Applications of Sketching and Pathways to Impact

Graham Cormode
g.cormode@warwick.ac.uk
Meta & University of Warwick
Coventry, UK

ABSTRACT

Data summaries (a.k.a., sketches) are compact data structures that can be updated flexibly and efficiently to capture certain properties of a data set. Well-known examples include set summaries (Bloom Filters) and cardinality estimators (e.g., Hyperloglog), amongst others. PODS and SIGMOD have been home to many papers on sketching, including several best paper recipients. Sketch algorithms have emerged from the theoretical research community, but have found wide impact in practice. This paper describes some of the impacts that sketches have had, from online advertising to privacy-preserving data analysis. It will consider the range of different strategies that researchers can follow to encourage the adoption of their work, and what has and has not worked for sketches as a case study.

CCS CONCEPTS

• **Theory of computation → Streaming models**; **Sketching and sampling**.

KEYWORDS

sketching; data sketches; summarization

ACM Reference Format:
Graham Cormode. 2023. Applications of Sketching and Pathways to Impact. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3584372.3589937

1 INTRODUCTION

The concept of a "sketch" of data is that of a compact data summary that is designed to capture certain properties of the input data. The sketch comprises a description of the data structure, and algorithms that can update the structure and query it. This can include routines to update the sketch with a single new piece of information (capturing a streaming model of data processing), or to merge together two sketches (capturing a distributed model of data processing).

The queries that are supported by a sketch are usually to efficiently approximate some function of the input data. For instance, some sketches report the cardinality of the set of input items that they have processed, leading to the count distinct (a.k.a. F_0) sketches. Sketches for dimensionality reduction approximate the Euclidean norm (or other norm) of their input, interpreted as a high-dimensional vector. Other sketches represent frequency histograms in order to answer heavy hitter or quantile queries. Sketches can also capture properties of more complex data types, such as graphs, and matrices.

Sketching algorithms make use of a number of basic algorithmic concepts. Deterministic sketches track counts and other simple statistics of the input data in order to give exact or approximate results. But the majority of sketches use randomization to provide a summary that obtains an accurate approximation with high probability. Key techniques include random sampling, or other probabilistic ways to select information about the set of input items; and hashing, to select items from the input in a random but repeatable way. These are combined with tracking counts or setting bits in a particular fashion. From this collection of methods, a wide variety of techniques can flourish. For instance, both the Bloom filter and Hyperloglog summary use hashing and bit vectors to represent the input, but for quite different purposes (tracking approximate set membership, and approximate set cardinality, respectively).

The purpose of this article, accompanying a talk at the PODS 2023 conference, is not to give a complete overview of sketches or their workings. Rather, the aim is to give a flavour of the development of sketch algorithms over the decades, and to cast some light on how they have found applications in practice.

We proceed as follows: in Section 2 we present a history of the development of sketches over the years. We complement this in Section 3 by looking at the corresponding timeline of how sketches have been motivated for different problems. In Section 4 we consider some of the ways that have been tried to encourage the adoption of sketches in practice, and offer concluding remarks in Section 5.

2 A (VERY) BRIEF HISTORY OF SKETCHES

The Pre-history of sketching (1970s and 1980s). The earliest instance of something that we could reasonably refer to as a sketch algorithm would be (uniform) random sampling, which far predates computers. Computer science and random sampling intersected with the notion of 'reservoir sampling': drawing a uniform random sample from a large stream of examples, whose cardinality is initially unknown. The simple incremental reservoir sampling algorithm is attributed variously to Fan et al. and to Waterman[1]. Generalizations of sampling have led to a wide range of statistical techniques, going far beyond what can be discussed in this short note.

Apart from sampling, the earliest sketches started to emerge along with the wider availability of programmable computers, in the second half of the 20th century, from the 1970s onwards. Perhaps the first example of something we can think of as a sketch is due

[1]See the discussion in https://markkm.com/blog/reservoir-sampling/

PODS '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0127-6/23/06.
https://doi.org/10.1145/3584372.3589937

to Bloom in 1970, in the form of the famed 'Bloom Filter' [9] (if any reader can point to earlier examples, I would be delighted to hear of them). The Bloom filter compactly represents a set as a collection of bits, and is easy to update with new entries, and to query for (approximate) set membership. It does however take space that is linear in the size of the set that is represented (albeit with a small constant of proportionality). For an asymptotic space reduction, we have to look to examples such as the Morris counter (1977) [37], which allows us to count n events approximately in space proportional to $O(\log \log n)$, rather than the exact binary counter that requires $\log_2 n$ bits. There is also the Flajolet and Martin distinct counter (1983), which uses $O(\log n)$ bits, but tracks the number of *distinct* items that have been observed within the input [22].

Other notions developed during these early years is the Munro-Paterson approach to finding quantiles in sublinear space (1980) [38]. The original focus of the work was to find the exact median (and other order statistics) with multiple passes over the input (assumed to be stored on tape). Later work reframed these results as providing deterministic approximations with a single pass. Boyer and Moore provided a simple algorithm to find the majority item in a sequence (1981) [11], which was generalized by Misra and Gries to find all frequently occurring items [35]. From mathematics, the Johnson-Lindenstrauss lemma (1984) [26] argued that Euclidean distances could be preserved among a set of high-dimensional points via a suitable projection. However, it took until the 1990s before explicit constructions emerged, based on random projections that were approximately distance preserving. A quirk of the literature is that these sketches are mostly known by the surnames of their designers: Bloom, Morris, Flajolet-Martin, Munro and Paterson, Johnson-Lindenstrauss.

The Streaming Years. The area of sketching accelerated from the mid 1990s through the first decade of the 2000s (approximately), due to the sudden growth in interest in 'data streaming' and the streaming model of computation. This model was formulated based on sources of large volumes of data, where it was necessary to process many small, incremental updates as a 'stream' of information. The work of Alon, Matias and Szegedy on the space complexity of the frequency moments launched the interest in this from an algorithmic perspective [5]. One key result was their "tug-of-war" or AMS sketch, based on maintaining the inner product of the input with Rademacher random variables (which can be viewed as small space version of the Johnson-Lindendstrauss lemma). In parallel, Indyk and Motwani introduced the notion of Locality Sensitive Hashing, which builds a sketch of a large object, such that similar objects are likely to have similar sketches – also relying in part on Johnson-Lindenstrauss ideas [25].

The problem of finding the quantiles from a stream of items has been a keystone problem for sketching over the years. Manku, Rajagopalan and Lindsay adapted the Munro-Paterson algorithm to the streaming setting, and proposed extensions that obtained polylogarithmic space bounds [32]. Greenwald and Khanna presented and analyzed a streaming algorithm for quantiles that obtained logarithmic space [23]. Then Shrivastava et al. presented the q-digest sketch for quantile estimation, which focused on mergability for distributed data [44].

Sketches based on carefully structured random projection appeared. The Count sketch can be viewed as an improvement of the AMS sketch, replacing averaging with hashing to speed up the computation [12]. Originally proposed for estimating item frequencies, it has been generalized as the basis of sparse Johnson-Lindenstrauss transforms. The Count-Min sketch seeks to further streamline sketching, by removing the Rademacher random variables, in order to provide frequency estimation with L_1 instead of L_2 guarantees [14]. The SpaceSaving algorithm was introduced to give a fast, deterministic solution to frequency estimation [34]; it was later connected with the similar Misra-Gries algorithm [35].

The distinct element counting problem was also revisited in the streaming model, with the aim of providing strong approximation guarantees with tighter space bounds. The loglog algorithm reduced the dependence on the cardinality from logarithmic to double-logarithmic [18]. Subsequently, the hyperloglog (HLL) further squeezed the space cost for this problem, while remaining very simple to implement (the same cannot be said about the algorithmic analysis, which is highly sophisticated) [21].

During this era, it was common to refer to sketches by initialisms of their authors, e.g., AMS, MRL, GK, CM; or by names given to the sketches, such as SpaceSaving, Q-Digest, LogLog, Hyperloglog and Count Sketch.

From streaming to mergable. Interest in sketches for the streaming model of computation has remained, but over the last decade there has been more emphasis on generalizing sketches to work in distributed settings (as opposed to on centralized streams), and on improving their performance for practical implementations.

Agarwal et al. [3] placed emphasis on the notion of mergability of summaries, drawing out a theme that was present in many prior works. They provided new results on mergability of deterministic frequency estimation algorithms and randomized quantile algorithms. A sequence of papers further tightened results on quantiles, leading to the Karnin-Lang-Liberty (KLL) [30] optimal quantile sketch, combining sampling with sketching ideas.

Some deep theoretical advances were made. Truly sparse constructions of the Johnson-Lindenstrauss lemma were presented by Kane and Nelson, similar in outline to the Count Sketch but with stronger guarantees [28]. Such dimensionality reduction techniques led to the development of the areas of compressed sensing [17] and subspace embeddings [48]. Sketch techniques for graphs were developed by Ahn, Guha and McGregor, based on L_p sampling, which allowed dynamic connectivity and minimum spanning trees to be solved in near-linear space [4].

On the practical side, work from Google discussed how to optimize the HLL algorithm for tracking cardinalities of very high magnitude, while improving accuracy at small cardinalities [24]. A team from Yahoo! started the "data sketches" project, which aimed to provide robust implementations of many sketches to ease adoption. The project emphasised the need for concurrency and mergability of sketches [41].

Sketching at PODS. The PODS conference has been a welcoming home for results on sketching, with many papers on sketching and related topics appearing here. A complete enumeration is out of the question, and instead I highlight some examples of papers on sketching that have been honoured with awards at PODS:

- "An optimal algorithm for the distinct elements problem" [29] (PODS 2010, best paper award) gives a sketch algorithm that achieves the lower bound for counting the number of distinct items in a stream of updates.
- "Tight bounds for L_p samplers" [27] (PODS 2011, Test of time award in 2021) uses sketch techniques to sample according to the L_p distribution, where the probability of picking an item is proportional to a monomial function of its frequency.
- "Mergeable Summaries" [3] (PODS 2012, Test of time award in 2022) formalizes the notion of mergeable summaries, and shows sketches that can be merged for frequency estimation, quantiles, and geometric approximations.
- "A framework for adversarially robust streaming algorithms" [7] (PODS 2020, best paper award) considers how randomized sketch algorithms can be built that are robust to an adversary trying to break the approximation guarantee.
- "Relative Error streaming quantiles" [13] (PODS 2021, best paper award) gives a near-optimal sketch for the problem of summarizing a stream of items to find the quantiles with a relative error guarantee.
- "Optimal Bounds for Approximate Counting" [39] (PODS 2022, best paper award) revisits the foundational problem of approximate counting [37] and shows a variant sketch that achieves improved dependency on the approximation parameters.

Collectively, these demonstrate the depth and challenge of problems relating to sketches, and their high level of interest to the PODS community.

Sketching in print. For further reading, and a technical presentation of the fundamentals of sketching, there are now several textbooks that cover the topic in depth. "Probability and Computing" by Mitzenmacher and Upfal [36] is a general introduction to probabilistic algorithms, which uses some sketch algorithms to illustrate the key techniques. "Mining of Massive Datasets" (Leskovec, Rajamaran and Ullman) [31] devotes a chapter to sketch techniques for data analysis. Likewise, "Foundations of Data Science" by Blum, Hopcroft and Kannan [10] has a chapter on sketching as a mathematical tool in data science. The book "Algorithms and Data Structures for Massive Datasets" by Medjedovic, Tahirovic and Dedovic [33] is almost entirely concerned with presenting core sketch algorithms and their analyses, and similarly, "Small Summaries for Big Data" (Cormode and Yi) [15] presents multiple sketch algorithms, and discusses implementation issues.

3 THE SHIFTING MOTIVATIONS FOR SKETCHING.

As hinted in the previous section, the motivations for using sketches have shifted over time, presenting different demands on the algorithms, and highlighting different concerns. Some applications have faded due to shifts in technologies, while others remain broadly relevant today.

Memory constrained systems (1970s—1980s). The initial motivation for the first sketches were the constraints of memory. Bloom filters were proposed as a compact way to perform spell checking when it was not feasible to keep a full dictionary in memory [9]. Similarly, the Munro-Paterson work on quantiles was in the context of tape-resident data sets which were too large to be brought into memory. The advent of hierarchical memory systems with larger main memory and ready access to (relatively) fast disks diminished the need for such algorithms during the subsequent decades. However, the emergence of systems performing analytics on very large volumes of data meant that these applications did not disappear entirely.

Massive Data Streams (mid 1990s—early 2000s). The growth of the internet, and associated ecosystems, provided the setting for "massive data" and "data streams". Initially, the focus came from the network/ISP world: for the first time, we could easily see examples where the volume of data moving through a system dwarfed the capacity to store it at rest. Although the (meta)data was mostly ephemeral, it was desirable to be able to summarize and query it in order to monitor and debug networked systems. This drove the demand for sketches that could be build in a streaming (incremental) fashion, and integrated into special-purpose data stream management systems. These included systems from Sprint (CMON [46]) and AT&T (Gigascope [16]) in industry, and academic systems such as Stream, Aurora and Borealis [1, 2, 6, 49]. Here, the need was often not to build one sketch, but to maintain huge numbers of sketches in parallel (i.e., to support GROUP BY aggregate queries over many groups). While impactful within their specialist domains, these applications tended to be internal and bespoke to specific network management problems. Attempts to generalize these ideas to distributed models, captured in settings such as sensor networks, provided rich fodder for research papers, but had more limited practical impact.

From ISPs to Internet Companies (2000s onwards). A shift in the motivation for sketch algorithms came in the first decade of the current century, when a new class of Internet-based companies came along with a focus on novel technologies. Starting with search engines, these companies handled vast amounts of data, and hence brought applications that could benefit from sketching. Google was the leading example here, and several sketches found important motivation from search data: the Count sketch was proposed by academic visitors to Google [12], while locality sensitive hashing was built into systems to perform multimedia (image) search. Even though many of the technologies have changed over the years, sketching still has relevance to these applications. For instance, the mechanism for image similarity search may have shifted from simple feature extraction to learned vector embeddings. However, both rely on notions of (high-dimensional) vector similarity which can be supported efficiently by LSH-based techniques.

Online advertising (2010s). The financial underpinning for these new tech companies primarily derived from online advertising: connecting internet users with adverts to draw their clicks. A basic question that advertisers wanted to understand was exactly how many individuals were their adverts reaching? This could be a non-trivial question, due to the complexity and scale of the online advertising ecosystem that quickly grew up. Sketches, specifically distinct count sketches such as loglog and hyperloglog, were proposed as an answer: these sketches could be used to track how many distinct users (based on cookie information) were exposed to

a particular campaign, while avoiding double counting. Properties of these sketches meant that it was possible to "slice and dice" these statistics, by reporting response rates across multiple dimensions (e.g., demographic attributes). Systems were built and put into production based on this principle, by companies such as Aggregate Knowledge. However, there were obstacles that prevented this approach having a major impact. First, a long-standing limitation of sketches that use randomization is the challenge in communicating a randomized approximation guarantee to non-technical consumers. This is not unique to sketching, and can be overcome with appropriate communication tools (e.g., confidence intervals on reported statistics), but presents an initial barrier. A more fundamental issue is that computer systems eventually scaled faster than advertising clicks: it became possible to track and process advertising information in highly performant data warehouses, giving "exact" results (up to sampling bias and other noise factors). While there remain cases where the data volume is very high (e.g., systems that track every tiny interaction, such as a mouse movement) that could benefit from the use of sketching, these may instead be handled by alternative downsampling techniques to reduce the data down to more managable amounts.

The Big Data era (2010s onwards). Meanwhile, the terminology shifted from 'massive data' to 'big data', and new applications emerged. Other applications, such as social media and video streaming, became mainstream, and brought their own data analytics questions with them. While the primary data (posts, videos etc.) have to be stored and delivered exactly, there are large volumes of secondary data that can be summarized in sketches. For example, Twitter used count-min sketches to keep track of how many views were received by "embedded tweets", such as a tweet that is presented within a news article. New algorithms for the core problems of heavy hitters, quantiles, and count distinct were developed (e.g., the KLL algorithm, the t-digest summary) and made available via libraries (the Apache Data Sketches Library) and platforms (e.g., Splunk and Salesforce). While it remains challenging to find detailed information on the extent to which sketches have been deployed in practice, anecdotally some sketches are very widely used, and many software engineers sing the praises of sketches such as Bloom filters and hyperloglog[2].

Private Data Analysis (late 2010s onwards). As the focus on data analysis has grown over time, so has the need for privacy enhancing technologies to support it. The data being analyzed is often related to individual people, and so it is necessary to modify the analysis procedure in order to protect the privacy of the individuals who have contributed to the data. Formal definitions of privacy have emerged in the form of k-anonymity [43] and differential privacy [19], which require that the data analysis procedure adheres to some requirements, such as coarsening the level of information available, or adding calibrated random noise to the output. Such definitions happen to cohere well with sketching: the compact representations formed by sketch algorithms tend to mix and concentrate the information from many individuals, making the perturbations due to privacy less disruptive than other representations would be [50].

A concrete example is the RAPPOR system deployed by Google to collect statistics on web browsing activity [20]. The system can be summarized as combining the Bloom filter summary [9] with randomized response [47], to randomly flip some of the bits. Similarly, Apple's deployment of differential privacy can be understood as taking a Count-Min sketch of a sparse input and applying randomized response to each entry [45]. More generally, the emerging area of Federated Analytics [8], which aims to collect data privately from a large population of distributed individuals can be crudely described as being based on sketches with privacy.

Optimizing Machine Learning (mid 2010s onwards). The vast growth in interest in machine learning over the last decade has drawn on many aspects of computer science: optimization to train models from data, hardware integration to speed up training at scale, and so one. One direction of interest has been on using sketches to reduce the cost of the training process. A basic idea is to make use of sketches that preserve the norm of data in high-dimensional space to perform the learning in the sketch space, rather than in the original space. This has been leveraged to reduce the communication cost of distributed machine learning [42]. Other potential directions include using sketching as a way to approximate expensive linear algebra operations, such as matrix multiplication, and to incorporate kernel transformations [40, 48]. To the best of my knowledge, such uses of sketches have so far been primarily of academic interest, but it is feasible that future work can more directly benefit from sketch techniques.

4 LESSONS LEARNED FROM SKETCHES IN PRACTICE

In studying and working with sketches over many years, and being excited about their potential for adoption in a wide range of applications, it is natural to have considered a number of ways to spread this enthusiasm, and to accelerate the pathway to adoption. This section reviews some of the different strategies that have been tried, and comments on their efficacy in this regard.

Launching a startup. The most direct way of pushing ideas from research into practice is to do it yourself: to launch a company around your latest paper. This idea has been suggested many times for sketches, given the powerful results that they can achieve. But, at the risk of stating the obvious, a successful startup needs a business idea: a product or service that can succeed in the marketplace. Sketches, like many ideas from the data management and algorithms worlds, are too far "under the hood" for a clear case to emerge: they don't obviously solve a problem that was impossible before, or tackle an issue that is a pain point for many users. Profitable companies that have come from academic research (e.g., Google with web search, and Akamai with consistent hashing) have ultimately succeeded due to having a business model that is supported by the technology (i.e., online advertising, and content delivery networks). So while sketching may be a useful tool in

[2]For instance, see https://medium.com/system-design-blog/bloom-filter-a-probabilistic-data-structure-12e4e5cf0638 https://pedrorijo.com/blog/bloom-filters/ https://www.ombulabs.com/blog/systemdesign/ruby/bloom-filter-and-what-makes-them-special.html https://engineering.fb.com/2018/12/13/data-infrastructure/hyperloglog/ http://content.research.neustar.biz/blog/hll.html https://redis.com/blog/count-min-sketch-the-art-and-science-of-estimating-stuff/ https://florian.github.io/count-min-sketch/

building software that is of use to people, it is not (yet) a piece that is vital to the success of a product, and so is hard to build a business around.

Pushing out code. A more direct route to getting research ideas into use is to provide code to implement them. Sketches are a particularly good test case for this: the algorithms needed are often relatively simple to code up for a researcher. But the concepts and techniques may be sufficiently unfamiliar to the typical software engineer (such as certain kinds of hash function, or fiddly bit manipulation tricks) that prototype code can be very valuable, simply for showing a proof of concept. A reference implementation, even if crudely written and lightly documented, is much preferable and more tangible than pseudocode in a paper. So it is strongly encouraged for researchers to make their code available to others, via github or other forms.

Inflict ideas on the next generation. The computer science curriculum is far more dynamic than, say, the mathematics curriculum, and it is still feasible to include research ideas in undergraduate classes. There may not be room to go in-depth on cutting edge ideas, but including a few results from the current century may help to keep students engaged. Sketches are a good exemplar for this, since the ideas can fit well into an algorithms or database class, and illustrate some of the underlying principles and concepts. A long-term benefit of this approach is that some students may just remember these ideas after graduation, and be motivated to make use of them in whatever career they choose to go in to.

Write accessible notes, and put them where people can read them. While we may think of peer-reviewed academic publications as the medium for sharing research ideas, these are unfortunately not the place where practitioners will find them. You can have more reach by writing accessible notes addressing the software engineering community. For sketches, we made web pages and wrote articles in practitioner-focused journals. Today, you should consider making more use of platforms like medium and substack, and promoting posts via social media.

Give talks and tutorials. On a similar note, talks can be more accessible than articles, particularly if they are captured as an online video that is easy to share. For some people, their first stop to learn about a new topic is on YouTube, rather than arXiv. Things don't have to be too polished, but there is potential for short-form, custom made introductory videos to reach a wide audience interested in learning new techniques that they can adopt.

Work directly with companies. Lastly, it can be valuable to work directly with a company that can benefit from translating research into practice. This is not a simple proposition: working very closely with a company requires building up a lot of trust and understanding, or going through a demanding application and recruitment process. Most organizations want a solution to their problems that can be implemented and deployed at scale. This does not necessarily align with research novelty: sometimes simple partial solutions will suffice, while research-inspired approaches are viewed as too complex and not scalable enough. However, the experience can be eye-opening, and can allow not only real-world use of research ideas, but inspiration for new research questions that are well-motivated.

5 CONCLUDING REMARKS

The notion of sketching is a compelling one, to build a compact representation of a large dataset that nonetheless allows certain properties of the data to be accurately approximated. Contributions to this topic have required substantial theoretical advances, appearing in venues such as PODS, but promising substantial impact in practice.

The road to genuine impact is a long and bumpy one. Many great theoretical ideas never make any significant impact, because the scenario they solve does not arise in practice as urgently, or can be tackled satisfactorily with heuristic or less efficient approaches. Sketches can rightly claim to have had meaningful impact on the practice of computer science. However, the motivations for sketches have changed over time, and once compelling demands may no longer be relevant. Moreover, of the hundreds, if not thousands, of papers that have presented ideas on sketches, there may be only a handful that have achieved very widespread use. Bloom filters and Hyperloglog sketches are the most well-known, along with Count sketches and Count-Min sketches, plus various sketches for quantile and frequent item estimation (SpaceSaving, Misra-Gries, Q-Digest, T-Digest, Greenwald-Khanna).

This illustrates the principle that good theory can lead to good impact. The biggest contributor to this may be time: it can take time for ideas to diffuse and to find their application. After this, active effort to connect the ideas with their potential beneficiaries is useful – make the ideas as easy as possible to access and digest. Finally, there is always some element of luck – will the right motivation align with the idea's potential, and will the right people be inspired by the ideas to put them into practice?

Acknowledgements. Thanks to Divesh Srivastava and S. Muthukrishnan for excellent feedback and encouragement, for this article and also throughout my career.

REFERENCES

[1] D. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Rasin, N. Tatbul, Y. Xing, and S. Zdonik. 2005. Distributed Operation in the Borealis Stream Processing Engine. In *ACM SIGMOD International Conference on Management of Data*.

[2] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Xing, R. Yan, and S. Zdonik. 2003. Aurora: a data stream management system. In *ACM SIGMOD International Conference on Management of Data*. 666.

[3] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zheiwei Wei, and Ke Yi. 2013. Mergeable Summaries. *ACM Transactions on Database Systems* 38, 4 (2013).

[4] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Analyzing graph structure via linear measurements. In *ACM-SIAM Symposium on Discrete Algorithms*.

[5] N. Alon, Y. Matias, and M. Szegedy. 1996. The Space Complexity of Approximating the Frequency Moments. In *ACM Symposium on Theory of Computing*. 20–29.

[6] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J; Widom. 2003. STREAM: the Stanford Stream Data Manager (demonstration description). In *ACM SIGMOD International Conference on Management of Data*. 665–665.

[7] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. 2020. A Framework for Adversarially Robust Streaming Algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, Dan Suciu, Yufei Tao, and Zhewei Wei (Eds.). ACM, 63–80. https://doi.org/10.1145/3375395.3387658

[8] Akash Bharadwaj and Graham Cormode. 2022. An Introduction to Federated Computation. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 2448–2451. https://doi.org/10.1145/3514221.3522561

[9] Burton Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (July 1970), 422–426.

[10] Avrim Blum, John Hopcroft, and Ravi Kannan. 2020. *Foundations of Data Science.* Cambridge University Press, Cambridge. https://www.cambridge.org/core/books/foundations-of-data-science/6A43CE830DE83BED6CC5171E62B0AA9E

[11] B. Boyer and J. Moore. 1981. *A Fast Majority Vote Algorithm.* Technical Report ICSCA-CMP-32. Institute for Computer Science, University of Texas.

[12] M. Charikar, K. Chen, and M. Farach-Colton. 2002. Finding Frequent Items in Data Streams. In *Procedings of the International Colloquium on Automata, Languages and Programming (ICALP).*

[13] Graham Cormode, Zohar S. Karnin, Edo Liberty, Justin Thaler, and Pavel Veselý. 2021. Relative Error Streaming Quantiles. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo (Eds.). ACM, 96–108. https://doi.org/10.1145/3452021.3458323

[14] G. Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-Min Sketch and its Applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[15] Graham Cormode and Ke Yi. 2020. *Small Summaries for Big Data.* Cambridge University Press.

[16] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. 2003. Gigascope: A Stream Database for Network Applications. In *ACM SIGMOD International Conference on Management of Data.*

[17] D. Donoho. 2004. Compressed Sensing. http://www-stat.stanford.edu/~donoho/Reports/2004/CompressedSensing091604.pdf. Unpublished Manuscript.

[18] Marianne Durand and Philippe Flajolet. 2003. Loglog Counting of Large Cardinalities (Extended Abstract). In *European Symposium on Algorithms (ESA).* 605–617.

[19] Cynthia Dwork. 2006. Differential privacy. In *ICALP.* 1–12.

[20] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Computer and Communications Security.* 1054–1067.

[21] Phillipe Flajolet, Eric Fusy, O. Gandouet, and F. Meunier. 2007. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *Analysis of Algorithms (AOFA).* 127–146.

[22] P. Flajolet and G. N. Martin. 1983. Probabilistic Counting. In *IEEE Conference on Foundations of Computer Science.* 76–82. Journal version in *Journal of Computer and System Sciences*, 31:182–209, 1985.

[23] M. Greenwald and S. Khanna. 2001. Space-efficient online computation of quantile summaries. In *ACM SIGMOD International Conference on Management of Data.*

[24] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *International Conference on Extending Database Technology.* 683–692.

[25] P. Indyk and R. Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *ACM Symposium on Theory of Computing.* 604–613.

[26] W.B. Johnson and J. Lindenstrauss. 1984. Extensions of Lipshitz mapping into Hilbert space. *Contemp. Math.* 26 (1984), 189–206.

[27] Hossein Jowhari, Mert Saglam, and Gábor Tardos. 2011. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *ACM Principles of Database Systems.*

[28] Daniel M. Kane and Jelani Nelson. 2012. Sparser Johnson-Lindenstrauss Transforms. In *ACM-SIAM Symposium on Discrete Algorithms.*

[29] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, Jan Paredaens and Dirk Van Gucht (Eds.). ACM, 41–52. https://doi.org/10.1145/1807085.1807094

[30] Zohar Karnin, Kevin Lang, and Edo Liberty. 2016. Optimal Quantile Approximation in Streams. In *IEEE Conference on Foundations of Computer Science.*

[31] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining of Massive Datasets.* Cambridge University Press. http://mmds.org

[32] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. 1998. Approximate Medians and other Quantiles in One Pass and with Limited Memory. In *ACM SIGMOD International Conference on Management of Data.* 426–435.

[33] Dzejla Medjedovic, Emin Tahirovic, and Ines Dedovic. 2022. *Algorithms and Data Structures for Massive Datasets.* Manning.

[34] A. Metwally, D. Agrawal, and A. El Abbadi. 2005. Efficient computation of Frequent and Top-k Elements in Data Streams. In *International Conference on Database Theory.*

[35] J. Misra and D. Gries. 1982. Finding Repeated Elements. *Science of Computer Programming* 2 (1982), 143–152.

[36] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press.

[37] Robert Morris. 1977. Counting large numbers of events in small registers. *Commun. ACM* 21, 10 (1977), 840–842.

[38] J. I. Munro and M. S. Paterson. 1980. Selection and Sorting with Limited Storage. *Theoretical Computer Science* 12 (1980), 315–323.

[39] Jelani Nelson and Huacheng Yu. 2022. Optimal Bounds for Approximate Counting. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Leonid Libkin and Pablo Barceló (Eds.). ACM, 119–127. https://doi.org/10.1145/3517804.3526225

[40] Ninh Pham and Rasmus Pagh. 2013. Fast and Scalable Polynomial Kernels via Explicit Feature Maps. In *ACM SIGKDD* (Chicago, Illinois, USA). 239–247. https://doi.org/10.1145/2487575.2487591

[41] Arik Rinberg, Alexander Spiegelman, Edward Bortnikov, Eshcar Hillel, Idit Keidar, Lee Rhodes, and Hadar Serviansky. 2022. Fast Concurrent Data Sketches. *ACM Trans. Parallel Comput.* 9, 2 (2022), 6:1–6:35. https://doi.org/10.1145/3512758

[42] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. FetchSGD: Communication-Efficient Federated Learning with Sketching. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119).* PMLR, 8253–8265. http://proceedings.mlr.press/v119/rothchild20a.html

[43] Pierangela Samarati and Latanya Sweeney. 1998. *Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression.* Technical Report SRI-CSL-98-04. SRI.

[44] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. 2004. Medians and Beyond: New Aggregation Techniques for Sensor Networks. In *ACM SenSys.*

[45] Differential Privacy Team. 2017. Learning with Privacy at Scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf. *Apple Machine Learning Journal* 1, 8 (Dec. 2017).

[46] K. To, T. Ye, and S. Bhattacharyya. 2004. *CMON: A general purpose continuous IP backbone traffic analysis platform.* Technical Report RR04-ATL-110309. Sprint ATL.

[47] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69. https://doi.org/10.1080/01621459.1965.10480775 PMID: 12261830.

[48] David P. Woodruff. 2014. Sketching As a Tool for Numerical Linear Algebra. *Found. Trends Theor. Comput. Sci.* 10, 1–2 (Oct. 2014), 1–157. https://doi.org/10.1561/0400000060

[49] S. Zdonik, M. Stonebraker, M. Cherniack, and U. Cetintemel. 2003. The Aurora and Medusa Projects. *Bulletin of the Technical Committee on Data Engineering* (March 2003), 3–10.

[50] Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. 2022. Differentially Private Linear Sketches: Efficient Implementations and Applications. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 12691–12704. https://proceedings.neurips.cc/paper_files/paper/2022/file/525338e0d98401a62950bc7c454eb83d-Paper-Conference.pdf