1 score. $F1_n$ scores are consistently lower than the $F1_o$ scores. This difference is especially pronounced in Task B, which contained the highest proportion of obvious cases (62%) of the SemEval tasks. Using the non-obvious F1 scores results in a different ranking compared to the official SemEval evaluation metrics (F1 or MAP), even resulting in a change in the highest ranked system in Task B (Talla instead of KeLP or SimBow) and C (KeLP instead of bunji or IIT-UHH).

	IIT UHH	bunji	KeLP	EICA	ran-dom
TPR_o	0.570	0.246	**0.911**	0.006	0.520
TPR_n	0.358	0.045	**0.836**	0.000	0.433
TNR_o	0.898	0.991	0.720	**0.998**	0.502
TNR_n	0.779	0.965	0.538	**0.999**	0.502
$F1_o$	0.283	**0.339**	0.209	0.011	0.076
$F1_n$	0.047	0.028	**0.054**	0.000	0.027
F1	0.144	**0.197**	0.121	0.008	0.053
MAP	**0.155**	0.147	0.144	0.135	0.058

Table 8: Proposed evaluation metrics for top 4 primary submissions on SemEval Task C.

mEval CQA Tasks based on prediction files obtained from http://alt.qcri.org/semeval2017/task3/index.php?id=results.

6 Conclusion

We present an automated criterion for automatically distinguishing between easy and difficult items in text pair similarity prediction tasks. We find that more than 50% of cases in current datasets are relatively obvious. Recently proposed models perform significantly worse on non-obvious cases compared to obvious cases. In or-

der to encourage the development of models that perform well on difficult items, we propose to use non-obvious F1 scores ($F1_n$) as a complementary ranking metric for model evaluation. We also recommend publishing prediction files along with models to facilitate error analysis.

Acknowledgments

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivý. 2016. Sentence Pair Scoring: Towards Unified Framework for Text Comprehension. *arXiv preprint arXiv:1603.06127*.

Daniele Bonadiman, Antonio Uva, and Alessandro Moschitti. 2017. Effective Shared Representations with Multitask Learning for Community Question Answering. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 726–732, Valencia, Spain. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP@IJCNLP)*, pages 9–16, Jeju Island, Korea. Asian Federation of Natural Language Processing.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1576–1586, Lisbon, Portugal. Association for Computational Linguistics.

Divyansh Kaushik and Zachary C Lipton. 2018. How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, Brussels, Belgium. Association for Computational Linguistics.

Jianhua Lin. 1991. Divergence Measures based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Preslav Nakov, Doris Hoogeveen, Llúis Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 27–48, Vancouver, Canada. Association for Computational Linguistics.

Preslav Nakov, Lluis Marquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, James Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2016)*, pages 525–545, San Diego, California. Association for Computational Linguistics.

Preslav Nakov, Lluis Marquez, Magdy Walid, Alessandro Moschitti, James Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015)*, pages 269–281, Denver, Colorado. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Joao Rodrigues, Chakaveh Saedi, Antonio Branco, and Joao Silva. 2018. Semantic Equivalence Detection: Are Interrogatives Harder than Declaratives? In *Proceedings of the 11th International Conference on Language Resources and Evaluation*, pages 3248–3253, Miyazaki, Japan. European Language Resources Association.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 801–809, Granada, Spain.

B Better Early than Late - Fusing Topics with Word Embeddings for Neural Question Paraphrase Identification

Better Early than Late: Fusing Topics with Word Embeddings for Neural Question Paraphrase Identification

Nicole Peinelt[1,2] and **Dong Nguyen**[1,3] and **Maria Liakata**[1,2]
[1]The Alan Turing Institute, London, UK
[2]University of Warwick, Coventry, UK
[3]Utrecht University, Utrecht, The Netherlands
{[redacted]}

Abstract

Question paraphrase identification is a key task in Community Question Answering (CQA) to determine if an incoming question has been previously asked. Many current models use word embeddings to identify duplicate questions, but the use of topic models in feature-engineered systems suggests that they can be helpful for this task, too. We therefore propose two ways of merging topics with word embeddings (early vs. late fusion) in a new neural architecture for question paraphrase identification. Our results show that our system outperforms neural baselines on multiple CQA datasets, while an ablation study highlights the importance of topics and especially *early* topic-embedding fusion in our architecture.

1 Introduction

Paraphrase identification is a core NLP task and has been widely studied (Socher et al., 2011; He et al., 2015; Wieting et al., 2016; Tomar et al., 2017). One interesting application area of paraphrase detection is Community Question Answering (CQA) (Nakov et al., 2017; Bonadiman et al., 2017; Rodrigues et al., 2018). The aim of CQA is to answer real open-ended questions based on user-generated content from question answering websites. Being able to identify similar — already answered — questions can be helpful for this purpose. Question paraphrase detection in CQA is difficult because texts tend to be longer and have less direct overlap compared to traditional paraphrase detection datasets (Rus et al., 2014; Peinelt et al., 2019).

Early work on paraphrase detection relied on hand-crafted features, while state-of-the-art approaches for paraphrase identification are primarily neural networks (Gong et al., 2018; Wang et al., 2017; Tomar et al., 2017) and hybrid techniques (Pang et al., 2016; Wu et al., 2017; Feng et al., 2017). Many recently proposed CQA paraphrase detection systems still use hand-crafted features (Agustian and Takamura, 2017; Filice et al., 2017) and some work has successfully integrated topic model features (Duan et al., 2008; Wu et al., 2017). This suggests that topic distributions could offer auxiliary information for identifying related questions and complement word embeddings (Mikolov et al., 2013; Pennington et al., 2014), which provide the main signal in neural systems. Contrary to hand-crafted static topic features, integrating topics in a neural framework brings the advantage of joint updates during training. Recent work successfully introduced topics in neural architectures for language generation: Wang et al. (2018) used a topic-enhanced encoder for summarisation, Chen et al. (2016) integrated topics in the decoder for machine translation and Narayan et al. (2018) included topics in both encoder and decoder of their summarisation model.

However, it remains unclear if topics can be useful in a neural paraphrase detection model and how to best fuse topics with word embeddings for this task. In this paper, we introduce a novel topic-aware neural architecture and specifically make the following contributions:

1. We define two settings (early and late fusion) for incorporating topics in our neural paraphrase prediction model (section 3).

2. Our topic-aware model improves over other neural models across multiple question paraphrase identification datasets (section 4).

3. In an ablation study, we highlight the importance of topics and early topic-embedding fusion in our proposed architecture (section 4).

2 Datasets and Tasks

We address the problem of CQA question paraphrase detection, where given two questions from

	Task	Source	Question pairs
Quora	PD	real-world	404 345
PAWS	PD	synthetic	12 665
SemEval	PR	real-world	4 749

Table 1: Selected CQA question paraphrase datasets. PD=paraphrase detection, PR=paraphrase ranking

question answering websites, denoted as q_1 and q_2 with length n and m, the task is to predict a binary label which indicates whether the two questions are paraphrases. For this study, we select three popular English question paraphrase identification datasets and summarise their main properties in Table 1.

The **Quora** duplicate questions dataset consists of over 400 000 question pairs, each annotated with a binary label indicating whether the two questions are paraphrases of each other or not.[1] We use the same split as mentioned in Wang et al. (2017).

PAWS (Zhang et al., 2019) is a synthetic paraphrase detection dataset which was created on the basis of Quora. The word order of Quora examples was automatically altered and words replaced, resulting in question pairs with high word overlap which were manually annotated with binary paraphrase labels.

The **SemEval** 2017 Task 3 dataset focuses on Community Question Answering (Nakov et al., 2017). While there are other subtasks, we only use subtask B (question paraphrase detection) here. In each example, the dataset provides a new question and a set of ten possibly related questions which were retrieved from the forum of the website *Qatar Living*[2] by a search engine. For each question pair, a relevancy label is provided.

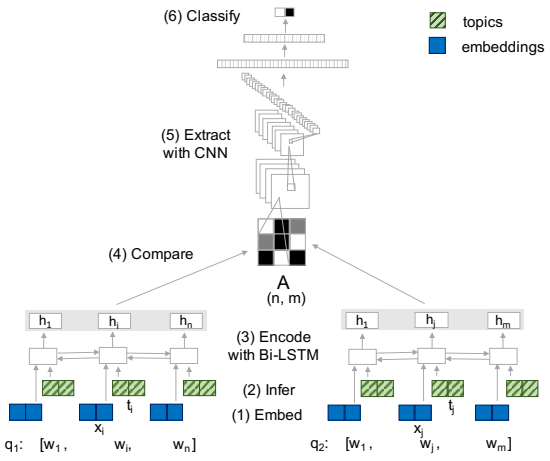


Figure 1: TAPA with early topic-embedding fusion. Illustrated with toy examples consisting of three tokens.

[1]https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning

3 Model Architecture

This paper examines whether we can successfully integrate topic model information in a neural paraphrase identification model. We also explore how to best combine topics with word embeddings, which are the main information source in existing models (Deriu and Cieliebak, 2017; Pang et al., 2016; Gong et al., 2018). For this purpose, we propose a novel architecture dubbed **T**opic-**A**ware **P**araphrase Identification **A**rchitecture (TAPA) depicted in Figures 1 and 2. Our model comprises the following steps: Obtaining a representation for both questions (Section 3.1), comparing these representations (Section 3.2), and aggregating the information for the final prediction (Section 3.2).

3.1 Encoding layer

Embeddings Given the sequence of words $(w_1, ..., w_n)$ in a question of length n, we map them to embeddings $x = (x_1, ..., x_n)$ (step 1 in Figures 1 and 2). For each embedding x_i, we combine pretrained word and ELMo embeddings (Peters et al., 2018) to leverage alternative representations:

$$x_i = [\text{emb}_i; \text{ELMo}_i] \in R^f \quad (1)$$

where ; denotes concatenation and f the resulting embedding dimension.

Topics We infer topic distributions from an LDA topic model (Blei et al., 2003) with f' number of topics for the whole question $t_D \in R^{f'}$ and every

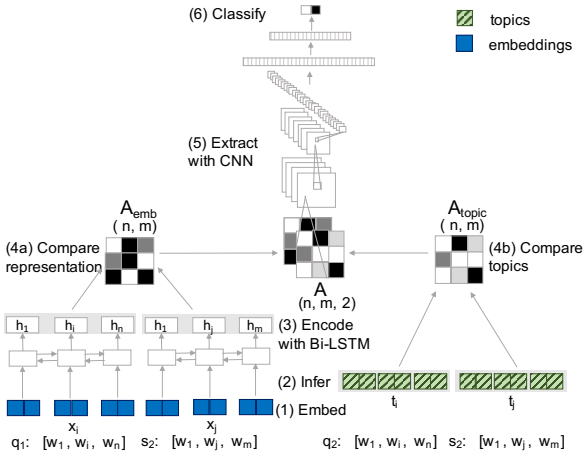


Figure 2: TAPA with late topic-embedding fusion

[2]https://www.qatarliving.com/

word in the question $t' = (t'_1, ..., t'_n)$ with $t_i \in R^{f'}$ (step 2 in Figures 1 and 2). We follow Narayan et al. (2018) in merging word-level and document-level topics to capture the interaction between both as

$$t_i = [t'_i \otimes t_D] \in R^{f'} \quad (2)$$

where $\otimes$ denotes element-wise multiplication as 'word+doc' setting. Previous work by Narayan et al. (2018) focused on a different task (summarisation) with longer texts. In contrast, for our setting we expect word-level topics to be more informative than document topics. Therefore, we further include only word-level topics as 'word' setting

$$t_i = [t'_i] \in R^{f'} \quad (3)$$

where the topic setting is treated as hyperparameter.

Fusion of embeddings and topics We propose two different ways of combining topic distributions with word embeddings: early fusion and late fusion. Early fusion combines topic distributions with word embeddings before the encoder step (Figure 1), guiding the encoder in selecting relevant information when computing a representation for the sentence. In contrast, late fusion combines information derived from topics and word embeddings *after* computing separate affinity matrices for topics and word representations (Figure 2), therefore introducing the topic information more directly into the architecture as separate sentence interaction dimension. As a result, the encoding layer of both variations differs slightly. In early fusion TAPA, we obtain a sentence representation $\mathbf{e} = (e_1, ..., e_n)$ by concatenating topics and embeddings as follows:

$$e_i = [x_i; t_i] \in R^{f+f'} \quad (4)$$

Late fusion TAPA only uses embeddings for $\mathbf{e}$

$$e_i = [x_i] \in R^{f} \quad (5)$$

and obtains separate topic representations

$$\text{TL} = [t_1, ..., t_n] \in R^{(n \times f')} \quad (6)$$

$$\text{TR} = [t_1, ..., t_m] \in R^{(m \times f')} \quad (7)$$

Encoder In our preliminary experiments, BiLSTMs (Hochreiter and Schmidhuber, 1997) worked better than CNN encoders, presumably due to their ability to capture long-range dependencies. As a result, we encode $\mathbf{e}$ with BiLSTMs (step 3 in Figures 1 and 2):

$$\text{L} = \text{BiLSTM}(\mathbf{e}) \in R^{(n \times d)} \quad (8)$$

$$\text{R} = \text{BiLSTM}(\mathbf{e}) \in R^{(m \times d)} \quad (9)$$

We decide to share weights between both BiLSTMs in a Siamese setting which reduces the number of required parameters and has been shown to work well for pairwise classification tasks (Deriu and Cieliebak, 2017; Feng et al., 2015; Mueller and Thyagarajan, 2016).

3.2 Comparison layer

We model the similarity between the two encoded questions by computing pairwise affinity scores between their words in affinity matrices similar to previous studies (Deriu and Cieliebak, 2017; Pang et al., 2016; Gong et al., 2018). In both early and late fusion TAPA, we calculate the affinity matrix A_{emb} based on the encoded sentences as

$$A_{emb_{i,j}} = s(\text{L}_{i,:}, \text{R}_{j,:}) \in \mathbb{R}^{n \times m} \quad (10)$$

using a similarity function s (step 4 in Figure 1 and 4a in Figure 2). For the late fusion model, we also calculate an affinity matrix based on the topic distributions in both sentences (4b in Figure 2):[3]

$$A_{topic_{i,j}} = s(\text{TL}_{i,:}, \text{TR}_{j,:}) \in \mathbb{R}^{n \times m} \quad (11)$$

Common choices for the similarity function s are Euclidean distance, dot product and cosine similarity (Yin et al., 2016; Deriu and Cieliebak, 2017). We use cosine similarity, as it worked best in preliminary experiments. The comparison layer output A is simply A_{emb} for early fusion TAPA, while we combine topic and embedding affinity matrices in the late fusion version:

$$A = [A_{emb}; A_{topic}]$$

3.3 Aggregation layer

Similar to extracting information from a grey scale image in computer vision, we follow Gong et al. (2018) and Pang et al. (2016) in aggregating useful affinity patterns from A with a CNN (LeCun et al., 1998) feature extractor (step 5 in in Figure 1 and 2). We use a two-layer CNN architecture (where one layer consists of convolution and pooling). The output of the last convolution layer is flattened into a vector. This is followed by multiple hidden layers of reducing size and a softmax layer for predicting the two classes (step 6 in Figure 1 and 2). The model is trained based on cross-entropy loss. We tune hyperparameters with hyperopt (Bergstra et al., 2013) and report them in Appendix B. For further implementation details refer to Appendix C.

[3] We experimented with additional topic encoders, but abandoned them as they didn't consistently improve results.

4 Results

We evaluate model performance on the basis of F1 scores as this is more reliable for datasets with label imbalance than accuracy and present the results in Table 2.

Baselines As baseline systems, we provide the three best performing **SemEval 2017 models**: KeLP (Filice et al., 2017), NLM-NIH (Abacha and Demner-Fushman, 2017) and Uinsuska TiTech (Agustian and Takamura, 2017). All of these models employ hand-crafted features, which provides an advantage on this small dataset (compare Table 1 for dataset sizes). Results have been reported for systems on PAWS and Quora with accuracies ranging between 75 and 89, but are not directly comparable to F1 scores (Gong et al., 2018; Tan et al., 2018; Tomar et al., 2017).

In addition to the above mentioned systems, we also compare our proposed model with a **Siamese network**, as this is a common neural baseline for paraphrase identification (Wang et al., 2017). The two questions are embedded with pretrained 300 dimensional Glove embeddings (Pennington et al., 2014) and encoded by two weight-sharing BiLSTM encoders. This is followed by a max pooling layer and two hidden layers. For Siamese network +ELMo, we concatenate word vectors with ELMo representations before the encoding step.

TAPA vs. baselines TAPA performs better than the neural baselines. It cannot compete with the highest ranked feature engineered SemEval system (KeLP) due to the lack of sufficient training data (compare Table 1), but gets within reach of the third placed system (Uinsuska TiTech). Our neural architecture may not outperform the top three systems on the SemEval dataset, but it can generalise better across datasets, while the three SemEval systems require dataset specific feature engineering.

	Type	PAWS	Quora	SemEval
KeLP	feat.	-	-	**50.6**
NLM-NIH	feat.	-	-	47.3
Uinsuska TiTech	feat.	-	-	46.7
Siamese network	neural	17.3	81.3	34.9
+ELMo	neural	37.2	83.2	34.5
TAPA	neural	**42.2**	**84.1**	46.4

Table 2: F1 scores of models on test sets. The first three rows are taken from Nakov et al. (2017), the rest are our own implementations. feat=feature-based

	PAWS	Quora	SemEval
full TAPA (early fusion)	42.2	84.1	46.4
-topics	40.6	83.9	45.1
-ELMO	26.9	84.5	45.0
TAPA with late fusion	39.8	83.9	40.1

Table 3: Ablation study for our TAPA model reporting F1 scores on test sets.

Influence of model components We conduct an ablation study to understand the contribution of individual model components (Table 3). Removing topics consistently reduces F1 scores on all datasets, while the effect of ELMo representations is dataset dependent. Deleting ELMo improves performance on Quora, but leads to a massive performance drop on PAWS. The large impact on PAWS can be explained by the fact that this dataset was automatically constructed to have high textual overlap between questions and differences between paraphrases are chiefly due to variations in syntax. Our full TAPA model uses early fusion as this was the best setting during hyperparameter tuning. When comparing the full (early fusion) model with a tuned[4] late fusion variant, we find that performance of the late fusion model drops to the same or even lower level than TAPA without topics. We conclude that topics contribute consistently to the performance of our proposed model, but that early topic-embedding fusion is crucial.

5 Conclusion

In this work, we introduced a novel topic-aware neural architecture for question paraphrase identification. Our model successfully fuses word embeddings with topics and improves over previous neural baselines on multiple CQA paraphrase identification datasets. We demonstrated that topics contributed consistently to the performance of our model and that an early fusion of word embeddings with topic distributions is preferable over integration at a later stage. Our work suggests that early fusion of topics with models which were pretrained with sentence pair classification tasks, such as BERT (Devlin et al., 2019) could be a promising direction for future research. Other future work could seek to enhance our proposed architecture with more sophisticated topic models.

[4] As late fusion may require slightly different hyperparameters, we compare with a tuned late fusion model to make a fair comparison between early and late fusion.

References

Asma Ben Abacha and Dina Demner-Fushman. 2017. NLM NIH at SemEval-2017 Task 3: From Question Entailment to Question Similarity for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 349–352, Vancouver, Canada. Association for Computational Linguistics.

Surya Agustian and Hiroya Takamura. 2017. UINSUSKA-TiTech at SemEval-2017 Task 3: Exploiting Word Importance Levels for Similarity Features for CQA. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 370–374, Vancouver, Canada. Association for Computational Linguistics.

James Bergstra, Dan Yamins, and David D. Cox. 2013. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning research*, 3:993–1022.

Daniele Bonadiman, Antonio Uva, and Alessandro Moschitti. 2017. Effective Shared Representations with Multitask Learning for Community Question Answering. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 726–732, Valencia, Spain. Association for Computational Linguistics.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided Alignment Training for Topic-Aware Neural Machine Translation. In *Proceedings of AMTA*, pages 121–134, Austin, USA.

Jan Milan Deriu and Mark Cieliebak. 2017. SwissAlps at SemEval-2017 task 3: Attention-based Convolutional Neural Network for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation. Vancouver, Canada, SemEval*, volume 17, pages 334–338. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 4171—4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching Questions by Identifying Question Topic and Question Focus. In *Proceedings of ACL-08: HLT*, page 9, Columbus, USA.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820, Scottsdale, USA. IEEE.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2017. Beihang-MSRA at SemEval-2017 Task 3- A Ranking System with Neural Matching Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 280–286, Vancouver, Canada. Association for Computational Linguistics.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 Task 3-Learning Pairwise Patterns in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 326–333, Vancouver, Canada. Association for Computational Linguistics.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural Language Inference over Interaction Space. In *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1576–1586, Lisbon, Portugal. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations (ICLR 2013)*, Scottsdale, USA.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 2786–2792, Phoenix, USA. AAAI Press.

Preslav Nakov, Doris Hoogeveen, Llúis Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 27–48, Vancouver, Canada. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 2793–2799, Phoenix, USA. AAAI Press.

Nicole Peinelt, Maria Liakata, and Dong Nguyen. 2019. Aiming beyond the Obvious: Identifying Non-Obvious Cases in Semantic Similarity Datasets. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2792–2798, Florence, Italy. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, New Orleans, USA. Association for Computational Linguistics.

Radim Rehurek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 46–50, Valletta, Malta. European Language Resources Association.

Joao Rodrigues, Chakaveh Saedi, Antonio Branco, and Joao Silva. 2018. Semantic Equivalence Detection: Are Interrogatives Harder than Declaratives? In *Proceedings of the 11th International Conference on Language Resources and Evaluation*, pages 3248–3253, Miyazaki, Japan. European Language Resources Association.

Vasile Rus, Rajendra Banjade, and Mihai Lintean. 2014. On Paraphrase Identification Corpora. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 14)*, pages 2422–2429, Reykjavik, Iceland. European Language Resources Association (ELRA).

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 801–809, Granada, Spain.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway Attention Networks for Modeling Sentence Pairs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4411–4417, Stockholm, Sweden.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural Paraphrase Identification of Questions with Noisy Pretraining. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 142–147, Copenhagen, Denmark. Association for Computational Linguistics.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A Reinforced Topic-Aware Convolutional Sequence-to-Sequence Model for Abstractive Text Summarization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4453–4460, Stockholm, Sweden.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4144–4150, Melbourne, Australia.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *4th International Conference on Learning Representations (ICLR 2016)*, page 19, San Juan, Puerto Rico.

Guoshun Wu, Yixuan Sheng, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 3- Using Traditional and Deep Learning Methods to Address Community Question Answering Task. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 356–360, Vancouver, Canada. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, USA. Association for Computational Linguistics.

C tBERT - Topic Models and BERT Joining Forces for Semantic Similarity Detection

tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection

Nicole Peinelt[1,2] and **Dong Nguyen**[1,3] and **Maria Liakata**[1,2]
[1]The Alan Turing Institute, London, UK
[2]University of Warwick, Coventry, UK
[3]Utrecht University, Utrecht, The Netherlands
{[redacted]

Abstract

Semantic similarity detection is a fundamental task in natural language understanding. Adding topic information has been useful for previous feature-engineered semantic similarity models as well as neural models for other tasks. There is currently no standard way of combining topics with pretrained contextual representations such as BERT. We propose a novel topic-informed BERT-based architecture for pairwise semantic similarity detection and show that our model improves performance over strong neural baselines across a variety of English language datasets. We find that the addition of topics to BERT helps particularly with resolving domain-specific cases.

1 Introduction

Modelling the semantic similarity between a pair of texts is a crucial NLP task with applications ranging from question answering to plagiarism detection. A variety of models have been proposed for this problem, including traditional feature-engineered techniques (Filice et al., 2017), hybrid approaches (Wu et al., 2017; Feng et al., 2017; Koreeda et al., 2017) and purely neural architectures (Wang et al., 2017; Tan et al., 2018; Deriu and Cieliebak, 2017). Recent pretrained contextualised representations such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have led to impressive performance gains across a variety of NLP tasks, including semantic similarity detection. These models leverage large amounts of data to pretrain text encoders (in contrast to just individual word embeddings as in previous work) and have established a new pretrain-finetune paradigm.

While large improvements have been achieved on paraphrase detection (Tomar et al., 2017; Gong et al., 2018), semantic similarity detection in Community Question Answering (CQA) remains a challenging problem. CQA leverages user-generated content from question answering websites (e.g. StackExchange) to answer complex real-world questions (Nakov et al., 2017). The task requires modelling the relatedness between question-answer pairs which can be challenging due to the highly domain-specific language of certain online forums and low levels of direct text overlap between questions and answers.

Topic models may provide additional signals for semantic similarity, as earlier feature-engineered models for semantic similarity detection successfully incorporated topics (Qin et al., 2009; Tran et al., 2015; Mihaylov and Nakov, 2016; Wu et al., 2017). They could be especially useful for dealing with domain-specific language since topic models have been exploited for domain adaptation (Hu et al., 2014; Guo et al., 2009). Moreover, recent work on neural architectures has shown that the integration of topics can yield improvements in other tasks such as language modelling (Ghosh et al., 2016), machine translation (Chen et al., 2016), and summarisation (Narayan et al., 2018; Wang et al., 2018). We therefore introduce a novel architecture for semantic similarity detection which incorporates topic models and BERT. More specifically, we make the following contributions:

1. We propose tBERT — a simple architecture combining topics with BERT for semantic similarity prediction (section 3).[1]

2. We demonstrate that tBERT achieves improvements across multiple semantic similarity prediction datasets against a finetuned vanilla BERT and other neural models in both F1 and stricter evaluation metrics (section 5).

3. We show in our error analysis that tBERT's gains are prominent on domain-specific cases, such as those encountered in CQA (section 5).

[1]Code is available at https://github.com/wuningxi/tBERT.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7047–7055

2 Datasets and Tasks

We select popular benchmark datasets featuring different sizes (small vs. large), tasks (QA vs. paraphrase detection) and sentence lengths (short vs. long) as summarised in Table 1. Examples for each dataset are provided in Appendix A.

MSRP The Microsoft Research Paraphrase dataset (MSRP) contains pairs of sentences from news websites with binary labels for paraphrase detection (Dolan and Brockett, 2005).

SemEval The SemEval CQA dataset (Nakov et al., 2015, 2016, 2017) comprises three subtasks based on threads and posts from the online expat forum *Qatar Living*.[2] Each subtask contains an initial post as well as 10 possibly relevant posts with binary labels and requires to rank relevant posts above non-relevant ones. In subtask A, the posts are questions and comments from the same thread, in an answer ranking scenario. Subtask B is question paraphrase ranking. Subtask C is similar to A but comments were retrieved from an external thread, which increases the difficulty of the task.

Quora The Quora duplicate questions dataset contains more than 400k question pairs with binary labels and is by far the largest of the datasets.[3] The task is to predict whether two questions are paraphrases. The setup is similar to SemEval subtask B, but framed as a classification rather than a ranking problem. We use Wang et al. (2017)'s train/dev/test set partition.

All of the above datasets provide two short texts (usually a sentence long but in some cases consisting of multiple sentences). From here onward we will use the term 'sentence' to refer to each short text. We frame the task as predicting the semantic similarity between two sentences in a binary classification task. We use a binary classification setup as this is more generic and applies to all above datasets.

Dataset	Task	Len	Size
Quora	paraphrase detection	13	404K
MSRP	paraphrase detection	22	5K
SemEval	(A) internal answer ranking	48	26K
	(B) paraphrase ranking	52	4K
	(C) external answer ranking	45	47K

Table 1: Text pair similarity data sets. Size = number of text pairs. Len = mean sentence length in tokens.

[2]Following convention, we use the 2016 test set as development set and 2017 test set as test set.

[3]https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning

3 tBERT

3.1 Architecture

In this paper, we investigate if topic models can further improve BERT's performance for semantic similarity detection. Our proposed **t**opic-informed **BERT**-based model (tBERT) is shown in Figure 1. We encode two sentences S_1 (with length N) and S_2 (with length M) with the uncased version of $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019), using the C vector from BERT's final layer corresponding to the *CLS* token in the input as sentence pair representation:

$$C = \text{BERT}(S_1, S_2) \in R^d \quad (1)$$

where d denotes the internal hidden size of BERT (768 for $\text{BERT}_{\text{BASE}}$). While other topic models can be used, we experiment with two popular topic models: LDA (Blei et al., 2003) and GSDMM (Yin and Wang, 2014), see section 3.2 for details. Based on previous research which successfully combined word and document level topics with neural architectures (Narayan et al., 2018), we further experiment with incorporating different topic types. For document topics D_1 and D_2, all tokens in a sentence are passed to the topic model to infer one topic distribution per sentence:

$$D_1 = \text{TopicModel}([T_1, ..., T_N]) \in R^t \quad (2)$$
$$D_2 = \text{TopicModel}([T'_1, ..., T'_M]) \in R^t \quad (3)$$

where t indicates the number of topics. Alternatively, for word topics W_1 and W_2, one topic distri-

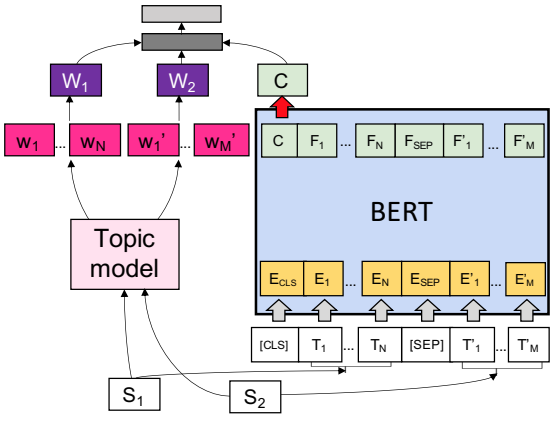


Figure 1: Architecture of tBERT with word topics.

bution w_i is inferred per token T_i

$$w_i = \text{TopicModel}(T_i) \in R^t \quad (4)$$

before averaging them to obtain a fixed-length topic representation on the sentence level:

$$W_1 = \frac{\sum_{i=1}^{N} w_i}{N} \in R^t \quad (5)$$

$$W_2 = \frac{\sum_{i=1}^{M} w'_i}{M} \in R^t \quad (6)$$

We combine the sentence pair vector with the sentence-level topic representations similar to Ostendorff et al. (2019) as

$$\text{F} = [C; D_1; D_2] \in R^{d+2t} \quad (7)$$

for document topics and as

$$\text{F} = [C; W_1; W_2] \in R^{d+2t} \quad (8)$$

for word topics (where ; denotes concatenation). This is followed by a hidden and a softmax classification layer. We train the model for 3 epochs with early stopping and cross-entropy loss. Learning rates are tuned per dataset and random seed.[4]

3.2 Choice of Topic Model

Topic number and alpha value The number of topics and alpha values are important topic model hyper-parameters and dataset dependent. We use the simple topic baseline (section 4) as a fast proxy (on average 12 seconds per experiment on CPU) to identify useful topic models for each dataset without expensive hyper-parameter tuning on the full tBERT model. In our experiments, 70 to 90 topics with alpha values of 1 or 10 worked well.[5]

	MSRP	Quora	SemEval		
			A	B	C
BERT	.906	.906	.714	.754	.414
tBERT with LDA					
+ word topics	.905	**.911**	.744	**.766**	**.439**
+ doc topics	**.907**	.909	**.748**	.761	.419
tBERT with GSDMM					
+ word topics	**.918**	.908	**.752**	**.760**	**.447**
+ doc topics	.915	**.909**	.751	.760	.424

Table 2: F1 scores of BERT-based models with different topic settings on development set. We report average performance for two different random seeds. Bold indicates the selected setting for our final model.

[4] We report tuned hyper-parameters in Appendix E.

[5] See Appendix D for detailed topic model settings.

Topic model and topic type LDA (Blei et al., 2003) is the most popular and widely used topic model, but it has been reported to be less suitable for short text (Hong and Davison, 2010). Therefore, we also experiment with the popular short text topic model GSDMM (Yin and Wang, 2014). To select the best setting for our final model (in Table 3), we evaluated different combinations of tBERT with LDA vs. GSDMM and word (W_1 and W_2) vs. document topics (D_1 and D_2) on the development partition of the datasets (Table 2). The tBERT settings generally scored higher than BERT, with word topics (W_1 and W_2) usually outperforming document topics.

4 Baselines

Topic baselines As a simple baseline, we train a topic model (LDA or GSDMM) on the training portion of each dataset (combining training sets for SemEval subtasks) and calculate the Jensen-Shannon divergence (Lin, 1991) (JSD) between the topic distributions of the two sentences. The model predicts a negative label if JSD is larger than a threshold and a positive label otherwise. We tune threshold, number of topics and alpha value based on development set F1.[5]

Previous systems For SemEval, we compare against the highest performing system of earlier work based on F1 score. As these models rely on hand-crafted dataset-specific features (providing an advantage on the small datasets), we also include the only neural system without manual features (Deriu and Cieliebak, 2017). For MSRP, we show a neural matching architecture (Pang et al., 2016). For Quora, we compare against the Interactive Inference Network (Gong et al., 2018) using accuracy, as no F1 has been reported.

Siamese BiLSTM Siamese networks are a common neural baseline for sentence pair classification tasks (Yih et al., 2011; Wang et al., 2017). We embed both sentences with pretrained GloVe embeddings (concatenated with ELMo for BiLSTM + ELMo) and encode them with two weight-sharing BiLSTMs, followed by max pooling and hidden layers.

BERT We encode the sentence pair with BERT's C vector (as in tBERT) followed by a softmax layer and finetune all layers for 3 epochs with early stopping. Following Devlin et al. (2019), we tune learning rates on the development set of each dataset.[4]

5 Results

Evaluation We evaluate systems based on F1 scores (Table 3) as this is more reliable for datasets with imbalanced labels (e.g. SemEval C) than accuracy. We further report performance on difficult cases with non-obvious F1 score (Peinelt et al., 2019) which identifies challenging instances in the dataset based on lexical overlap and gold labels. Dodge et al. (2020) recently showed that early stopping and random seeds can have considerable impact on the performance of finetuned BERT models. We therefore use early stopping during finetuning and report average model performance across two seeds for BERT and tBERT models.

Overall trends The BERT-based models outperform the other neural systems, while closely competing with the feature-engineered system on the relatively small SemEval A dataset. The simple topic baselines perform surprisingly well in comparison to much more sophisticated models, indicating the usefulness of topics for the tasks.

Do topics improve BERT's performance? Adding LDA topics to BERT consistently improves F1 performance across all datasets. Moreover, it improves performance on non-obvious cases over BERT on all datasets (except for Quora which contains many generic examples and few domain-specific cases, see Table 4). The addition of GSDMM topics to BERT is slightly less stable: improving performance on MSRP, Semeval A and B, while dropping on Semeval C. The largest performance gains regardless of the chosen topic model are observed in the internal question-answering task (SemEval A).

	MSRP	Quora	SemEval A	SemEval B	SemEval C
F1 on cases with **named entities** (total: 230/500)					
BERT	.20	**.54**	.50	**.53**	.32
tBERT	**.35**	.49	**.52**	.21	**.56**
(# of cases)	(23)	(31)	(58)	(60)	(58)
F1 on cases with **domain-specific words** (total: 159/500)					
BERT	.18	.00	.36	.36	.26
tBERT	**.67**	**.50**	**.62**	**.40**	**.58**
(# of cases)	(14)	(7)	(36)	(41)	(61)
F1 on cases with **non-standard spelling** (total: 53/500)					
BERT	.00	N/A	.20	**.71**	.43
tBERT	.00	N/A	**.80**	.00	**.62**
(# of cases)	(1)	(0)	(20)	(19)	(13)

Table 4: F1 for BERT and tBERT on annotated development set examples (100 cases per dataset) by manually annotated properties. Number of cases in parenthesis.

Where can topics help? We randomly sampled 100 examples (half only correct by BERT, half only correct by LDA-tBERT) from the development set of each dataset and manually annotated them (500 in total) with binary labels regarding three properties that may be associated with topic-related gains or losses (Table 4). Named entities (e.g. *iPhone*) and domain-specific words (e.g. *murabaha*) occurred frequently in the datasets, while there were too few examples with non-standard spelling (e.g. *thanx*) for meaningful comparisons. tBERT generally performed better than BERT on examples with domain-specific cases. Overall patterns were

	F1					non-obvious F1				
	MSRP	Quora	SemEval A	SemEval B	SemEval C	MSRP	Quora	SemEval A	SemEval B	SemEval C
Previous systems										
Filice et al. (2017) - feature-based	-	-	-	.506	-	-	-	-	.199	-
Wu et al. (2017) - feature-based	-	-	**.777**	-	-	-	-	.707	-	-
Koreeda et al. (2017) - feature-based	-	-	-	-	.197	-	-	-	-	.028
Deriu and Cieliebak (2017) - neural	-	-	.433	-	-	-	-	.352	-	-
Pang et al. (2016) - neural	.829	-	-	-	-	-	-	-	-	-
Gong et al. (2018) (accuracy) - neural	-	(.891)	-	-	-	-	-	-	-	-
Our implementation										
LDA topic baseline	.799	.736	.684	.436	.096	.780	.606	.684	.172	.019
GSDMM topic baseline	.796	.679	.663	.403	.102	.769	.448	.488	.130	.015
Siamese BiLSTM	.763	.813	.671	.349	.126	.781	.740	.597	.168	.049
Siamese BiLSTM + ELMo	.765	.832	.661	.345	.149	.775	.754	.599	.180	.073
BERT	.876	.902	.704	.473	.268	.827	**.860**	.656	.243	.085
tBERT with LDA topics	**.884**	**.905**	**.768**	**.524**	**.273**	**.866**	.859	.708	.258	**.100**
tBERT with GSDMM topics	.883	**.905**	.766	.518	.233	.844	.856	**.714**	**.266**	.081

Table 3: Model performance on test set. The first 6 rows are taken from the cited papers. Bold font highlights the best system overall and our best implementation is underlined. Italics indicate that F1 and accuracy were identical.

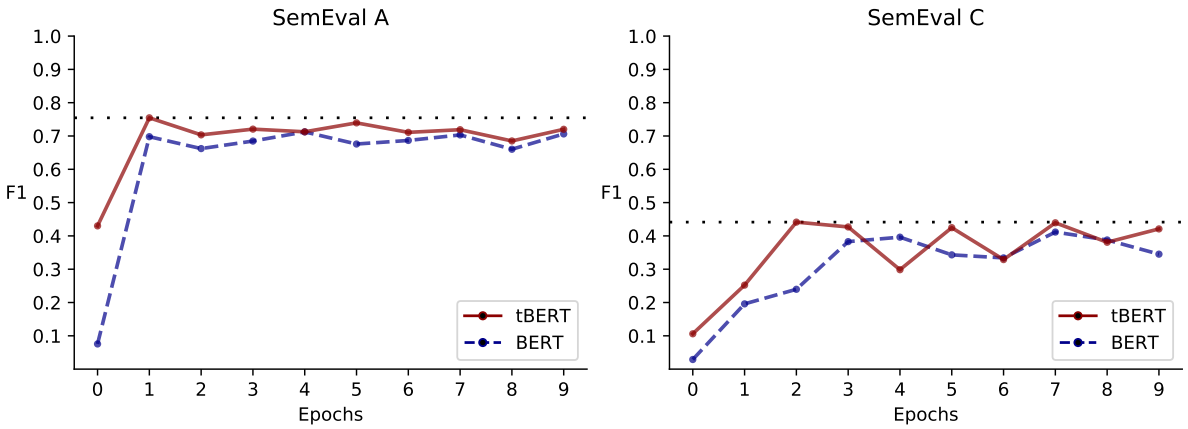


Figure 2: Performance of BERT and tBERT on dev set when trained for up to 9 epochs. The dotted line indicates tBERT's best performance within the first 3 epochs. Plots for the other datasets are provided as Appendix G.

less clear for named entities; based on manual inspection BERT dealt better with common named entities likely to have occurred in pretraining (such as well-known brands), while tBERT improved on dataset-specific named entities. We reason that for domain-specific words which are unlikely to have occurred in pretraining (e.g. *Fuwairit* in Table 5), BERT may not have learned a good representation (even after finetuning) and hence can't make a correct prediction. Here, topic models could serve as an additional source for dataset-specific information. The usefulness of topics for such cases is also supported by previous work, which successfully leveraged topics for domain adaptation in machine translation (Hu et al., 2014) and named entity recognition (Guo et al., 2009).

Could we just finetune BERT longer? Based on our observation that tBERT performs better on dataset-specific cases, one could assume that BERT may simply need to be finetuned longer than the usual 3 epochs to pick up more domain-specific information. In an additional experiment, we finetuned BERT and tBERT (with LDA topics) for 9 epochs (see Figure 2 and Appendix G). On most datasets, BERT reached peak performance within the first 3 epochs. Although training for 4 or 7 epochs achieved marginal gains on Semeval A and C, longer finetuning of BERT could not exceed tBERT's best performance from the first 3 epochs (dotted line) on any dataset. We conclude that longer finetuning does not considerably boost BERT's performance. Adding topics instead is more effective, while avoiding the burden of greatly increased training time (compare Appendix F).

s1	Are there good beaches in the Northern part of Qatar?
s2	Fuwairit is very clean !
gold label	True
predictions	BERT:False, BERT+topics:True
manual annotation	domain-specific word:True, named entity:True, non-standard spelling:False

Table 5: Predictions and annotation for an example from SemEval.

6 Conclusion

In this work, we proposed a flexible framework for combining topic models with BERT. We demonstrated that adding LDA topics to BERT consistently improved performance across a range of semantic similarity prediction datasets. In our qualitative analysis, we showed that these improvements were mainly achieved on examples involving domain-specific words. Future work may focus on how to directly induce topic information into BERT without corrupting pretrained information and whether combining topics with other pretrained contextual models can lead to similar gains. Another research direction is to investigate if introducing more sophisticated topic models, such as named entity promoting topic models (Krasnashchok and Jouili, 2018) into the proposed framework can further improve results.

Acknowledgments

This work was supported by Microsoft Azure and The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning research*, 3:993–1022.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided Alignment Training for Topic-Aware Neural Machine Translation. In *Proceedings of AMTA*, pages 121–134, Austin, USA.

Jan Milan Deriu and Mark Cieliebak. 2017. SwissAlps at SemEval-2017 task 3: Attention-based Convolutional Neural Network for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation.*, volume 17, pages 334–338, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 4171—4186, Minneapolis, USA. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP@IJCNLP)*, pages 9–16, Jeju Island, Korea. Asian Federation of Natural Language Processing.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2017. Beihang-MSRA at SemEval-2017 Task 3- A Ranking System with Neural Matching Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 280–286, Vancouver, Canada. Association for Computational Linguistics.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 Task 3- Learning Pairwise Patterns in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 326–333, Vancouver, Canada. Association for Computational Linguistics.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) Models for Large Scale NLP Tasks. *arXiv:1602.06291 [cs]*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural Language Inference over Interaction Space. In *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. 2009. Domain Adaptation with Latent Semantic Association for Named Entity Recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09*, pages 281–289, Boulder, USA. Association for Computational Linguistics.

Liangjie Hong and Brian D. Davison. 2010. Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics - SOMA '10*, pages 80–88, Washington D.C., USA. ACM Press.

Yuening Hu, Ke Zhai, Vladimir Eidelman, and Jordan Boyd-Graber. 2014. Polylingual Tree-Based Topic Models for Translation Domain Adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1166–1176, Baltimore, Maryland. Association for Computational Linguistics.

Yuta Koreeda, Takuya Hashito, Yoshiki Niwa, Misa Sato, Toshihiko Yanase, Kenzo Kurotsuchi, and Kohsuke Yanai. 2017. Bunji at SemEval-2017 Task 3: Combination of Neural Similarity Features and Comment Plausibility Features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 353–359, Vancouver, Canada. Association for Computational Linguistics.

Katsiaryna Krasnashchok and Salim Jouili. 2018. Improving Topic Quality by Promoting Named Entities in Topic Modeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 247–253, Melbourne, Australia. Association for Computational Linguistics.

Jianhua Lin. 1991. Divergence Measures based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Todor Mihaylov and Preslav Nakov. 2016. SemanticZ at SemEval-2016 Task 3: Ranking Relevant Answers in Community Question Answering Using Semantic Similarity Based on Fine-tuned Word Embeddings. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2016)*, pages 879–886, San Diego, USA. Association for Computational Linguistics.

Preslav Nakov, Doris Hoogeveen, Llúis Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on*

Semantic Evaluation (SemEval@ACL 2017), pages 27–48, Vancouver, Canada. Association for Computational Linguistics.

Preslav Nakov, Lluis Marquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, James Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2016)*, pages 525–545, San Diego, USA. Association for Computational Linguistics.

Preslav Nakov, Lluis Marquez, Magdy Walid, Alessandro Moschitti, James Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2015)*, pages 269–281, Denver, USA. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Malte Ostendorff, Peter Bourgonje, Maria Berger, Julian Moreno-Schneider, Georg Rehm, and Bela Gipp. 2019. Enriching BERT with Knowledge Graph Embeddings for Document Classification. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS)*, Erlangen, Germany.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 2793–2799, Phoenix, USA. AAAI Press.

Nicole Peinelt, Maria Liakata, and Dong Nguyen. 2019. Aiming beyond the Obvious: Identifying Non-Obvious Cases in Semantic Similarity Datasets. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2792–2798, Florence, Italy. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, New Orleans, USA. Association for Computational Linguistics.

Zengchang Qin, Marcus Thint, and Zhiheng Huang. 2009. Ranking Answers by Hierarchical Topic Models. In *Next-Generation Applied Intelligence, 22nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2009)*, volume 5579, pages 103–112, Tainan, Taiwan. Springer.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway Attention Networks for Modeling Sentence Pairs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4411–4417, Stockholm, Sweden.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural Paraphrase Identification of Questions with Noisy Pretraining. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 142–147, Copenhagen, Denmark. Association for Computational Linguistics.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. JAIST: Combining Multiple Features for Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 215–219, Denver, USA. Association for Computational Linguistics.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A Reinforced Topic-Aware Convolutional Sequence-to-Sequence Model for Abstractive Text Summarization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4453–4460, Stockholm, Sweden.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4144–4150, Melbourne, Australia.

Guoshun Wu, Yixuan Sheng, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 3- Using Traditional and Deep Learning Methods to Address Community Question Answering Task. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 356–360, Vancouver, Canada. Association for Computational Linguistics.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning Discriminative Projections for Text Similarity Measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, pages 247–256, Portland, USA. Association for Computational Linguistics.

Jianhua Yin and Jianyong Wang. 2014. A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14*, pages 233–242, New York, USA. ACM Press.

D GiBERT - Introducing Linguistic Knowledge into BERT through a Lightweight Gated Injection Method

GiBERT: Introducing Linguistic Knowledge into BERT through a Lightweight Gated Injection Method

Nicole Peinelt[1,2] and **Marek Rei**[3] and **Maria Liakata**[1,2,4]
[1]The Alan Turing Institute, UK
[2]University of Warwick, UK
[3]Imperial College London, UK
[4]Queen Mary University of London, UK
{[redacted]}

Abstract

Large pretrained language models such as BERT have been the driving force behind recent improvements across many NLP tasks. However, BERT is only trained to predict missing words - either behind masks or in the next sentence - and has no knowledge of lexical, syntactic or semantic information beyond what it picks up through unsupervised pre-training. We propose a novel method to explicitly inject linguistic knowledge in the form of word embeddings into any layer of a pre-trained BERT. Our performance improvements on multiple semantic similarity datasets when injecting counter-fitted and dependency-based embeddings indicate that such information is beneficial and currently missing from the original model. Our qualitative analysis shows that counter-fitted embedding injection particularly helps with cases involving synonym pairs.

1 Introduction

With the recent success of pre-trained language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) across many areas of NLP, there is increased interest in exploring how these architectures can be further improved. One line of work aims at model compression, making BERT smaller and accessible while mostly preserving its performance (Xu et al., 2020; Goyal et al., 2020; Sanh et al., 2019; Aguilar et al., 2020; Lan et al., 2020; Chen et al., 2020). Other studies seek to further enhance model performance by introducing external information into BERT, such as information from knowledge bases (Peters et al., 2019; Wang et al., 2020) or multi-modal information (Lu et al., 2019; Lin et al., 2020).

Before the rise of contextualised models, transfer of pre-trained information between datasets and tasks in NLP was based on word embeddings. Over multiple years, substantial effort was placed into the creation of such embeddings. While originally capturing mainly collocation patterns (Mikolov et al., 2013; Pennington et al., 2014), subsequent work enriched these embeddings with additional information, such as dependencies (Levy and Goldberg, 2014), subword information (Luong et al., 2013; Bojanowski et al., 2017) and semantic lexicons (Faruqui et al., 2015). As a result, there exists a wealth of pre-trained embedding resources for many languages in a unified format which could provide complementary information for contemporary pre-trained contextual models. Moreover, aligning contextual embeddings with static embeddings has shown to increase the performance of the former (Liu et al., 2020).

In this work, we propose a new method for injecting pre-trained embeddings into any layer of BERT's internal representation. Our approach differs from previous work by introducing linguistically-enriched embeddings directly into BERT through a novel injection method. We apply our method to multiple semantic similarity detection benchmark datasets and show that injecting pre-trained dependency-based and counter-fitted embeddings can further enhance BERT's performance. More specifically, we make the following contributions:

1. We propose GiBERT - a lightweight gated method for injecting externally pre-trained embeddings into BERT (section 4.1). [1]
2. We provide ablation studies and detailed analysis for core model components (section 6).
3. We demonstrate that our model improves BERT's performance on multiple semantic similarity detection datasets. In comparison to multi-head attention injection, our gated injection method uses fewer parameters while achieving comparable performance for dependency embeddings and improved results for counter-fitted embeddings (section 6).

[1]Code will be provided in the camera-ready version.

4. Our qualitative analysis provides insights into GiBERT's improved performance, such as in cases of sentence pairs involving synonyms. (section 6).

2 Related work

BERT modifications Due to BERT's widespread success in NLP, many recent studies have focused on further improving BERT by introducing external information. Studies differ regarding the type of external information provided, the application area and their technical approach. We broadly categorise existing approaches based on their modification method into input-related, external and internal. *Input modifications* (Zhao et al., 2020; Singh et al., 2020; Lai et al., 2020; Ruan et al., 2020) adapt the information that is fed to BERT - e.g. feeding text triples separated by [SEP] tokens instead of sentence pairs as in Lai et al. (2020) - while leaving the architecture unchanged. *Output modifications* (Xuan et al., 2020; Zhang et al., 2020) build on BERT's pre-trained representation by adding external information after the encoding step - e.g. combining it with additional semantic information as in Zhang et al. (2020) - without changing BERT itself. By contrast, *internal modifications* introduce new information directly into BERT by adapting its internal architecture. Relatively few studies have taken this approach as this is technically more difficult and might increase the risk of so-called catastrophic forgetting - completely forgetting previous knowledge when learning new tasks (French, 1999; Wen et al., 2018). However, such modifications also offer the opportunity to directly harness BERT's powerful architecture to process the external information alongside the pretrained one. Most existing work on internal modifications has attempted to combine BERT's internal representation with visual and knowledge base information: Lu et al. (2019) modified BERT's transformer block with co-attention to integrate visual and textual information, while Lin et al. (2020) introduced a multimodal model which uses multi-head attention to integrate encoded image and text information between each transformer block. Peters et al. (2019) suggested a word-to-entity attention mechanism to incorporate external knowledge into BERT and Wang et al. (2020) proposed to inject factual and linguistic knowledge through separate adapter modules. Our approach differs from previous research as we propose to introduce external information with an addition-based mechanism which uses fewer parameters than existing attention-based techniques (Lu et al., 2019; Lin et al., 2020; Peters et al., 2019). We further incorporate a gating mechanism to scale injected information in an attempt to reduce the risk of catastrophic forgetting. Moreover, our work focuses on injecting pretrained word embeddings, rather than multimodal or knowledge base information as in previous studies.

Semantic similarity detection Detecting paraphrases and semantically related posts in Community Question Answering requires modelling the semantic relationship between a text pair. This is a fundamental and well known NLP problem for which many methods have been proposed. Early work has focused on feature-engineering techniques, exploring various syntactic (Filice et al., 2017), semantic (Balchev et al., 2016) and lexical features (Tran et al., 2015; Almarwani and Diab, 2017). Subsequent work attempted to model text pair relationships solely based on increasingly complex neural architectures (Deriu and Cieliebak, 2017; Wang et al., 2017; Tan et al., 2018) or by combining both approaches through hybrid techniques (Wu et al., 2017a; Feng et al., 2017; Koreeda et al., 2017). Most recently, contextual models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have reached state-of-the-art performance through pretraining large context-aware language models on vast amounts of textual data. Our study joins up earlier lines of work with current state-of-the-art contextual representations by combining BERT with dependency-based and counter-fitted embeddings, previously shown to be useful for semantic similarity detection.

3 Datasets and Tasks

We focus on the task of semantic similarity detection which is a fundamental problem in NLP and involves modelling the semantic relationship between two sentences in a binary classification setup. We work with the following five widely used English language datasets which cover a range of related tasks and sizes (see Appendix A).

MSRP The Microsoft Research Paraphrase dataset (MSRP) contains 5K pairs of sentences from news websites which were obtained based on heuristics and an SVM classifier. Gold labels are based on human binary annotations for sentential

paraphrase detection (Dolan and Brockett, 2005).

SemEval The SemEval 2017 CQA dataset (Nakov et al., 2017) consists of three subtasks involving posts from the online forum Qatar Living[2]. Each subtask provides an initial post as well as 10 posts which were retrieved by a search engine and annotated with binary labels by humans. The task requires the distinction between relevant and non-relevant posts. The original problem is a ranking setting, but since the gold labels are binary, we focus on a classification setup. In **subtask A**, the posts are questions and comments from the same thread, in an answer relevancy detection scenario (26K instances). **Subtask B** is question paraphrase detection (4K instances). **Subtask C** is similar to A but comments were retrieved from an external thread (47K). We use the 2016 test set as the dev set and the 2017 test set as the test set.

Quora The Quora duplicate questions dataset is the largest of the selected datasets, consisting of more than 400K question pairs with binary labels.[3] The task is to predict whether two questions are paraphrases, similar to SemEval subtask B. We use Wang et al. (2017)'s train/dev/test set partition.

All of the above datasets provide two short texts, each usually a single sentence but sometimes consisting of multiple sentences. For simplicity, we refer to each short text as 'sentence'. We frame the task as semantic similarity detection between two sentences in a binary classification task.

4 GiBERT

4.1 Architecture

We propose GiBERT - a **G**ated **I**njection Method for **BERT**. GiBERT's architecture is illustrated with a toy example in Figure 1 and comprises the following phases: obtaining BERT's intermediate representation from Transformer block i (step 1-2 in Figure 1), obtaining an alternative input representation based on linguistically-enriched word embeddings (step 3-4), combining both representations (steps 5-7) and passing on the injected information to subsequent BERT layers to make a final prediction (steps 8-9).

BERT representation We encode a sentence pair with a pre-trained BERT model (Devlin et al. 2019) and obtain BERT's internal representation at different layers (see section 6 for injection layer choices).[4] Following standard practice, we process the two input sentences S_1 and S_2 with a word piece tokenizer (Wu et al., 2017b) and combine them using '[CLS]' and '[SEP]' tokens which indicate sentence boundaries. Then, the word pieces are mapped to ids, resulting in a sequence of word piece ids $\mathbf{E^W} = [w_1, ..., w_N]$ where N indicates the number of word pieces in the sequence (step 1 in Figure 1). In the case of embedding layer injection, we use BERT's embedding layer output denoted with $\mathbf{H^0}$ which results from summing the word piece embeddings $\mathbf{E^W}$, positional embeddings $\mathbf{E^P}$ and segment embeddings $\mathbf{E^S}$ (step 2):

$$\begin{aligned} \mathbf{H^0} &= \text{LayerNorm}(\mathbf{E^W} + \mathbf{E^P} + \mathbf{E^S}) \\ \mathbf{E^W}, \mathbf{E^P}, \mathbf{E^S}, \mathbf{H^0} &\in \mathbb{R}^{N \times D} \end{aligned} \quad (1)$$

where D is the internal hidden size of BERT ($D = 768$ for $\text{BERT}_{\text{BASE}}$). For injecting information at later layers, we obtain BERT's internal representation $\mathbf{H^i} \in \mathbb{R}^{N \times D}$ after transformer block i with $1 \leq i \leq L$ (step 2):

$$\begin{aligned} \mathbf{M^i} &= \text{LayerNorm}(\mathbf{H^{i-1}} + \text{MultiheadAtt}(\mathbf{H^{i-1}})) \\ \mathbf{H^i} &= \text{LayerNorm}(\mathbf{M^i} + \text{FeedForward}(\mathbf{M^i})) \end{aligned} \quad (2)$$

where L is the number of Transformer blocks ($L = 12$ for $\text{BERT}_{\text{BASE}}$) and MultiheadAtt denotes multihead attention.

External embedding representation To enrich this representation, we obtain alternative representations for S_1 and S_2 by looking up word embeddings in a matrix of pre-trained embeddings $\mathbf{E} \in \mathbb{R}^{|V| \times E}$ where $|V|$ indicates the vocabulary size and E is the dimensionality of the pre-trained embeddings (step 3, refer to section 4.2 for details on our choice of pre-trained embeddings). To ensure alignment between BERT's representation at word piece level and the word embedding representation at token level, an alignment function copies embeddings of tokens that were separated into multiple word pieces and adds BERT's special '[CLS]' and '[SEP]' tokens, resulting in an injection sequence $\mathbf{I} \in \mathbb{R}^{P \times E}$ (step 4). For example, we copy the pre-trained embedding of the word 'prompt' to match the two corresponding word pieces 'pro' and '##mpt' (see Figure 1).

[2] https://www.qatarliving.com/

[3] https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning

[4] We use the the uncased version of $\text{BERT}_{\text{BASE}}$ available through Tensorflow Hub.

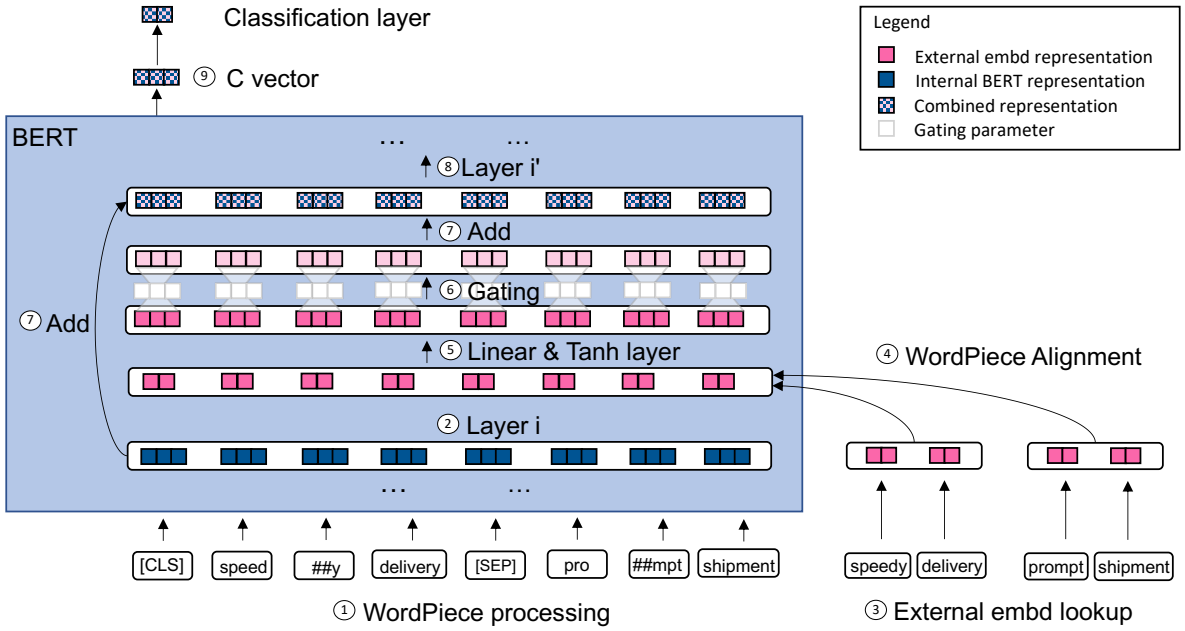


Figure 1: Our proposed GiBERT architecture illustrated with a toy example (where internal BERT dimension $d = 3$ and embedding dimension $e = 2$). The model input consists of a sentence pair which is processed with a WordPiece tokenizer (step 1) and encoded with BERT up to layer i (step 2). We obtain an alternative representation for the sentence pair based on pretrained word embeddings (step 3), while ensuring that external word embeddings are aligned with BERT's WordPieces by repeating embeddings for tokens which have been broken down into several WordPieces (step 4). The aligned word embedding sequence is passed through a linear and tanh layer to match BERT's embedding dimension (step 5). We apply a gating mechanism (step 6) before adding the injected information to BERT's representation from layer i (step 7). The combined representation is passed on to the next layer (step 8). At the final layer, the C vector is used as the sentence pair representation, followed by a classification layer (step 9).

Attention Injection Multihead attention was proposed by Vaswani et al. (2017):

$$\begin{aligned}&\text{MultiheadAtt}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\text{head}_1;\ldots;\text{head}_h]\mathbf{W^O}\\&\text{where head}_i = \text{Attention}(\mathbf{QW_i^Q},\mathbf{KW_i^K},\mathbf{VW_i^V})\end{aligned} \tag{3}$$

and is employed in Transformer networks in the form of self-attention (where queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ come from the previous layer) or encoder-decoder attention (where queries come from the decoder, keys and values from the encoder). Previous work has successfully employed multihead attention to combine BERT with external information (see section 2). For example, in their multimodal VilBERT model, Lu et al. (2019) combined textual and visual representations by passing the keys and values from each modality as input to the other modality's multi-head attention block. Similarly, Peters et al. (2019) used multihead attention to combine projected BERT representations (as queries) with entity-span representations (as keys and values) in their knowledge-enrichment method for BERT. For our case of combining BERT with the injection sequence, it is therefore intuitive to try to use the following multi-head attention injection method:

$$\mathbf{H^{i'}} = \mathbf{H^i} + \text{MultiHeadAtt}(\mathbf{H^i},\mathbf{I},\mathbf{I}) \tag{4}$$

where queries are provided by BERT's internal representation, while keys and values come from the injected embeddings. The output of the attention mechanism is then combined with the previous layer through addition.

Gated Injection The above multihead attention injection mechanism is rather complex and requires many parameters. We therefore propose an alternative way of combining external embeddings with BERT which requires only 14% of parameters used in multi-head attention (0.2M instead of 1.6M, see Appendix C). First, we add a feed-forward layer – consisting of a linear layer with weights $\mathbf{W^P} \in \mathbb{R}^{D\times E}$ and bias $\mathbf{b^P} \in \mathbb{R}^D$ with a tanh activation function – to project the aligned embedding sequence to BERT's internal dimensions and squash the output values to a range between -1

and 1 (step 5):

$$\mathbf{P} = \text{FeedForward}(\mathbf{I}) \in \mathbf{R}^{N \times D} \quad (5)$$

Then, we use a residual connection to inject the projected external information into BERT's representation from Transformer block i (see section 6 for injection at different locations) and obtain a new enriched representation $\mathbf{H^{i'}} \in \mathbb{R}^{N \times D}$:

$$\mathbf{H^{i'}} = \mathbf{H^{i}} + \mathbf{P} \quad (6)$$

However, as injection values can get rather large (between -1 and 1) in comparison to BERT's internal representation (based on our observation usually ranging around -0.1 to 0.1), a downside of directly injecting external information in this way is that BERT's pre-trained information can be easily overwritten by the injection, resulting in catastrophic forgetting. To address this potential pitfall, we further propose a gating mechanism which uses a gating vector $\mathbf{g} \in \mathbb{R}^{D}$ to scale the injected information before combining it with BERT's internal representation as follows:

$$\mathbf{H^{i'}} = \mathbf{H^{i}} + \mathbf{g} \odot \mathbf{P} \quad (7)$$

where $\odot$ denotes element-wise multiplication using broadcasting (step 6 & 7). The gating parameters are initialised with zeros and updated during training. This has the benefit of starting finetuning from representations which are equivalent to vanilla BERT and gradually introducing the injected information during finetuning along certain dimensions. In case the external representations are not beneficial for the task, it is easy for the model to ignore them by keeping the gating parameters at zero.

Output layer The combined representation $\mathbf{H^{i'}}$ is then fed as input to BERT's next Transformer block $i + 1$ (step 8). At the final Transformer block $\mathbf{L}$, we use the $\mathbf{c} \in \mathbb{R}^{D}$ vector which corresponds to the '[CLS]' token in the input and is typically used as the sentence pair representation (step 9). As proposed by Devlin et al. (2019), this is followed by a softmax classification layer (with weights $\mathbf{W^L} \in \mathbb{R}^{C \times D}$ and $\mathbf{b^L} \in \mathbb{R}^{C}$) to calculate class probablilities where C indicates the number of classes. During finetuning, we train the entire model for 3 epochs with early stopping and cross-entropy loss. Learning rates are tuned for each seed and dataset based on development set performance (reported in Appendix D).

4.2 Injected Embeddings

While any kind of information could be injected, we focus on two types of pretrained embeddings: dependency-based (Levy and Goldberg, 2014) and counter-fitted embeddings (Mrkšić et al., 2016). Our choice is motivated by previous research which found syntactic features useful for semantic similarity detection (Filice et al., 2017; Feng et al., 2017) and counter-fitted embeddings helpful in several other tasks (Alzantot et al., 2018; Jin et al., 2020).

The 300-dim dependency-based embeddings by Levy and Goldberg (2014) extend the SkipGram embedding algorithm proposed by Mikolov et al. (2013) by replacing linear bag-of-word contexts with dependency-based contexts which are extracted from parsed English Wikipedia sentences. As BERT has not been exposed to dependencies during pretraining and previous studies have found that BERT's knowledge of syntax is only partial (Rogers et al., 2020), we reason that these embeddings could provide complementary information.

The 300-dim counter-fitted embeddings by Mrkšić et al. (2016) integrate antonymy and synonymy relations into word embeddings based on an objective function which combines three principles: repelling antonyms, attracting synonymy and preserving the vector space. For training, they obtain synoynmy and antonymy pairs from the Paraphrase Database and WordNet, demonstrating an increased performance on SimLex-999. We use their highest-scoring vectors which they obtained by applying their counter-fitting method to Paragram vectors from Wieting et al. (2015). We reason that the antonym and synonym relations contained in the word embeddings could be especially useful for paraphrase detection by explicitly capturing these semantic relations.

5 Evaluation

5.1 Setup

Preprocessing and Hyperparameters We lowercase and tokenise all datasets, replacing images and URLs with placeholders. Sequences exceeding the maximum length are cut off. Hyperparameter settings of all BERT-based models are identical, except for learning rate and injection location which are tuned with grid search, see Appendix D. Each model is trained on a single NVIDIA Tesla K80 GPU.

Metrics Our main evaluation metric is F1 score as this is more meaningful for datasets with imbalanced label distributions (such as SemEval C, see Appendix A) than accuracy. We also report performance on difficult cases using the non-obvious F1 score (Peinelt et al., 2019). This metric distinguishes obvious from non-obvious instances in a dataset based on lexical overlap and gold labels and calculates a separate F1 score for challenging cases. It therefore tends to be lower than the normal F1 score. As Dodge et al. (2020) recently showed that early stopping and random seeds can have considerable impact on the performance of finetuned BERT models, we finetune all models for 3 epochs with early stopping (based on dev F1) and report average model performance across two different seeds.

5.2 Baselines

SemEval systems We compare against the best SemEval 2017 system for each subtask based on F1 score: KeLP (Filice et al., 2017), ECNU (Wu et al., 2017a) and Bunji (Koreeda et al., 2017).

BERT Following standard practice, we encode the sentence pair with BERT's C vector from the final layer, followed by a softmax layer as proposed by Devlin et al. (2019). We use Tensorflow Hub's distribution of $\text{BERT}_{\text{BASE}}$.

SemBERT We also compare with the semantics-aware BERT model (SemBERT, Zhang et al. 2020) which leverages a semantic role labeler. As the original paper reports results on different dataset versions, we ran the official code on our datasets. The longer sentences in SemEval A and C could not be fit on a single GPU due to the larger model size.

tBERT Moreover, we attempt to combine embeddings with BERT using an averaging and concatenation method proposed in tBERT (Peinelt et al., 2020). Instead of the word topics in the original system, we use pretrained counter-fitted and dependency embeddings for a direct comparison with our methods.

AiBERT We further provide an alternative **A**ttention-based embedding **I**njection method for **BERT** based on the multihead attention injection mechanism described in equations 3 to 4. Following the same procedure as GiBERT, we tune the injection location (see Appendix E).

6 Results

Gating Mechanism Catastrophic forgetting is a potential problem when introducing external information into a pre-trained model as the injected information could disturb or completely overwrite existing knowledge (Wang et al., 2020). In our proposed model, a gating mechanism is used to scale injected embeddings before adding them to the pre-trained internal BERT representation (see section 4.1). To understand the importance of this mechanism, we contrast development set performance for injecting information after the embedding layer with gating - as defined in equation 7 - and without - as in equation 6 - (Table 1). For dependency embedding injection without gating, performance only improves on 2 out of 5 datasets over the baseline and in some cases even drops below BERT's performance, while it outperforms the baseline on all datasets when using the gating mechanism. Counter-fitted embedding injection without gating improves on 4 out of 5 datasets, with further improvements when adding gating, outperforming the vanilla BERT model across all datasets. In addition, gating makes model training more stable and reduces failed runs (where the model predicted only the majority class) from 30% to 0% on the particularly imbalanced SemEval C dataset. This highlights the importance of the gating mechanism in our proposed architecture.

	MSRP	Quora	SemEval		
			A	B	C
BERT	.906	.906	.714	.754	.414
GiBERT with dependency embeddings					
- no gating	.906	.905	.732	.751	.424
- with gating	**.913**	**.908**	**.755**	**.778**	**.433**
GiBERT with counter-fitted embeddings					
- no gating	**.907**	.906	.733	.763	.435
- with gating	**.907**	**.908**	**.751**	**.767**	**.451**

Table 1: Development set F1 scores of GiBERT models injecting pretrained embeddings after the embedding layer with vs. without gating mechanism.

Injection Location In our proposed model, information can be injected between any of BERT's pre-trained transformer blocks. We reason that different locations may be more appropriate for certain kinds of embeddings as previous research has found that different types of information tend to be encoded and processed at specific BERT layers (Rogers et al., 2020). We experiment with injecting

	MSRP	Quora	SemEval		
			A	B	C
BERT	.906	.906	.714	.754	.414
GiBERT with dependency embeddings					
- embd layer	.913	.908	.755	**.778**	.433
- layer 6	.911	.908	.755	.776	.438
- layer 11	**.914**	**.910**	**.760**	.773	**.444**
GiBERT with counter-fitted embeddings					
- embd layer	.907	.908	.751	.767	.451
- layer 6	**.917**	**.909**	**.760**	**.771**	**.464**
- layer 11	.910	.907	.755	**.771**	.450

Table 3: Development set F1 scores of embedding injection at different layers.

embeddings at three possible locations: after the embedding layer (using $\mathbf{H^0}$), after the middle layer (using $\mathbf{H^6}$ in $\text{BERT}_{\text{BASE}}$) and after the penultimate layer (using $\mathbf{H^{11}}$ in $\text{BERT}_{\text{BASE}}$). Table 3 shows that midlayer injection is ideal for counter-fitted embeddings, while late injection appears to work best for dependency embeddings (Table 3). This is in line with previous work which found that BERT tends to processes syntactic information at later layers than linear word-level information (Rogers et al., 2020). We consequently use these injection locations in our final model (see Appendix E for the tuned injection location in AiBERT).

Full Model GiBERT with counter-fitted embeddings outperforms all other systems in both average F1 and average non-obvious F1 score (see Table 2). This shows that the model improves on challenging dataset instances, rather than merely leveraging shallow surface patterns. It is worth noting that

	Total parameters
BERT	110.1M
GiBERT	110.3M
tBERT	111.0M
AiBERT	111.7M
SemBERT	111.9M

Table 4: BERT-based models ordered by size (using $\text{BERT}_{\text{BASE}}$).

GiBERT has the fewest parameters of all BERT-enhancing models (see Table 4) and doesn't require any additional preprocessing tools (such as the neural SRL tagger required by SemBERT), making it more efficient than SemBERT, tBERT and AiBERT. The largest improvement of GiBERT over BERT is observed with counter-fitted embeddings, especially on the internal CQA datasets SemEval A and B (the datasets with the highest proportion of examples involving synonym pairs, see section 6). GiBERT with dependency embeddings still improves over vanilla BERT, but performance gains tend to be smaller and roughly similar to the more complex AiBERT injection method. We reason that the injection of dependency embeddings is less effective because semantic information is more important for the tasks at hand or because syntactic information needs to be introduced in a more direct way.

Error Analysis Counter-fitted embeddings are designed to explicitly encode synonym and antonym relationships between words. To better understand how the injection of counter-fitted embeddings affects the ability of our model to deal

	F1						**non-obvious F1**					
	MSRP	Quora	SemEval			avg	MSRP	Quora	SemEval			avg
			A	B	C				A	B	C	
Previous systems												
KeLP◇	-	-	-	.506	-	-	-	-	-	.199	-	-
ECNU◇	-	-	.777	-	-	-	-	-	.707	-	-	-
Bunji◇	-	-	-	-	.197	-	-	-	-	-	.028	-
BERT⋆	.876	.902	.704	.473	.268	.645	.827	.860	.656	.243	.085	.534
SemBERT⋆	.876	.901	✗	✗	✗	-	.820	**.874**	✗	✗	✗	-
$\text{tBERT}_{\text{dependency}}$⋆	.882	.906	.780	.510	.242	.664	.827	.858	**.728**	.262	.090	.553
$\text{tBERT}_{\text{counter-fitted}}$⋆	.879	.906	.756	.500	.215	.651	.824	.857	.699	.258	.064	.540
Our implementation												
$\text{AiBERT}_{\text{dependency}}$	.863	.903	.738	.498	**.282**	.657	.792	.866	.681	**.268**	**.090**	.539
$\text{AiBERT}_{\text{counter-fitted}}$	.877	.904	.724	.496	.263	.653	.835	.867	.662	.264	.076	.541
$\text{GiBERT}_{\text{dependency}}$	.883	.904	.768	.474	.238	.653	.849	.864	.704	.231	.087	.547
$\text{GiBERT}_{\text{counter-fitted}}$	**.884**	**.907**	**.780**	**.511**	.256	**.668**	**.858**	.862	.719	.248	**.090**	**.555**

Table 2: Model performance on test set. All BERT-based methods use $\text{BERT}_{\text{BASE}}$. avg = average performance across all datasets, ◇ = results from publication, ⋆ = official code run on our data, ✗= run failed due to insufficient GPU memory, $_{\text{dep}}$ = dependency embeddings, $_{\text{counter}}$ = counter-fitted embeddings.

	Sentence 1	Sentence 2	Gold label	BERT prediction	GiBERT prediction
(1)	it took me more than 10 people; over the course of the whole day to convince my point at qatar **airways**... as to how my points needs to be redeemed... at long last my point was made... dont seem know what they are doing??? appalling to say the least	this isn't the first time. so many rants by irate customers on so many diverse situations signals a very serious problem. so called first class **airlines** and no basic customer care. over confidence much?	is related	not related	is related
(2)	hi; my **wife** was on a visit visa; today; her residency visa was issued; so i went to immigration and paid 500 so there is no need to leave the country and enter again on the residency visa . she has done her medical before for the visit visa extension; do we need to do the medical again for the residency visa? thanks	dear all; please let me know how many days taking for approve family visa nw; am last wednesday (12/09/2012) apply family visa for my **husband** and daughter; but still now showing in moi website itz under review; itz usual reply? why delayed like this? please help me regards divya	is related	is related	not related

Table 5: Examples from the Semeval development set. Synonym and antonym pairs are highlighted in bold.

with instances involving such semantic relations, we use synonym and antonym pairs from the PPDB and Wordnet (provided by Mrkšić et al. 2016) and search the development partition of the datasets for sentence pairs where the first sentence contains one word of the synonym/antonym pair and the second sentence the other word. Table 6 reports F1 performance of our model on cases with synonym pairs, antonym pairs and neither one. We find that our model's F1 performance particularly improves over BERT on instances containing synonym pairs, as illustrated in example (1) in Table 5. By contrast, the performance on cases with antonym pairs stays roughly the same, although slightly decreasing on Quora. As illustrated by example (2) in Table 5, word pairs can be antonyms in isolation (e.g. husband - wife), but not in the specific context of a given example. In rare cases, the injection of distant antonym pair embeddings can therefore deter the model from detecting related sentence pairs. We also observe a slight performance boost for cases without synonym or antonym pairs which could be due to improved representations for words which occurred in examples without their synonym or antonym counterpart.

	MSRP	Quora	SemEval		
			A	B	C
Instances with **antonym** pairs					
	(4%)	(4%)	(21%)	(28%)	(20%)
BERT	**.81**	**.87**	**.77**	**.75**	**.46**
GiBERT	**.81**	.86	**.77**	**.75**	**.46**
Instances with **synonym** pairs					
	(11%)	(9%)	(22%)	(31%)	(17%)
BERT	.87	.90	.81	.78	**.54**
GiBERT	**.90**	**.91**	**.82**	**.83**	**.54**
Instances **without** synonym/antonym pairs					
	(85%)	(87%)	(64%)	(51%)	(68%)
BERT	.91	**.91**	.71	.72	.36
GiBERT	**.92**	**.91**	**.73**	**.73**	**.41**

Table 6: F1 score on instances containing synonymy pairs, antonymy pairs or no pairs across datasets. (Note that an instance can contain both synonym and antonym pairs, therefore the added percentage of the three groups can exceed 100.)

7 Conclusion

In this paper, we introduced a new approach for injecting external information into BERT. Our proposed method adds linguistically enriched embeddings to BERT's internal representation through a lightweight gating mechanism which requires considerably fewer parameters than previous approaches. Evaluating our injection method on multiple semantic similarity detection datasets, we demonstrated that injecting counter-fitted embeddings clearly improved performance over vanilla BERT and on average outperformed all baselines on the task, while dependency embedding injection achieved slightly smaller gains. In comparison to the multihead attention injection mechanism, we found the gated method at least as effective, with comparable performance for dependency embeddings and improved results for counter-fitted embeddings. Our qualitative analysis highlighted that counter-fitted injection was particularly helpful for instances involving synonym pairs. Future work could explore combining multiple embedding sources or injecting other types of information. Another direction is to investigate the usefulness of embedding injection for other tasks.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. 2020. Knowledge Distillation from Internal Representations. *arXiv:1910.03723 [cs]*.

Nada Almarwani and Mona Diab. 2017. GW_QA at SemEval-2017 Task 3- Question Answer Reranking on Arabic Fora. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 344–348, Vancouver, Canada. Association for Computational Linguistics.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating Natural Language Adversarial Examples. *arXiv:1804.07998 [cs]*.

Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov. 2016. PMI-cool at SemEval-2016 Task 3: Experiments with PMI and Goodness Polarity Lexicons for Community Question Answering. In *SemEval@ NAACL-HLT*, pages 844–850.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Daoyuan Chen, Yaliang Li, Minghui Qiu, Zhen Wang, Bofang Li, Bolin Ding, Hongbo Deng, Jun Huang, Wei Lin, and Jingren Zhou. 2020. AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search. *arXiv:2001.04246 [cs]*.

Jan Milan Deriu and Mark Cieliebak. 2017. SwissAlps at SemEval-2017 Task 3: Attention-based Convolutional Neural Network for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation.*, volume 17, pages 334–338, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 4171—4186, Minneapolis, USA. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv:2002.06305 [cs]*.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP@IJCNLP)*, pages 9–16, Jeju Island, Korea. Asian Federation of Natural Language Processing.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2015)*, pages 1606–1615, Denver, USA. Association for Computational Linguistics.

Wenzheng Feng, Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2017. Beihang-MSRA at SemEval-2017 Task 3- A Ranking System with Neural Matching Features for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 280–286, Vancouver, Canada. Association for Computational Linguistics.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 Task 3- Learning Pairwise Patterns in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017*, pages 326–333, Vancouver, Canada. Association for Computational Linguistics.

Robert French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Saurabh Goyal, Anamitra Roy Choudhary, Venkatesan Chakaravarthy, Saurabh ManishRaje, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT inference for Classification Tasks. *arXiv:2001.08950 [cs, stat]*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *arXiv:1907.11932 [cs]*.

Yuta Koreeda, Takuya Hashito, Yoshiki Niwa, Misa Sato, Toshihiko Yanase, Kenzo Kurotsuchi, and Kohsuke Yanai. 2017. Bunji at SemEval-2017 Task 3: Combination of Neural Similarity Features and Comment Plausibility Features. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 353–359, Vancouver, Canada. Association for Computational Linguistics.

Tuan Manh Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2020. A Simple but Effective BERT Model for Dialog State Tracking on Resource-Limited Systems. *arXiv:1910.12995 [cs]*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv:1909.11942 [cs]*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, USA. Association for Computational Linguistics.

Junyang Lin, An Yang, Yichang Zhang, Jie Liu, Jingren Zhou, and Hongxia Yang. 2020. InterBERT: Vision-and-Language Interaction for Multi-modal Pretraining. *arXiv:2003.13198 [cs]*.

Qianchu Liu, Diana McCarthy, and Anna Korhonen. 2020. Towards Better Context-aware Lexical Semantics:Adjusting Contextualized Representations through Static Anchors. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 4066–4075.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *arXiv:1908.02265 [cs]*.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations (ICLR 2013)*, Scottsdale, USA.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. *arXiv:1603.00892 [cs]*.

Preslav Nakov, Doris Hoogeveen, Llúis Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 27–48, Vancouver, Canada. Association for Computational Linguistics.

Nicole Peinelt, Maria Liakata, and Dong Nguyen. 2019. Aiming beyond the Obvious: Identifying Non-Obvious Cases in Semantic Similarity Datasets. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2792–2798, Florence, Italy. Association for Computational Linguistics.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. 2020. tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7047–7055, Seattle, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, New Orleans, USA. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge Enhanced Contextual Word Representations. *arXiv:1909.04164 [cs]*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What we know about how BERT works. *arXiv:2002.12327 [cs]*.

Yu-Ping Ruan, Zhen-Hua Ling, Jia-Chen Gu, and Quan Liu. 2020. Fine-Tuning BERT for Schema-Guided Zero-Shot Dialogue State Tracking. *arXiv:2002.00181 [cs]*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*.

Gaurav Singh, Zahra Sabet, John Shawe-Taylor, and James Thomas. 2020. Constructing Artificial Data for Fine-tuning for Low-Resource Biomedical Text Tagging with Applications in PICO Annotation. *arXiv:1910.09255 [cs, stat]*.

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway Attention Networks for Modeling Sentence Pairs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4411–4417, Stockholm, Sweden.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. JAIST: Combining Multiple Features for Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 215–219, Denver, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All

You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 5998–6008, Long Beach, USA.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu ji, Cuihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. *arXiv:2002.01808 [cs]*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4144–4150, Melbourne, Australia.

Junfeng Wen, Yanshuai Cao, and Ruitong Huang. 2018. Few-Shot Self Reminder to Overcome Catastrophic Forgetting. *arXiv:1812.00543 [cs, stat]*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Guoshun Wu, Yixuan Sheng, Man Lan, and Yuanbin Wu. 2017a. ECNU at SemEval-2017 Task 3- Using Traditional and Deep Learning Methods to Address Community Question Answering Task. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval@ACL 2017)*, pages 356–360, Vancouver, Canada. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017b. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *Transactions of the Association for Computational Linguistics*, pages 339–351.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. BERT-of-Theseus: Compressing BERT by Progressive Module Replacing. *arXiv:2002.02925 [cs]*.

Zhenyu Xuan, Chuyu Ma, and Shengyi Jiang. 2020. FGN: Fusion Glyph Network for Chinese Named Entity Recognition. *arXiv preprint: 2001.05272*.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware BERT for Language Understanding. *arXiv:1909.02209 [cs]*.

Lingyun Zhao, Lin Li, and Xinhao Zheng. 2020. A BERT based Sentiment Analysis and Key Entity Detection Approach for Online Financial Texts. *arXiv:2001.05326 [cs]*.