SIGMOD Officers, Committees, and Awardees

Chair
Juliana Freire
Computer Science & Engineering
New York University
Brooklyn, New York
USA
+1 646 997 4128
juliana.freire <at> nyu.edu

Vice-Chair
Ihab Francis Ilyas
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario
CANADA
+1 519 888 4567 ext. 33145
ilyas <at> uwaterloo.ca

Secretary/Treasurer
Fatma Ozcan
IBM Research
Almaden Research Center
San Jose, California
USA
+1 408 927 2737
fozcan <at> us.ibm.com

SIGMOD Executive Committee:
Juliana Freire (Chair), Ihab Francis Ilyas (Vice-Chair), Fatma Ozcan (Treasurer), K. Selçuk Candan, Yanlei Diao, Curtis Dyreson, Christian S. Jensen, Donald Kossmann, and Dan Suciu.

Advisory Board:
Yannis Ioannidis (Chair), Phil Bernstein, Surajit Chaudhuri, Rakesh Agrawal, Joe Hellerstein, Mike Franklin, Laura Haas, Renee Miller, John Wilkes, Chris Olsten, AnHai Doan, Tamer Özsu, Gerhard Weikum, Stefano Ceri, Beng Chin Ooi, Timos Sellis, Sunita Sarawagi, Stratos Idreos, Tim Kraska

SIGMOD Information Director:
Curtis Dyreson, Utah State University

Associate Information Directors:
Huiping Cao, Georgia Koutrika, Wim Martens, Paris Koutris, Asterios Katsifodimos

SIGMOD Record Editor-in-Chief and Associate Editor-in-Chief:
Yanlei Diao (EiC), Rada Chirkova (Associate EiC)

SIGMOD Record Associate Editors:
Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Zachary Ives, Anastasios Kementsietsidis, Frank Neven, Olga Papaemmanouil, Aditya Parameswaran, Alkis Simitsis, Pinar Tözün, Marianne Winslett, and Jun Yang

SIGMOD Conference Coordinator:
K. Selçuk Candan, Arizona State University

PODS Executive Committee:
Dan Suciu (Chair), Tova Milo, Diego Calvanese, Wang-Chiew Tan, Rick Hull, Floris Geerts

Sister Society Liaisons:
Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment), Christian Jensen (IEEE TKDE)

SIGMOD Awards Committee:
Martin Kersten (Chair), Surajit Chadhuri, David DeWitt, Sunita Sarawagi, Michael Carey

Jim Gray Doctoral Dissertation Award Committee:
Ioana Manolescu (co-Chair), Lucian Popa (co-Chair), Peter Bailis, Michael Cafarella, Feifei Li, Qiong Luo, Felix Naumann, Pinar Tozun

SIGMOD Systems Award Committee:
Michael Cafarella (Chair), Michael Carey, David DeWitt, Yanlei Diao, Paul Larson, Gustavo Alonso

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Recipients of the award are the following:

Michael Stonebraker (1992)
David DeWitt (1995)
Serge Abiteboul (1998)
Rudolf Bayer (2001)
Ronald Fagin (2004)
Jennifer Widom (2007)
Umeshwar Dayal (2010)
Stefano Ceri (2013)
Gerhard Weikum (2016)
Anastasia Ailamaki (2019)

Jim Gray (1993)
C. Mohan (1996)
Hector Garcia-Molina (1999)
Patricia Selinger (2002)
Michael Carey (2005)
Moshe Y. Vardi (2008)
Surajit Chaudhuri (2011)
Martin Kersten (2014)
Goetz Graefe (2017)

Philip Bernstein (1994)
David Maier (1997)
Rakesh Agrawal (2000)
Don Chamberlin (2003)
Jeffrey D. Ullman (2006)
Masaru Kitsuregawa (2009)
Bruce Lindsay (2012)
Laura Haas (2015)
Raghu Ramakrishnan (2018)

SIGMOD Systems Award

For technical contributions that have had significant impact on the theory or practice of large-scale data management systems.

Michael Stonebraker and Lawrence Rowe (2015); Martin Kersten (2016); Richard Hipp (2017); Jeff Hammerbacher, Ashish Thusoo, Joydeep Sen Sarma; Christopher Olston, Benjamin Reed, Utkarsh Srivastava (2018); Xiaofeng Bao, Charlie Bell, Murali Brahmadesam, James Corey, Neal Fachan, Raju Gulabani, Anurag Gupta, Kamal Gupta, James Hamilton, Andy Jassy, Tengiz Kharatishvili, Sailesh Krishnamurthy, Yan Leshinsky, Lon Lundgren, Pradeep Madhavarapu, Sandor Maurice, Grant McAlister, Sam McKelvie, Raman Mittal, Debanjan Saha, Swami Sivasubramanian, Stefano Stefani, Alex Verbitski (2019)

SIGMOD Contributions Award

For significant contributions to the field of database systems through research funding, education, and professional services. Recipients of the award are the following:

Maria Zemankova (1992)
Jeffrey Ullman (1996)
Raghu Ramakrishnan (1999)
Daniel Rosenkrantz (2001)
Surajit Chaudhuri (2004)
Hans-Jörg Schek (2007)
David Lomet (2010)
H.V. Jagadish (2013)
Samuel Madden (2016)
Ahmed Elmagarmid (2019)

Gio Wiederhold (1995)
Avi Silberschatz (1997)
Michael Carey (2000)
Richard Snodgrass (2002)
Hongjun Lu (2005)
Klaus R. Dittrich (2008)
Gerhard Weikum (2011)
Kyu-Young Whang (2014)
Yannis E. Ioannidis (2017)

Yahiko Kambayashi (1995)
Won Kim (1998)
Laura Haas (2000)
Michael Ley (2003)
Tamer Özsu (2006)
Beng Chin Ooi (2009)
Marianne Winslett (2012)
Curtis Dyreson (2015)
Z. Meral Özsoyoğlu (2018)

SIGMOD Jim Gray Doctoral Dissertation Award

SIGMOD has established the annual SIGMOD Jim Gray Doctoral Dissertation Award to *recognize excellent research by doctoral candidates in the database field.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau. *Honorable Mentions*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.
- **2014** *Winners*: Aditya Parameswaran and Andy Pavlo.
- **2015** *Winner*: Alexander Thomson. *Honorable Mentions*: Marina Drosou and Karthik Ramachandra
- **2016** *Winner*: Paris Koutris. *Honorable Mentions*: Pinar Tozun and Alvin Cheung

- **2017** *Winne*r: Peter Bailis. *Honorable Mention*: Immanuel Trummer
- **2018** *Winne*r: Viktor Leis. *Honorable Mention*: Luis Galárraga and Yongjoo Park
- **2019** *Winne*r: Joy Arulraj. *Honorable Mention*: Bas Ketsman

A complete list of all SIGMOD Awards is available at: **https://sigmod.org/sigmod-awards/**

[Last updated: June 30, 2019]

Editor's Notes

Welcome to the June 2019 issue of the ACM SIGMOD Record!

This issue starts with the Database Principles column featuring an article by Rahul and Tao on designing top-k indexes. The design goal is to address the problem of retrieving only the best k records, in those cases where users do not need to examine all the answers. Clearly, focusing on retrieving only up to a predefined number of the highest-ranked query answers has the potential to improve the efficiency of query processing. In the space of potential mechanisms for preventing the query processor from accessing the lower-ranked answers, the article guides the reader through the state of the art on designing top-k indexes that provide strong theoretical guarantees for both time and space, and are also efficiently updatable. Powerful results detailed in the article show that the problem of designing top-k indexes is no harder than certain natural related problems. For practitioners, this implies that solutions for those related problems can be used as black boxes in the design of top-k indexes with desired expected complexity. The article also discusses open problems, and provides references on related areas.

The Surveys column features an article by Pierri and Ceri that discusses the issue of false news on social media. The article provides a comprehensive study of recent algorithmic advances in detecting, characterizing, and mitigating false news on popular social-media platforms, and also discusses emerging approaches. The discussion centers on 2017-18 results, and includes detailed comparative descriptions of the most promising ideas, methods, and approaches. The authors also outline potential interventions, while also bringing into the picture ethical concerns about censorship. The article provides an extensive bibliography, as well as pointers to, and detailed comparative descriptions of, reference data sets.

The Systems and Prototypes column features an article by Scherzinger, which reports on a course titled "Modern database concepts" taught in summer 2018 at OTH Regensburg. The goals of the intensive course included teaching ideas behind engines such as Hive, as well as design decisions regarding query-language constructs. The article focuses on the hands-on project that was offered as an option to the students in the course. The project objective was to build miniHive, an SQL-on-Hadoop engine for compiling SQL queries into MapReduce workflows. The article describes the scope of the project milestones, the required self-study materials, and some coding challenges. It concludes with observations about the student outcomes as a result of their having taken the project; the discussion is based on the students' self reporting, as well as on their performance in the course. For those interested in the details, complete course materials can be made available by the author on request.

The Distinguished Profiles column features Richard Hipp, winner of the 2017 SIGMOD Systems Award and of the 2005 Google O'Reilly Open Source Award for SQLite. Richard has his own consulting firm, Hwaci; his Ph.D. is from Duke University. In this interview, Richard talks about SQLite, the most widely deployed database engine in the world. He discusses the reasons for the popularity of SQLite, the technical challenges in creating it, the aviation-grade testing involved in the design process, and the ongoing projects. He also shares why working on SQLite has been a dream job, and what things he could work on in the future. Richard discusses potential advantages of consulting careers for graduates, and gives advice on topics that database researchers could find rewarding to work on.

The Reports column features an article by Ailamaki and colleagues that describes the submission-evaluation process for the SIGMOD 2019 research track. The article discusses actionable goals concerning reviews and submissions, and describes in detail the infrastructure deployed to address those. In the SIGMOD 2019 research track, many remarkable things were done to scale the submission-evaluation work while ensuring fairness, including ingenious use of features of the chosen tools, as well as optimization approaches based on integer programming for assigning reviewers to submissions. The article discusses the submission-evaluation pipeline and provides details on the specifics of the needed tuning, on the experience of working with the chosen tools, and on the feedback from authors. The article goes on to discuss the success metrics, summarizes the overall rewarding outcomes, and also suggests some points for improvement.

On behalf of the SIGMOD Record Editorial board, we hope that you enjoy reading the June 2019 issue of the SIGMOD Record!

Your submissions to the SIGMOD Record are welcome via the submission site:
http://sigmod.hosting.acm.org/record

Prior to submission, please read the Editorial Policy on the SIGMOD Record's website:
https://sigmodrecord.org/sigmod-record-editorial-policy/

Yanlei Diao and Rada Chirkova

June 2019

Past SIGMOD Record Editors:

Ioana Manolescu (2009-2013)
Ling Liu (2000–2004)
Arie Segev (1989–1995)
Thomas J. Cook (1981–1983)
Daniel O'Connell (1971–1973)
Alexandros Labrinidis (2007–2009)
Michael Franklin (1996–2000)
Margaret H. Dunham (1986–1988)
Douglas S. Kerr (1976-1978)
Harrison R. Morse (1969)
Mario Nascimento (2005–2007)
Jennifer Widom (1995–1996)
Jon D. Clark (1984–1985)
Randall Rustin (1974-1975)

A Guide to Designing Top-k Indexes

Saladi Rahul
University of Illinois Urbana-Champaign
USA
saladi.rahul@gmail.com

Yufei Tao
Chinese University of Hong Kong
Hong Kong
taoyf@cse.cuhk.edu.hk

ABSTRACT

Top-k search, which reports the k elements of the highest importance from all the elements in an underlying dataset that satisfy a certain predicate, has attracted significant attention from the database community. The search efficiency crucially depends on the quality of an index structure that can be utilized to filter the underlying data by both the user-specified predicate and the ranking of importance. This article introduces the reader to a list of techniques for designing such indexes with strong performance guarantees. Several promising directions for future work are also discussed.

1. INTRODUCTION

Interactive exploration of a database system is often hampered by the fact that a query may return a set of records that is excessively large for a user to examine. On the other hand, a user would rarely be interested in all the records satisfying her/his query predicate q. In many situations, what matters most is the subset that contains only the k records — for a small integer k — in the result with the highest "importance", where importance is measured by an appropriate ranking function. For example, while the query "find all the creditcard transactions of today" can return millions of records, what a bank manager would actually like to do could be just to scrutinize the 100 transactions with the largest payments. Retrieving only the k best records is commonly known as *top-k search*, and becomes increasingly useful as database volumes continue to grow at a rapid pace.

Besides controlling the output size, top-k search also brings opportunities to boost the efficiency of query processing because returning only k records can potentially be significantly faster than fetching the full result. This requires a mechanism that can be deployed to avoid accessing the records that satisfy the search predicate q but do not have sufficiently high importance to be reported. Designing such mechanisms has been a major research topic in the database area during the past two decades (see, e.g., [1, 7–9, 19, 25–27, 30, 31] and the references therein). At the core of almost every mechanism is an index structure — henceforth referred to as a *top-k index* — which stores certain information pre-computed from the underlying data that can lead the query algorithm to finding the k target records efficiently. Unlike conventional access methods in a database system, a top-k index must allow a query to filter the data not only by the predicate q, but also by the ranking of importance.

In this article, we introduce the reader to a suite of representative techniques that have been proposed in the literature for designing top-k indexes. Our discussion will focus on obtaining indexes with strong theoretical guarantees, and therefore, will not be concerned with methods that are designed purely for empirical evaluation. We will also point out some directions that call for further research efforts on this fascinating topic.

The content of this article has little overlap with *Fagin's algorithm* [14] and the *threshold algorithm* by Fagin, Lotem, and Naor [15] which were developed for a class of problems on *distributed computation* which are sometimes referred to also under the name "top-k". As surveyed in [19], there have been multiple attempts to apply the algorithms of [14, 15] to answer top-k queries in centralized systems. However, none of those attempts has succeeded in attaining performance guarantees that are interesting through the lens of this article. The techniques we will describe all follow ideas different from those of [14, 15].

2. PROBLEM DEFINITIONS

In Section 2.1, we provide a problem formulation that encapsulates a broad class of top-k problems. Section 2.2 gives two representative problems that will be utilized to demonstrate the techniques to be discussed. Section 2.3 clarifies the computation model to be adopted, and the performance guarantees to be achieved.

2.1 Generic Query Formulation

Let $\mathbb{D}$ be an arbitrary set which serves as the data domain. Denote by $\mathbb{Q}$ a set of *predicates* that can be applied to the elements of $\mathbb{D}$. Specifically, given a predicate $q \in \mathbb{Q}$, we can evaluate q on every element $e \in \mathbb{D}$ to

obtain a boolean value 1 or 0; in the former case e is said to *satisfy* q, while in the latter e does not. We assume that $\mathbb{Q}$ has a special predicate `true`, which evaluates to 1 for all $e \in \mathbb{D}$.

The input dataset D is a subset of $\mathbb{D}$. For each predicate $q \in \mathbb{Q}$, define $q(D)$ to be the set of elements in D satisfying q. Given a predicate $q \in \mathbb{Q}$, a *reporting query* returns $q(D)$ in its entirety.

As mentioned earlier, a user is often interested only in the most "important" elements of $q(D)$. We formalize importance by resorting to a weight function $w : \mathbb{D} \to \mathbb{R}$, where $\mathbb{R}$ represents the set of real numbers. For each element $e \in \mathbb{D}$, we refer to the value $w(e)$ as the *weight* of e under w. Denote by $\mathbb{W}$ a non-empty set of such weight functions. Every reporting query has a top-k counterpart:

> Given a predicate $q \in \mathbb{Q}$, a weight function $w \in \mathbb{W}$, and an integer $k \geq 1$, a *top-k query* reports the k elements in $q(D)$ with the highest weights under w. Specially, if $|q(D)| < k$, then the entire $q(D)$ is reported.

If two elements have the same weight, we assume that the tie is broken by a consistent policy, e.g., regarding the element with a larger id to have a greater weight. Note that a top-1 query, which will be referred to as a *max query*, returns the element in $q(D)$ with the maximum weight.

Our exploration into the theory of top-k queries will encounter frequently another variant of reporting queries:

> Given a predicate $q \in \mathbb{Q}$, a weight function $w \in \mathbb{W}$, and a real value τ, a *prioritized query* reports all the elements $e \in q(D)$ with $w(e) \geq \tau$.

In general, the set $\mathbb{W}$ may contain an arbitrarily large number of functions. In the other extreme, $\mathbb{W}$ may include only a *single* function. In that case, obviously, all the top-k and prioritized queries must choose the same, unique, function w in $\mathbb{W}$, such that we can as well regard the weight $w(e)$ directly as an attribute associated with each element $e \in D$. When this happens, we will refer to D as a *weight-augmented* dataset.

2.2 Two Specialized Instances

Next, we specialize the above formulation into two concrete instances that are representative for several reasons. First, the top-k queries in both instances are important top-k problems that have been extensively studied. Second, their specialization is based on drastically different choices of $\mathbb{D}$, $\mathbb{Q}$, and $\mathbb{W}$, thereby demonstrating the generality of our query formulation. Third, they are among the simplest instances suitable for illustrating the techniques to be presented in later sections.

Problem 1: Linear Ranking. In this instance:

- $\mathbb{D} = \mathbb{R}^d$, where d is a positive constant integer;
- $\mathbb{Q} = \{\texttt{true}\}$, namely, $\mathbb{Q}$ has only a single predicate that always evaluates to 1;
- $\mathbb{W}$ is the set of linear functions which are defined by d real values $c_1, ..., c_d$, and map a point $p = (x_1, x_2, ..., x_d) \in \mathbb{D}$ to $\sum_{i=1}^{d} c_i x_i$.

Equivalently, a dataset D is a set of points in the d-dimensional space. Given the coefficients $c_1, ..., c_d$, a top-k query reports the k points $p = (x_1, ..., x_d) \in D$ that maximize $\sum_{i=1}^{d} c_i x_i$. Such queries constitute the top-k problem that has received by far the most attention from the *system* community (see [19] for a survey). As a classic application, consider that $d = 2$, and each point $e \in D$ captures the `price` and `rating` of a hotel as the x- and y-coordinates, respectively. A top-10 query finds the 10 best hotels maximizing $c_1 \cdot (-\texttt{price}) + c_2 \cdot \texttt{rating}$, where c_1 and c_2 are coefficients chosen by a user, reflecting her/his personal weighting on the two attributes.

Problem 2: One-Dimensional Range Searching. This is an instance on weight-augmented datasets:

- $\mathbb{D} = \mathbb{R}$;
- $\mathbb{Q}$ consists of all such predicates, each of which specifies an interval q in $\mathbb{R}$, and evaluates to 1 on every point $e \in \mathbb{D} \cap q$, and to 0 on every point $e \notin \mathbb{D} \cap q$.

Equivalently, a dataset D is a set of points in $\mathbb{R}$, each of which is associated with a real-valued weight. Given an interval q, a top-k query reports the k points in $D \cap q$ with the highest weights. Such queries constitute the most extensively studied top-k problem in the *theory* community [1, 7, 8, 25, 30, 31]. For an example, consider a `TRANSACTION` table with attributes `id`, `date` and `payment`, on which a top-100 query is "find the 100 tuples with the highest `payment` values among those with `date` in [01/2019, 03/2019]".

2.3 Computation Model and Design Goals

Our discussion will assume the standard word-RAM model. We further assume that every element in $\mathbb{D}$ can be stored in $O(1)$ words, and so can the encoding of each predicate in $\mathbb{Q}$ (e.g., in 1D range searching, each predicate is specified by an interval, which can be encoded in two words).

Define $n = |D|$, i.e., the number of elements in the input dataset. Our primary objective is to preprocess D into a top-k index that consumes near-linear space, and can be used to solve top-k queries efficiently. This means that the index should use $\tilde{O}(n)$ space, and answer a top-k query in $\mathcal{Q}_{top}(n) + \tilde{O}(k)$ time where notation

$\tilde{O}$ hides an $O(\text{polylog}\, n)$ factor, and $\mathcal{Q}_{top}$ is a slow-growing function of n. Ideally, we would also like the index to be *dynamic*, namely, updatable in $\tilde{O}(1)$ time per insertion and deletion.

Finally, it is worth mentioning that the result of a top-k query may return k elements in an arbitrary order. A sorted order can be generated by trivially sorting those elements in $O(k \log k)$ time. Remember that the value of k is supplied by a query as a parameter, instead of being fixed in advance.

3. TOP-K IMPLIES PRIORITIZED

Top-k and prioritized queries represent two similar ways to trim the result of a reporting query. Recall that, given a predicate $q \in \mathbb{Q}$, a reporting query returns $|q(D)|$ elements. The corresponding top-k query limits the output size to at most k explicitly. The corresponding prioritized query, on the other hand, filters out the elements of $q(D)$ with weights less than τ, after which the number t of elements reported can be anywhere from 0 to $|q(D)|$.

There is a subtle but important difference between top-k and prioritized queries. For a top-k query, whether an element $e \in q(D)$ should be reported does *not* depend solely on e, because it is also affected by the weights of the other elements in $q(D)$. In contrast, for a prioritized query, whether e should be reported can be decided by looking at e *itself*. Intuitively, this suggests that top-k queries ought to be at least as hard as prioritized queries.

This intuition turns out to be correct, namely, if we can find a top-k index with a certain space-query tradeoff, we must be able to achieve the same tradeoff asymptotically for prioritized queries:

THEOREM 1. *Fix D, $\mathbb{Q}$, and $\mathbb{W}$, and set $n = |D|$. Suppose that there is a top-k index on D that consumes $\mathcal{S}_{top}(n)$ space, and answers a top-k query in $\mathcal{Q}_{top}(n) + O(k)$ time, where $\mathcal{Q}_{top}(n) > 0$ for all n. Then, there is a data structure on D that uses $\mathcal{S}_{top}(n)$ space, and answers a prioritized query in $O(\mathcal{Q}_{top}(n)+t)$ time, where t is the number of reported elements.*

PROOF. Let $\mathcal{T}$ be the top-k index on D as described in the theorem. Given a prioritized query with parameters $q \in \mathbb{Q}, w \in \mathbb{W}$, and $\tau \in \mathbb{R}$, we use $\mathcal{T}$ to answer it by executing multiple rounds as follows. In round j (starting with $j = 1$), we issue a top-k_j query on $\mathcal{T}$ with the same q and w by setting $k_j = 2^{j-1} \cdot \mathcal{Q}_{top}(n)$. Two cases may arise:

- If the top-k_j query returns exactly k_j elements, we scan them to find the element e with the smallest weight. If $w(e) < \tau$, we do not go to the next round; otherwise, round $j + 1$ is launched.
- If the top-k_j query returns less than k_j elements, no more rounds are performed.

Let i be the number of rounds executed. Among the elements reported by the top-k_i query, we remove those with weights less than τ, and return the rest of the elements as the output of the prioritized query.

The space consumption of $\mathcal{T}$ is clearly $\mathcal{S}_{top}(n)$. Now we analyze the query time. If only one round is executed, the time is bounded by $\mathcal{Q}_{top}(n)+O(k_1) = O(\mathcal{Q}_{top}(n))$, noticing that $k_1 = \mathcal{Q}_{top}(n)$. If $i \geq 2$ rounds are performed, the time is bounded by

$$O\Big(\sum_{j=1}^{i} \mathcal{Q}_{top}(n) + 2^{j-1} \cdot \mathcal{Q}_{top}(n)\Big) = O(2^i \cdot \mathcal{Q}_{top}(n)).$$

By the fact that the execution entered i-th round, we know that $t \geq 2^{i-2} \cdot \mathcal{Q}_{top}(n)$, which gives $2^i \cdot \mathcal{Q}_{top}(n) \leq 4t$. It thus follows that the total query time is $O(t)$. □

The above result, which was independently observed by the authors of [24, 25], has an interesting implication: attempts to design an effective top-k index should be carried out *after* one has succeeded in obtaining a structure capable of resolving a prioritized query with the desired space-query tradeoff. Indeed, the prioritized query serves as a good starting point to approach a top-k problem, which is a pattern that will show up repeatedly in the rest of this article.

4. A FRAMEWORK FOR DESIGNING TOP-K INDEXES

In this section, we will establish a powerful framework for obtaining top-k indexes that enjoy strong theoretical guarantees in expectation. This framework is remarkably easy to apply, and works for all top-k queries captured by the formulation in Section 2.1.

4.1 Equivalence between Top-k and the Combination of Prioritized and Max

We have seen in Section 3 that, to design a top-k index with a certain space-query tradeoff, one must be able to obtain a structure with the same tradeoff for the corresponding prioritized query. Another similar but more obvious fact is that any top-k index guaranteeing $\mathcal{Q}_{top}(n) + O(k)$ time must be able to answer the corresponding max (a.k.a. top-1) query in $O(\mathcal{Q}_{top}(n))$ time. In other words, the top-k query is at least as hard as *both* of its corresponding prioritized and max queries.

It turns out that, in terms of expected efficiency, the opposite is also true: the top-k query is *no harder* than solving both of the prioritized and max queries. To state this formally, let us first define a *geometrically converging function* to be a function $f : \mathbb{R}^+ \to \mathbb{R}^+$ (where $\mathbb{R}^+$ is the set of positive real numbers) satisfying two conditions:

- When $x \leq 2$, $f(x) = O(1)$;

- When $x > 2$,

$$\sum_{i=0}^{h} f\left(\frac{x}{c^i}\right) \quad = \quad O(f(x))$$

holds for any $c \geq 2$, where h is the largest integer i satisfying $x/c^i \geq 2$.

Note that all polynomial functions are geometrically converging. We are now ready to present the theorem, which is due to Rahul and Tao [27], for reducing a top-k problem to its prioritized and max counterparts:

THEOREM 2. *Fix D, $\mathbb{Q}$, and $\mathbb{W}$, and set $n = |D|$. Suppose that*

- *there is a structure on D that uses $\mathcal{S}_{pri}(n) = O(n^2)$ space, and answers a prioritized query in $\mathcal{Q}_{pri}(n) + O(t)$ time, where t is the number of elements reported;*
- *there is a structure on D that uses $\mathcal{S}_{max}(n)$ space, and answers a max query in $\mathcal{Q}_{max}(n)$ time, where function $\mathcal{S}_{max}(n)$ is geometrically converging.*

Then, there is a structure on D that uses $\mathcal{S}_{top}(n)$ space in expectation, and answers a top-k query in $\mathcal{Q}_{top}(n) + O(k)$ expected time, where

$$\mathcal{S}_{top}(n) = O\left(\mathcal{S}_{pri}(n) + \mathcal{S}_{max}\left(\frac{6n}{\mathcal{Q}_{pri}(n)}\right)\right) \quad (1)$$

$$\mathcal{Q}_{top}(n) = O\left(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n)\right). \quad (2)$$

Furthermore, if the prioritized and max structures support an update in $\mathcal{U}_{pri}(n)$ and $\mathcal{U}_{max}(n)$ time respectively, then the top-k structure supports an update in $O(\mathcal{U}_{pri}(n) + \mathcal{U}_{max}(n))$ expected time. If any of $\mathcal{U}_{pri}(n)$ and $\mathcal{U}_{max}(n)$ is amortized, the update cost of the top-k structure is amortized expected.

Several remarks are in order:

- The above reduction is optimal in the sense that there is no performance degradation (in expectation): the space, query, and update costs of the top-k structure are all determined by the worse between the prioritized and max structures. Theorems 1 and 2 together establish the equivalence (again, in terms of expected performance) between answering top-k queries and settling the combination of prioritized queries and max queries.
- Somewhat surprisingly, $\mathcal{S}_{top}(n)$ may even be smaller than $O(\mathcal{S}_{max}(n))$. For instance, plugging in $\mathcal{S}_{pri}(n) = O(n)$, $\mathcal{S}_{max}(n) = O(n \log n)$, and any $\mathcal{Q}_{pri}(n) \geq \log n$, we obtain from Theorem 2 that $\mathcal{S}_{top}(n) = O(n)$. Indeed, the theorem achieves a "bootstrapping" effect such that one does not need to try very hard to minimize the space of the max structure.
- The condition $\mathcal{S}_{max}(n) = O(n^2)$ essentially captures all the max structures useful in practice. The power 2 is not compulsory, and can be replaced by any constant.

We will prove Theorem 2 in Section 4.3.

4.2 Applications of Theorem 2

Theorem 2 provides a clear direction for designing top-k indexes with strong performance guarantees. Next, we demonstrate this on the two problems listed in Section 2.2.

Linear Ranking. Let D be the input set of n points in $\mathbb{R}^d$. Given real-valued coefficients $c_1, ..., c_k$ and a real value τ, a prioritized query returns all the points $p = (p_1, ..., p_d)$ in D such that $\sum_{i=1}^{d} c_i \cdot p_i \geq \tau$. Given c_1, ..., c_k, a max query returns the point $p = (p_1, ..., p_d) \in D$ that maximizes $\sum_{i=1}^{d} c_i \cdot p_i$.

Both types of queries have been well studied in computational geometry. The prioritized query is known as *halfspace reporting*, while the max query as the *extreme-point query*. When $d \leq 3$:

- Afshani and Chan [2] described a structure of $O(n)$ space that can answer a halfspace reporting query in $O(\log n + t)$ time, where t is the number of points reported;
- A technique of Dobkin and Kirkpatrick [12] yields a structure of $O(n)$ space that can answer an extreme-point query in $O(\log n)$ time.

Immediately, Theorem 2 guarantees a top-k index that uses $O(n)$ space and answers a top-k query in $O(\log n + k)$ time, both in expectation.

When $d \geq 4$, no known structure is able to answer a halfspace reporting query in $O(\text{polylog}\, n + t)$ expected time under the space budget of $\tilde{O}(n)$ expected. In fact, a lower bound of [13] even rules out the possibility of such structures for $d \geq 5$. Together with Theorem 1, these facts indicate that it is unrealistic to hope for a top-k index of near-linear space that can answer a top-k query in $O(\text{polylog}\, n + k)$ time. We will continue this discussion in Section 6, where a more suitable technique will be applied to design top-k indexes for $d \geq 4$.

1D Range Searching. Let D be the input set of n points in $\mathbb{R}$. Each point $e \in D$ is associated with a weight $w(e)$. Given an interval $q = [x, y]$ in $\mathbb{R}$ and a real value τ, a prioritized query returns all the points $e \in D$ satisfying $x \leq e \leq y$ and $w(e) \geq \tau$. Given an interval $q = [x, y]$, a max query reports the point of the maximum weight among the points $e \in D$ satisfying $x \leq e \leq y$.

It is rudimentary (see, e.g., [11]) to design a structure of $O(n)$ space which answers a max query in $O(\log n)$ time, and can be updated in $O(\log n)$ time. The prioritized query is what is called *3-sided range reporting*

in computational geometry. Specifically, let us create a set of 2D points $P = \{(e, w(e)) \mid e \in D\}$. A prioritized query with parameters $q = [x, y]$ and τ essentially returns all the points in P that fall in the 3-sided rectangle $[x, y] \times [\tau, \infty)$. By creating a *priority search tree* (PST) [21] on P, we can answer the query in $O(\log n + t)$ time, where t is the number of points reported. The PST occupies $O(n)$ space, and supports each update in $O(\log n)$ time.

Immediately, Theorem 2 yields a dynamic top-k index of $O(n)$ space that answers a top-k query in $O(\log n + k)$ time, and can be updated in $O(\log n)$ time, where all complexities hold in expectation.

4.3 Proof of Theorem 2

This subsection serves as a proof of Theorem 2. We consider that $\mathcal{Q}_{max}(n) = O(n)$ because a max query can be trivially answered by scanning D once.

Rank Sampling. Given a set S of real values, and a real value $0 < p \leq 1$, we define a *p-sample set* of S to be a set R obtained by the following random process. At the beginning, $R = \emptyset$; then, each element of S is added to R with probability p independently. Furthermore, we say that an element $e \in S$ has *rank* i if e is the i-th greatest in S. The following is a technical lemma that will be useful later.

LEMMA 1. *Let S be a set of n elements, and K a real value satisfying $2 \leq K \leq n/4$. For a $(1/K)$-sample set R of S, the following hold simultaneously with probability at least 0.09:*

- $|R| \geq 1$
- *The largest element in R has rank in S greater than K but at most $4K$.*

PROOF. The first bullet fails only if none of the elements in S was sampled, which occurs with probability

$$(1 - 1/K)^n \leq (1 - 1/K)^{4K} \leq 1/e^4$$

where the last inequality used the fact that $(1 - x)^{1/x} < 1/e$ for all $x > 0$.

Let x be the largest element in R, and denote by $\hat{K}$ the rank of x in S. Next, we bound the probability of the event $\hat{K} > 4K$, which occurs only if none of the $4K$ largest elements in D were sampled. Hence:

$$\mathbf{Pr}[\hat{K} > 4K] = (1 - 1/K)^{4K} \leq 1/e^4.$$

Finally, we bound the probability of the event $\hat{K} \leq K$, which occurs only if at least one of the K largest elements in D was sampled. Hence:

$$\mathbf{Pr}[\hat{K} \leq K] = 1 - (1 - 1/K)^K.$$

Applying the fact that $(1 - 1/x)^x \geq 1/e^2$ for all $x \geq 2$, we know:

$$\mathbf{Pr}[\hat{K} \leq K] \leq 1 - 1/e^2.$$

The union bound now shows that the probability of violating at least one bullet of Lemma 1 is at most

$$2/e^4 + (1 - 1/e^2) < 0.91$$

which completes the proof. □

Structure. We now describe how to design a top-k index using the given prioritized and max structures as black boxes. First, build a prioritized structure on the input dataset D. Then, fix a constant $\sigma = 1/20$, and define for each integer $i \geq 1$:

$$K_i \quad = \quad \mathcal{Q}_{max}(n) \cdot (1 + \sigma)^{i-1}.$$

Let h be the largest i such that $K_i \leq n/4$; clearly, $h = O(\log n)$. For each $i \in [1, h]$, we take a $(1/K_i)$-sample set R_i of D, and create a max structure on R_i. The top-k index consists of the prioritized structure and the h max structures constructed.

Query. Suppose that we need to answer a top-k query that chooses a predicate $q \in \mathbb{Q}$ and a weight function $w \in \mathbb{W}$. If $k < \mathcal{Q}_{max}(n)$, we obtain the result S of a top-$\mathcal{Q}_{max}(n)$ query with the same q and w, and extract the k elements in S with the largest weights using the k-selection algorithm of [6] in $O(|S|) = O(\mathcal{Q}_{max}(n))$ time. The total cost is therefore the time of the top-$\mathcal{Q}_{max}(n)$ query, which we will prove later is $O(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n))$ in expectation, plus $O(\mathcal{Q}_{max}(n))$.

Next, we consider $k \geq \mathcal{Q}_{max}(n)$. If $k \leq K_h$, set j^* to the smallest integer i satisfying $K_i \geq k$; note that $K_j = \Theta(k)$. Starting with $j = j^*$, we perform a *round* as follows:

1. Determine whether $|q(D)| < 4K_j$. This can be done in $\mathcal{Q}_{pri}(n) + O(K_j)$ time by performing a prioritized query in a *cost-monitoring* manner. Specifically, run a prioritized query with the parameters q, w, and $\tau = -\infty$, but terminate the query *manually* as soon as $4K_j$ elements have been reported. If manual terminate occurs, $|q(D)| \geq 4K_j$, and we proceed to the next step. Otherwise, the prioritized query finishes normally and must have returned the entire $q(D)$, implying $|q(D)| < 4K_j$, in which case we declare the round *successful* and terminate the whole algorithm by returning $q(D)$ as the result of the original top-k query.

2. Identify the element e in $q(R_j)$ with the maximum weight by issuing a max query on R_j with the parameters q and w which takes $\mathcal{Q}_{max}(n)$ time. In the special case where $q(R_j)$ is empty, treat e as a dummy element with $w(e) = -\infty$.

3. Perform a prioritized query on D with q, w, and $\tau = w(e)$ in a cost-monitoring manner:

(a) Either the query terminates *by itself*, outputting a set S of elements,

(b) Or we terminate it as soon as $4K_j+1$ elements have been reported.

In both cases, the cost is $\mathcal{Q}_{pri}(n) + O(K_j)$.

4. Declare this round *failed* if either of the following is true:

   - Case 3(a) occurred, but $|S| \leq K_j$.
   - Case 3(b) occurred.

   Otherwise, declare this round *successful*.

5. If the round is successful, perform k-selection on S to produce the k elements in $q(D)$ with the largest weights, and terminate the algorithm by returning them as the result of the top-k query.

6. Otherwise (i.e., failed), increase j by 1.

   (a) If $j \leq h$, perform the next round from Step 1.

   (b) Else (i.e., $j = h + 1$), answer the top-k query naively by reading the whole D in $O(n) = O(K_j)$ time. The algorithm then terminates. This is the only scenario where termination can happen in a failed round.

To analyze the cost of the algorithm, notice that a round fails only if $|q(|D|)| > 4K_j$ (otherwise, Line 1 terminates the algorithm), and one of the two bullets in Step 4 is true. Thus, Lemma 1 tells us that failure happens with probability at most 0.91, noticing that $q(R_j)$ is a $(1/K_j)$-sample set of $q(D)$. This implies that round j, for any $j \geq j^*$, is executed only with probability 0.91^{j-j^*}, namely, only when all the preceding rounds have failed. Also observe that round j, regardless of whether it fails, takes $\mathcal{Q}_{pri}(n)+\mathcal{Q}_{max}(n)+O(K_j)$ time. Thus, the expected cost of the algorithm is bounded by

$$\sum_{j=j^*}^{h} O\Big(\Big(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + K_j\Big) \cdot 0.91^{j-j^*}\Big)$$

$$= O\Big(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + \sum_{j=j^*}^{h} K_j \cdot 0.91^{j-j^*}\Big) \quad (3)$$

Note that $K_j = K_{j^*} \cdot (1+\sigma)^{j-j^*} = O(k) \cdot (1+\sigma)^{j-j^*}$. Plugging this into (3) shows that the expected cost is bounded by

$$O\left(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + k \sum_{j=j^*}^{h} ((1+\sigma) \cdot 0.91)^{j-j^*}\right)$$

which is $O(\mathcal{Q}_{pri}(n) + \mathcal{Q}_{max}(n) + k)$ because $(1+\sigma) \cdot 0.91 < 1$.

Space. The prioritized structure on D obviously takes up $\mathcal{S}_{pri}(n)$ space. We claim that all the max structures occupy $o(n) + O(\mathcal{S}_{max}(\frac{6n}{\mathcal{Q}_{max}(n)}))$ expected space in total, which implies the space result in Theorem 2 because $\mathcal{S}_{pri}(n) = \Omega(n)$.

The claim is fairly intuitive because $\mathbf{E}[|R_i|] = n/K_i$ geometrically decreases as i increases, which, together with the fact that $\mathcal{S}_{max}(n)$ is geometrically converging, seems to yield the claim immediately. The complication, however, is that $\mathcal{S}_{max}(n)$ may be a convex function, such that $\mathbf{E}[\mathcal{S}_{max}(|R_i|)]$ is *not* necessarily $O(\mathcal{S}_{max}(\mathbf{E}[|R_i|]))$. Next, we show how to circumvent this obstacle.

We will prove that all the max structures occupy $o(n) + O(\mathcal{S}_{max}(\frac{6n}{\mathcal{Q}_{max}(n)}))$ space in total with probability at least $1 - 1/n^2$. Combining this with the fact that all those structures obviously demand no more than $O(\mathcal{S}_{max}(n) \cdot h) = O(n^2 \cdot \log n)$ space gives the target claim.

Let i^* be the largest i such that $K_i \leq n/(3 \ln n)$. Consider an $i \in [1, i^*]$. Since $|R_i|$ is the sum of n independent Bernoulli variables each of which equals 1 with probability $1/K_i$, a standard application of Chernoff bounds[1] gives:

$$\begin{aligned} \mathbf{Pr}[|R_i| \geq 6 \cdot \mathbf{E}[|R_i|]] &\leq \exp(-\mathbf{E}[|R_i|]) \\ &= \exp(-n/K_i) \leq 1/n^3. \end{aligned}$$

Therefore, with probability at least $1 - h/n^3$, the max structures on $R_1, R_2, ..., R_{i^*}$ use at most

$$\begin{aligned} &\sum_{i=1}^{i^*} O\left(\mathcal{S}_{max}\left(\frac{6n}{\mathcal{Q}_{max}(n) \cdot (1+\sigma)^{i-1}}\right)\right) \\ = \; &O\left(h + \mathcal{S}_{max}\left(\frac{6n}{\mathcal{Q}_{max}(n)}\right)\right) \end{aligned}$$

space overall.

Let us now concentrate on $i \in [i^* + 1, h]$. Notice that there are only $O(\log \log n)$ such values of i. Also, by definition of i^*, we know that $\mathbf{E}[|R_i|] = n/K_i$ is in the range from 4 to $O(\log n)$. Another application of Chernoff bounds gives:

$$\begin{aligned} &\mathbf{Pr}[|R_i| \geq (\ln n^4) \cdot \mathbf{E}[|R_i|]] \\ \leq \; &\exp(-(\ln n^4) \cdot \mathbf{E}[|R_i|]/6) \\ \leq \; &\exp(-\ln n^{4 \cdot 2/3}) \\ = \; &1/n^{8/3}. \end{aligned}$$

Hence, with probability at least $1 - O(\log \log n)/n^{8/3}$, it holds that for all $i \in [i^* + 1, h]$:

$$|R_i| \leq 4 \ln n \cdot \mathbf{E}[|R_i|] = O(\log^2 n).$$

[1] Let $X_1, ..., X_n$ be independent Bernoulli variables such that $\mathbf{Pr}[X_i = 1] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[X] = \sum_{i=1}^{n} p_i$. For any $\alpha \in (0, 1)$, $\mathbf{Pr}[X \leq (1-\alpha)\mu] \leq e^{-\alpha^2 \mu/3}$, while for any $\alpha \geq 2$, $\mathbf{Pr}[X \geq \alpha\mu] \leq e^{-\alpha\mu/6}$.

By the fact that $\mathcal{S}_{max}(n) = O(1 + n^2)$, the max structures on R_{i^*}, R_{i^*+1}, ..., R_h together consume no more than $O(h + \log\log n \cdot \log^4 n) = o(n)$ space. We thus conclude that, with probability at least $1 - h/n^3 - O(\log\log n)/n^{8/3} > 1 - 1/n^2$, all the max structures use $o(n) + O(\mathcal{S}_{max}(\frac{6n}{\mathcal{Q}_{max}(n)}))$ space.

Update. It remains to discuss how to support insertions and deletions on the input set D. A crucial observation is that, in expectation, each element of D appears only in a constant number of max structures, noticing that the element belongs to R_i with probability $1/K_i$, which geometrically decreases as i increases. We can record using in expectation $O(1)$ words which max structures include e. It thus becomes obvious that the insertion or deletion of an element can be supported in $O(\mathcal{U}_{pri} + \mathcal{U}_{max})$ expected time. The above argument still works even if one or both of $\mathcal{U}_{pri}$ and $\mathcal{U}_{max}$ are amortized. This completes the whole proof of Theorem 2.

5. TECHNIQUES FOR PROBLEMS ON WEIGHT-AUGMENTED DATASETS

This section will focus on top-k problems where $|\mathbb{W}| = 1$, and hence, the weight $w(e)$ of each element $e \in D$ can be taken directly as an extra attribute of e. We will introduce several techniques that are effective on such problems, and at the same time amenable to practical implementation. Our description will be based on top-k range searching in 1D (see Section 2.2) whose simplicity will facilitate the understanding of the core ideas underneath. Unlike the solutions in Section 4 that are efficient in expectation, we will aim to obtain top-k indexes whose guarantees hold deterministically.

5.1 High-Level Ideas behind Sections 5.2-5.4

In 1D top-k range searching, the input dataset D is a set of n points in $\mathbb{R}$. Given an interval $q = [x, y]$, a top-k query returns the k points $e \in q(D)$ with the highest $w(e)$, where $q(D)$ is the set of points in D covered by q. Conceptually, we will answer the query in three steps:

1. *Size checking:* Decide whether $|q(D)| < k$. If so, simply retrieve the entire $q(D)$, and return it as the final result. Proceed to Step 2 only if $|q(D)| \geq k$.

2. *Thresholding:* Find a real-valued threshold τ such that at least k but at most $O(k)$ points $e \in q(D)$ satisfy $w(e) \geq \tau$.

3. *Prioritized reporting:* Find the set S of points in $q(D)$ whose weights are at least τ. The definition of τ makes sure that $k \leq |S| = O(k)$. Collect the k points in S with the highest weights, and return them as the result of the top-k query.

It is rudimentary to implement Step 1 in $O(\log n + k)$ time by resorting to a binary search tree (BST) on D. In Step 3, S can be found using a PST (priority search tree) on D in $O(\log n + |S|) = O(\log n + k)$ time, as we explained in Section 4.2. Finding the k points of the largest weights in S can be done with the k-selection algorithm [6], which takes $O(|S|) = O(k)$ time. Note that all the data structures needed in Steps 1 and 3 can be updated in $O(\log n)$ time per insertion and deletion.

The challenge is to design a data structure for Step 2, a phenomenon that is very typical in attacking a top-k problem through the above 3-step approach. In fact, Step 2 itself makes an interesting stand-alone problem, which was named the *approximate k-threshold problem* in [30]. In Sections 5.2-5.4, we will present several techniques to tackle the challenge

5.2 Technique 1: Binary Search

Let us start with a simple approach to find the target threshold τ — as in Step 2 of the algorithm in Section 5.1 — in $O(\log^2 n + k \log n)$ time using linear space. In fact, for a top-k query with search interval $q = [x, y]$, this approach returns a value of τ such that *precisely* k elements $e \in q(D)$ satisfy $w(e) \geq \tau$; furthermore, the τ returned is guaranteed to be the weight of some element in $q(D)$.[2]

Imagine that the weights of the points in D have been sorted in descending order into a list L. We can find the target τ by performing binary search on L. Specifically, the search starts by setting z to the median of L. Define c as the number of points $e \in q(D)$ with $w(e) \geq z$. We then determine which of the following is true: $c < k$, $c = k$, or $c > k$. If $c = k$, the algorithm finishes by returning $\tau = z$; otherwise, the search continues by focusing on the first or second half of L recursively.

The comparison between c and k can be resolved in $O(\log n + k)$ time by searching a PST in a cost-monitoring manner. Specifically, issue a prioritized query in the way explained in Section 4.2 to find all the points in $q(D)$ whose weights are at least z, with the difference that we manually terminate the prioritized query as soon as $k+1$ points have been reported. If manual termination occurs, c must be greater than k. Otherwise, c must be at most k, and all those c points must have been returned by the prioritized query. Overall, the binary search attempts $O(\log n)$ values of z, and therefore, finishes in $O(\log^2 n + k \log n)$ time.

The above strategy actually has a deeper implication regarding any top-k problem on weight-augmented datasets. Suppose that there is a structure of $\mathcal{S}_{pri}(n)$ space that can answer the corresponding prioritized query in $\mathcal{Q}_{pri}(n) + O(t)$ time (where t is the number of el-

[2]The operation finding such a τ is known as the *range quantile query* [17]

ements reported). Then, there must exist a top-k index of $O(\mathcal{S}_{pri}(n))$ space that answers a top-k query in $O(\mathcal{Q}_{pri}(n) \cdot \log n + k \log n)$ time. In Section 6, we will see a stronger result that has a better query bound, and eliminates the requirement of $|\mathbb{W}| = 1$.

5.3 Technique 2: Resorting to Counting

The query efficiency of the strategy in Section 5.2 can usually be improved, provided that there is a specialized structure for finding the number c faster. This is indeed the case for 1D range searching. Recall that c equals the number of points $e \in D$ satisfying $x \leq e \leq y$ and $w(e) \geq z$. If we introduce a 2D dataset $P = \{(e, w(e)) \mid e \in D\}$, c equals precisely the number of points in P that are covered by the rectangle $[x, y] \times [z, \infty)$. Computing this number is known as *orthogonal range counting*, which has been very well understood. We can preprocess P into a structure of Chazelle [10] which uses $O(n)$ space, and finds $|P \cap r|$ for any axis-parallel rectangle r in $O(\log n)$ time. With this, the query time of the solution in Section 5.2 is improved to $O(\log^2 n + k)$.

For a general top-k problem, a counting structure for finding c efficiently may not be readily available. The merit of the technique in Section 5.2 is to assure a reasonably good bound on the query cost using *only* a prioritized structure, which *must* be available for the reason explained in Section 3.

5.4 Technique 3: Dyadic Intervals

Assume, for simplicity, that n is a power of 2, and set $\Delta = \log_2 n$. Let us partition $\mathbb{R}$ into n/Δ disjoint intervals — referred to as *slabs* henceforth — such that each slab has exactly Δ points. Given an arbitrary interval $[x, y]$, we call it *aligned* if x and y are both slab boundaries, and define its *span* as the number of slabs that are fully contained in $[x, y]$. A *dyadic interval* is an aligned interval whose span is a power of 2. Note that the total number of dyadic intervals is $O((n/\Delta) \log n)$.

LEMMA 2. *For any aligned interval q, there exist two possibly overlapping dyadic intervals I_1 and I_2 that satisfy $I_1 \cup I_2 = q$.*

The proof is simple and omitted from this article.

Structure. For every dyadic interval I, store a *sketch* which consists of the 2^i-th largest weight of the points in $D \cap I$, for each $i \in [0, \log_2 n]$. If $|D \cap I| < 2^i$, then the 2^i-th largest weight is defined to be $-\infty$. All the $O((n/\Delta) \log n)$ sketches constitute our structure, whose space is $O((n/\Delta) \log^2 n)$.

Query. Given a top-k query with interval $q = [x, y]$, we now explain how to find a value τ that satisfies the requirements in Step 2 of the algorithm in Section 5.1. Assume, without loss of generality, that k is a power of 2 (otherwise, bump k up to the nearest power of 2). The base case happens when q is completely *within* a certain slab σ. In that case, we retrieve the set S' of points in $q \cap \sigma$, which takes $O(\log n + \Delta)$ time by searching a BST on D. Then, τ can be simply set to the k-th largest weight of the points in S', which can be found by performing k-selection in $O(\Delta)$ time.

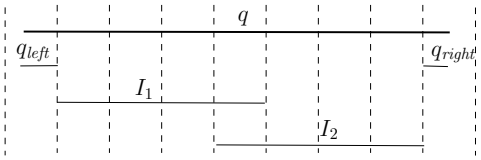

Figure 1: Partitioning a query interval

Let us now suppose that q intersects at least two slabs. Define $q_{mid} = [x', y']$ to be the longest *aligned* interval inside q, which gives rise to $q_{left} = [x, x']$ and $q_{right} = [y', y]$. Lemma 2 guarantees the existence of dyadic intervals I_1 and I_2 such that $I_1 \cup I_2 = q_{mid}$. See Figure 1 for an illustration, where the dashed lines represent slab boundaries.

Next, we obtain four values:

- τ_{left}: the k-th largest weight in $D \cap q_{left}$, or $-\infty$ if $|D \cap q_{left}| < k$;
- τ_1 (or τ_2): the k-th largest weight of the points in $D \cap I_1$ (or $D \cap I_2$, resp.);
- τ_{right}: the k-th largest weight in $D \cap q_{right}$, or $-\infty$ if $|D \cap q_{right}| < k$.

The values τ_{left} and τ_{right} can be obtained in $O(\Delta)$ time using the strategy illustrated earlier for the base case, while τ_1 and τ_2 can be fetched directly from the sketches of I_1 and I_2.

The τ returned is the *maximum* of the four values. It is easy to prove that at least k but at most $4k$ points in $q(D)$ can have weights at least τ.

Remark. Setting $\Delta = \log_2^2 n$ yields a linear space structure with $O(\log^2 n)$ query time, while $\Delta = \log_2 n$ gives a structure of $O(n \log n)$ space but $O(\log n)$ query time. It is possible to achieve linear space and $O(\log n)$ query time by recursively applying the same idea in each slab, but we will not delve into those details because competing for efficiency is not the purpose of this section.

5.5 Technique 4: Heap Selection

Let us define a *max-heap* H to be a tree where

- each internal node has a constant number of children, and
- each node u stores a real-valued *key* that is greater than all the keys stored in the proper subtree of u.

Note that H does not need to be balanced in any way. Given any max-heap H, Frederikson [16] described an algorithm to extract the k largest keys from H in $O(k)$ time, for any k ranging from 1 to the number of elements in H.

The above algorithm is useful for designing top-k indexes. Given a top-k query with predicate q, let us imagine that $q(D)$ has been divided into a number of sets $S_1, ..., S_s$ for some $s \geq 1$, and that the elements in each S_i ($1 \leq i \leq s$) have been stored in a max-heap H_i, using their weights as the keys. In such a scenario, we can find the top-k result in $O(s + k)$ time as follows. First coalesce $H_1, ..., H_s$ into a single max heap H on $S_1 \cup ... \cup S_s$. This can be done in $O(s)$ time, noticing that we only need to take the root of each H_i, and build a max-heap on those s roots. Once this is done, Frederikson's algorithm can be directly applied to find the k elements with the largest weights from H in $O(k)$ time.

Next, we apply the above idea to obtain an elegant top-k index for 1D range searching that consumes $O(n)$ space, guarantees $O(\log n + k)$ query time, and can be updated in $O(\log n)$ time per insertion and deletion.

Structure. Let us consider once again the set of n 2D points $P = \{(e, w(e) \mid e \in D\}$ constructed from D. It suffices to build a PST $\mathcal{T}$ on P, which can be defined recursively as follows (this is the first time in this article that we need to be concerned with the details of a PST):

- If $P = \emptyset$, $\mathcal{T}$ is an empty tree.
- If P contains only a single point $p = (p_x, p_y)$, $\mathcal{T}$ has only one node u which stores p, an *x-key* equal to p_x, and a *y-key* equal to p_y.
- Otherwise, let x_{med} be the median of the x-coordinates of the points in P, and $p^* = (p_x^*, p_y^*)$ be the highest point in P, i.e., having the greatest y-coordinate. Create the root u of $\mathcal{T}$, which stores p^*, an *x-key* equal to x_{med}, and a *y-key* equal to p_y^*.

  Define P_1 to be the set of points $p = (p_x, p_y)$ in $P \setminus \{p^*\}$ satisfying $p_x < x_{med}$, and P_2 symmetrically to be the set of points $p = (p_x, p_y)$ in $P \setminus \{p^*\}$ satisfying $p_x \geq x_{med}$. Recursively construct BSTs $\mathcal{T}_1$ and $\mathcal{T}_2$ on P_1 and P_2, respectively. Then, $\mathcal{T}$ is the tree obtained by making the root of $\mathcal{T}_1$ the left child of u, and that of $\mathcal{T}_2$ the right child of u.

Observe that $\mathcal{T}$ is a BST on the x-keys of the nodes, and simultaneously also a max-heap on the y-keys. It is clear that $\mathcal{T}$ has height $O(\log n)$, and occupies $O(n)$ space.

Query. Suppose that we are given a top-k (1D range searching) query with the search interval $q = [x_1, x_2]$. Without loss of generality, let us assume that both x_1 and x_2 are x-keys in $\mathcal{T}$. Denote by Π_1 (or Π_2) the path from the root of $\mathcal{T}$ to the node with x-key x_1 (or x_2, resp.).

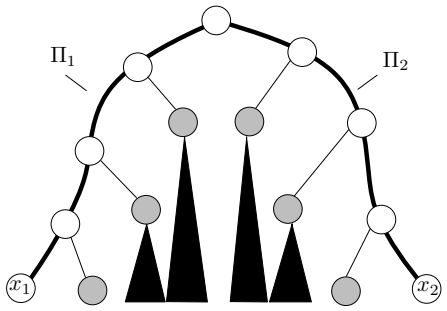

Figure 2: Searching a PST to perform top-k range searching in 1D space

From Π_1 and Π_2, we can obtain $s = O(\log n)$ nodes $v_1, ..., v_s$ with three properties:

- *Property 1*: The parent of each v_i ($1 \leq i \leq s$) is on Π_1 or Π_2.
- *Property 2*: The subtrees of $v_1, ..., v_s$, which are called the *canonical subtrees*, are mutually disjoint.
- *Property 3*: Every node, whose x-key is contained in q, must be either on $\Pi_1 \cup \Pi_2$ or in a canonical subtree.

Figure 2 shows an example where $s = 6$, and $v_1, ..., v_6$ are the nodes colored in gray.

Define $q(P)$ as the set of points $p = (p_x, p_y) \in P$ such that $x_1 \leq p_x \leq x_2$. Answering the top-k query is equivalent to finding the k highest points in $q(P)$. Property 3 ensures that every point in $q(P)$ must be stored at a node on $\Pi_1 \cup \Pi_2$, or a node in one of the canonical subtrees. Let us divide $q(P)$ into (i) P_1 (or P_2), which is the set of points in $q(P)$ stored on Π_1 (or P_2, resp.), and (ii) P_3, which the set of points stored in the canonical subtrees.

Let S_i ($1 \leq i \leq s$) be the set of y-keys in the canonical subtree rooted at v_i. Note that the canonical subtree is a max-heap on S_i. The k largest y-keys in $S_1 \cup ... \cup S_s$ can therefore be extracted in $O(k)$ time using Frederikson's algorithm. The points corresponding to those y-keys constitute the set S of k highest points in P_3. The final result of the top-k query is the k highest points in $S \cup P_1 \cup P_2$, which can be found using k-selection in $O(|S \cup P_1 \cup P_2|) = O(\log n + k)$ time.

Update. In [21], McCreight described a slightly different PST by allowing x_{med} to be an "approximate median". The benefit is that the resulting PST also supports an update in $O(\log n)$ time. The same query algorithm applies to that PST as well.

6. PRIORITIZED AS HARD AS TOP-K?

We now turn our attention back to all the top-k problems capured by our formulation in Section 6, i.e., no

matter whether $|\mathbb{W}| = 1$. We already know from Theorem 1 that top-k queries are no easier than prioritized queries, that is, a top-k index implies a prioritized structure with the same space-query tradeoff.

In this section, we will discuss the question opposite to the one resolved by Theorem 1. Let us fix D, $\mathbb{Q}$, and $\mathbb{W}$. Suppose that there is a structure on D that uses $\mathcal{S}_{pri}(n)$ space (recall that $n = |D|$), and answers any prioritized query in $\mathcal{Q}_{pri}(n) + O(t)$ time (where t is the number of elements reported). We want to use the structure as a black box to design a top-k structure. Let $\mathcal{S}_{top}(n)$ be the space consumption of the top-k structure, and $\mathcal{Q}_{top}(n) + O(k)$ its query cost. How good can the functions $\mathcal{S}_{top}(n)$ and $\mathcal{Q}_{top}(n)$ be?

Ideally, we would like to show $\mathcal{S}_{top}(n) = O(\mathcal{S}_{pri}(n))$ and $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n))$. This would imply that the top-k query was no *harder* than the corresponding prioritized query which, in turn, would conclude that top-k and prioritized queries in fact had the same computational hardness! Unfortunately, whether this is true still remains elusive today.

Nevertheless, decent progress has been made towards settling this open question. We now know that, when $\mathcal{Q}_{pri}(n) = \Omega(n^\epsilon)$ for any constant $\epsilon > 0$, it indeed holds that $\mathcal{S}_{top}(n) = O(\mathcal{S}_{pri}(n))$ and $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n))$. In other words, for *hard* problems whose prioritized queries demand a polynomial $\mathcal{Q}_{pri}(n)$, we can turn a prioritized structure into a top-k index with no efficiency loss! For easier problems with $\mathcal{Q}_{pri}(n) = \Omega(\log n)$, on the other hand, it is possible to show $\mathcal{S}_{top}(n) = O(\mathcal{S}_{pri}(n))$ and $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n) \cdot \log n)$. In other words, we can still obtain a top-k index from a prioritized structure by entailing only an $O(\log n)$ deterioration factor in query time.

Next, we introduce the theorem behind the above claims. Let us start with the notion of *polynomial boundedness*. Fix an integer $k \in [1, n]$. Remember that a top-k query selects a predicate $q \in \mathbb{Q}$ and a weight function $w \in \mathbb{W}$. In other words, the query result is a function of q and w. Since $|\mathbb{Q}|$ or $|\mathbb{W}|$ may be very large, the number of possible queries can be unbounded, but even so, it is possible that the number of *distinct* top-k results may be much smaller. In particular, if that number is always bounded by $n^{O(1)}$ for any input dataset $D \subseteq \mathbb{D}$ of size n (regardless of k), we say that the triplet $(\mathbb{D}, \mathbb{Q}, \mathbb{W})$ is *polynomially bounded*.

Rahul and Tao established[3] the following in [27]:

THEOREM 3. *Fix a polynomially bounded triplet of $(\mathbb{D}, \mathbb{Q}, \mathbb{W})$, and a set $D \subseteq \mathbb{D}$ of size n. Suppose that there is a structure on D that uses $\mathcal{S}_{pri}(n)$ space, and answers a prioritized query in $\mathcal{Q}_{pri}(n) + O(t)$ time, where t is the number of reported elements, such that*

- *$\mathcal{S}_{pri}(n)$ is geometrically converging, and*
- *$\mathcal{Q}_{pri}(n) = \Omega(\log n)$.*

Then, there is a top-k index of space $\mathcal{S}_{top}(n)$ and query time $\mathcal{Q}_{top}(n) + O(k)$ with

$$\begin{aligned} \mathcal{S}_{top}(n) &= O(\mathcal{S}_{pri}(n)) \\ \mathcal{Q}_{top}(n) &= O\left(\mathcal{Q}_{pri}(n) \cdot \frac{\log n}{\log \frac{\mathcal{Q}_{pri}(n)}{\log n}}\right). \end{aligned}$$

The proof in [27], which is technically involved and omitted from this article, shows how to construct a top-k index in the theorem using $n^{O(1)}$ expected time, where the constant power depends on the underlying problem.

Note that all complexities in Theorem 3 hold in the worst case. For $\mathcal{Q}_{pri}(n) = \omega(\log n)$, $\mathcal{Q}_{top}(n)$ is actually $o(\mathcal{Q}_{pri}(n) \cdot \log n)$, namely, the deterioration factor with respect to $\mathcal{Q}_{pri}(n)$ is $o(\log n)$. As an example, if $\mathcal{Q}_{pri}(n) = \Omega(\log^{1+\epsilon} n)$ for any positive constant $\epsilon > 0$, it holds that $\mathcal{Q}_{top}(n) = O(\mathcal{Q}_{pri}(n) \cdot \frac{\log n}{\log \log n})$. For $\mathcal{Q}_{pri}(n) = \Omega(n^\epsilon)$, the deterioration factor is $O(1)$, as mentioned earlier.

Polynomial boundedness is a property of many top-k problems. One example is the linear ranking problem defined in Section 2.2 under any constant dimensionality d. As explained in Section 4.2, the corresponding prioritized query of this problem is known as *halfspace reporting*. For $d \geq 4$, Afshani and Chan [2] described a structure of $O(n)$ space that answers any halfspace reporting query in $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor}) + O(t)$ time where t is the number of points reported. Immediately, Theorem 3 guarantees a top-k index of $O(n)$ space that answers a top-k query in $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor}) + O(k)$ time.

7. BEYOND THIS ARTICLE

We have reviewed only a small portion of the existing work on the class of top-k problems formulated in Section 2.1. Efficient indexes have been developed for the top-k versions of many traditional reporting problems, e.g., orthogonal range reporting [1, 7, 8, 25, 26, 30, 31], halfspace reporting [25, 27], rectangle stabbing [9, 27], and so on. The design of those indexes harbors numerous inspiring ideas which unfortunately cannot be included in this article.

Another non-trivial direction that has received significant development is the theory of top-k indexes in the *external memory* (EM) model [1,7,26,27,30,31]. Closely relevant to database systems, this model is widely used to study the behavior of *I/O-oriented* algorithms, whose performance bottlenecks lie in the data exchanges between different levels of the memory hierarchy — e.g.,

[3]Strictly speaking, [27] proved the theorem only for problems with $|\mathbb{W}| = 1$. However, the proof can be adapted to cover all polynomially bounded problems under the formulation in Section 2.1.

between the main memory and the disk — rather than in CPU computation. Specifically, in the EM model, a machine is equipped with M words of memory, and a disk that has been formatted into *blocks* of B words each. An *I/O* either reads a disk block into memory, or writes B words of memory into a disk block. CPU operations can be performed only on the data in memory. The *time* of an algorithm is measured in the number of I/Os performed (CPU computation is for free), while the *space* of a structure is measured in the number of disk blocks occupied. A "good" top-k index on an input dataset of size n should consume $\tilde{O}((n/B))$ space, and answer a query in $\mathcal{Q}_{top}(n, B) + O(k/B)$ I/Os, where $\mathcal{Q}_{top}(n, B)$ is a slow-growing function of n and B. Many of the techniques discussed in this article can be adapted to work in EM. In particular, see [26] for the counterpart of Theorem 1, and [27] for the counterparts of Theorems 2 and 3.

Finally, it is worth pointing out that the theory community has studied other top-1 or top-k problems that do not fit directly into the formulation in Section 2.1, e.g., problems on text retrieval [5, 18, 20, 22, 23, 29], uncertain data [3, 4, 32], colored reporting [28], etc.

8. FUTURE WORK DIRECTIONS

We conclude this article by mentioning four directions for future research:

- **Direction 1:** *Resolve the conjecture that the prioritized query is as hard as the corresponding top-k query.* Currently, there is an $O(\log n)$ gap in the query cost between the two (see Theorem 3). If this gap could be closed, we would have the surprising fact that every top-k problem in the class formulated in Section 2.1 is essentially the same as its prioritized version in terms of space-query tradeoff.
- **Direction 2:** *Obtain a high-probability version of Theorem 2.* The guarantees in that theorem currently hold in expectation only. Can we make them hold with a high probability (e.g., at least $1 - 1/n^2$)?
- **Direction 3:** *Fast construction of a top-k index in Theorem 3.* As mentioned in Section 6, currently it takes $n^{O(1)}$ expected time to build a top-k index with the guarantees stated in the theorem, which limits the theorem's applicability in practice. Can we reduce the cost to $\tilde{O}(n)$, provided that the given prioritized structure can be built in $\tilde{O}(n)$ time?
- **Direction 4:** *Study individual top-k problems with significant importance in practice.* The top-k perspective in Section 2.1 offers motivation for studying prioritized queries some of which otherwise would not appear sufficiently important to justify serious research efforts. On good example is *top-k halfspace reporting*, whose prioritized query is the following problem. We are given a set D of points in $\mathbb{R}^d$, each of which is associated with a real-valued weight. Given a halfspace q in $\mathbb{R}^d$ and a real value τ, a query returns all the points in $D \cap q$ whose weights are at least τ. The challenge is to preprocess D into a structure that can answer any such query efficiently. As a particularly interesting question, for $d = 2$, can we obtain a structure of $O(n)$ space that answers a query in $O(\log n + t)$ time, where t is the number of points reported?

9. REFERENCES

[1] Peyman Afshani, Gerth Stolting Brodal, and Norbert Zeh. Ordered and unordered top-k range reporting in large data sets. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 390–400, 2011.

[2] Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009.

[3] Pankaj K. Agarwal, Boris Aronov, Sariel Har-Peled, Jeff M. Phillips, Ke Yi, and Wuzhou Zhang. Nearest-neighbor searching under uncertainty II. *ACM Transactions on Algorithms*, 13(1):3:1–3:25, 2016.

[4] Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. Range-max queries on uncertain data. *Journal of Computer and System Sciences (JCSS)*, 94:118–134, 2018.

[5] Iwona Bialynicka-Birula and Roberto Grossi. Rank-sensitive data structures. In *String Processing and Information Retrieval (SPIRE)*, pages 79–90, 2005.

[6] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *Journal of Computer and System Sciences (JCSS)*, 7(4):448–461, 1973.

[7] Gerth Stolting Brodal. External memory three-sided range reporting and top-k queries with sublogarithmic updates. In *Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 23:1–23:14, 2016.

[8] Gerth Stolting Brodal, Rolf Fagerberg, Mark Greve, and Alejandro Lopez-Ortiz. Online sorted range reporting. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 173–182, 2009.

[9] Timothy Chan, Yakov Nekrich, Saladi Rahul, and Konstantinos Tsakalidis. Orthogonal point location

and rectangle stabbing queries in 3-d. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 31:1–31:14, 2018.

[10] Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal of Computing*, 17(3):427–462, 1988.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[12] David P. Dobkin and David G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6(3):381–392, 1985.

[13] Jeff Erickson. Better lower bounds for halfspace emptiness. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 472–481, 1996.

[14] Ronald Fagin. Combining fuzzy information from multiple systems. *Journal of Computer and System Sciences (JCSS)*, 58(1):83–99, 1999.

[15] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[16] Greg N. Frederickson. An optimal algorithm for selection in a min-heap. *Information and Computation*, 104(2):197–214, 1993.

[17] Travis Gagie, Simon J. Puglisi, and Andrew Turpin. Range quantile queries: Another virtue of wavelet trees. In *String Processing and Information Retrieval (SPIRE)*, pages 1–6, 2009.

[18] Wing-Kai Hon, Rahul Shah, Sharma V. Thankachan, and Jeffrey Scott Vitter. Space-efficient frameworks for top-k string retrieval. *Journal of the ACM (JACM)*, 61(2):9:1–9:36, 2014.

[19] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys*, 40(4):11:1–11:58, 2008.

[20] Marek Karpinski and Yakov Nekrich. Top-k color queries for document retrieval. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 401–411, 2011.

[21] Edward M. McCreight. Priority search trees. *SIAM Journal of Computing*, 14(2):257–276, 1985.

[22] S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002.

[23] Gonzalo Navarro and Yakov Nekrich. Time-optimal top-k document retrieval. *SIAM Journal of Computing*, 46(1):80–113, 2017.

[24] Manish Patil, Sharma V. Thankachan, Rahul Shah, Yakov Nekrich, and Jeffrey Scott Vitter. Categorical range maxima queries. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 266–277, 2014.

[25] Saladi Rahul and Ravi Janardan. A general technique for top-k geometric intersection query problems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 26(12):2859–2871, 2014.

[26] Saladi Rahul and Yufei Tao. On top-k range reporting in 2d space. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 265–275, 2015.

[27] Saladi Rahul and Yufei Tao. Efficient top-k indexing via general reductions. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 277–288, 2016.

[28] Biswajit Sanyal, Prosenjit Gupta, and Subhashis Majumder. Colored top-k range-aggregate queries. *Information Processing Letters (IPL)*, 113(19-21):777–784, 2013.

[29] Rahul Shah, Cheng Sheng, Sharma V. Thankachan, and Jeffrey Scott Vitter. Top-k document retrieval in external memory. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 803–814, 2013.

[30] Cheng Sheng and Yufei Tao. Dynamic top-k range reporting in external memory. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 121–130, 2012.

[31] Yufei Tao. A dynamic I/O-efficient structure for one-dimensional top-k range reporting. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 256–265, 2014.

[32] Ke Yi, Feifei Li, George Kollios, and Divesh Srivastava. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 20(12):1669–1682, 2008.

False News On Social Media: A Data-Driven Survey

Francesco Pierri
Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
francesco.pierri@polimi.it

Stefano Ceri
Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
stefano.ceri@polimi.it

ABSTRACT

In the past few years, the research community has dedicated growing interest to the issue of false news circulating on social networks. The widespread attention on detecting and characterizing deceptive information has been motivated by considerable political and social backlashes in the real world. As a matter of fact, social media platforms exhibit peculiar characteristics, with respect to traditional news outlets, which have been particularly favorable to the proliferation of false news. They also present unique challenges for all kind of potential interventions on the subject.

As this issue becomes of global concern, it is also gaining more attention in academia. The aim of this survey is to offer a comprehensive study on the recent advances in terms of detection, characterization and mitigation of false news that propagate on social media, as well as the challenges and the open questions that await future research on the field. We use a data-driven approach, focusing on a classification of the features that are used in each study to characterize false information and on the datasets used for instructing classification methods. At the end of the survey, we highlight emerging approaches that look most promising for addressing false news.

1. INTRODUCTION

This section serves as an introduction to the topic of false news on social media; we provide some terminology, describe the social media platforms where false news are most widespread, overview psychological and social factors that are involved, discuss some of the effects on the real world and some open challenges. Finally, we discuss the focus of our survey in comparison with other existing surveys.

1.1 Terminology

In recent years, the terms **false news** and **fake news** have been broadly and interchangeably used to indicate information which can take a variety of flavors: disinformation, misinformation, hoaxes, propaganda, satire, rumors, click-bait and junk news. We provide next a list of the definitions encountered in the literature, which is by no means exhaustive. While there is common agreement that these terms indicate deceptive information, we believe that an agreed and precise definition is still missing.

Some researchers define **false news** as news articles that are potentially or intentionally misleading for the readers, as they are verifiable and deliberately false [3, 63]. They can represent fabricated information which mimics traditional news content in form, but not in the intent or the organizational process [32]. It has been highlighted how the neologism **fake news** is usually employed with a political connotation with respect to the more traditional *false news* [32, 76].

Misinformation is defined as information that is inaccurate or misleading [32]. It could spread unintentionally [14] due to honest reporting mistakes or incorrect interpretations [22, 13]. In contrast, **disinformation** is false information that is spread deliberately to deceive people [32] or promote biased agenda [74].

Similarly to disinformation, **hoaxes** are intentionally conceived to deceive readers; qualitatively, they are described as *humorous and mischievous* (as defined in The Oxford English Dictionary) [31].

Satirical news are written with the primary purpose of entertaining or criticizing the readers, but similarly to hoaxes they can be harmful when shared out of context [8, 54]. They are characterized by humor, irony and absurdity and they can mimic genuine news [55].

Propaganda is defined as information that tries to influence the emotions, the opinions and the actions of target audiences by means of deceptive, selectively omitted and one-sided messages. The purpose can be political, ideological or religious [74, 73].

Click-bait is defined as low quality journalism which is intended to attract traffic and monetize

via advertising revenue [74].

The term **junk news** is more generic and it aggregates several types of information, from propaganda to hyper-partisan or conspiratorial news and information. It usually refers to the overall content that pertains to a publisher rather than a single article [79].

Finally, we came across several different definitions for **rumor**. Briefly, a rumor can be defined as a claim which did not originate from news events and that has not been verified while it spreads from one person to another [66, 3, 63]. As there exists a huge literature on the subject, we refer the interested reader to [82] for an extensive review.

1.2 Social media platforms as news outlets

The appearance of false news on news outlets is by no means a new phenomenon: in 1835 a series of articles published on the New York Sun, known as the *Great Moon Hoax*, described the discovery of life on the moon [3]. Nowadays the world is experiencing much more elaborated hoaxes; social media platforms have favored the proliferation of false news with much broader impact.

Most of nowadays **news consumption** has shifted towards online social media, where it is more comfortable to ingest, share and further discuss news with friends or other readers [19, 65, 63]. As producing content online is easier and faster, **barriers** for entering online media industry have dropped [3]. This has conveyed the dissemination of **low quality** news, which reject traditional journalistic standards and lack of third-party filtering and fact-checking [3]. These factors, together with a **decline** of general trust and confidence in traditional mass media, are the primary drivers for the explosive growth of false news on social media [3, 32].

Two main motivations have been proposed as to explain the rise of disinformation websites: 1) a **pecuniary** one, where viral news articles draw significant advertising revenue and 2) a more **ideological** one, as providers of false news usually aim to influence public opinion on particular topics [3]. Besides, the presence of **malicious agents** such as bots and trolls has been highlighted as another major cause to the spreading of misinformation [60, 30].

We refer the interested reader to [3] for an extensive analysis of various factors explaining the spreading of false news in social media platforms.

1.3 Human factors

Aside from the technical aspects of social network platforms, the research community has leveraged a set of psychological, cognitive and social aspects which are considered as key contributors to the proliferation of false news on social media.

Humans have no natural expertise at distinguishing real from false news [63, 31]. Two major psychological theories explain this difficulty, respectively called **naive realism** and the **confirmation bias**. The former refers to the tendency of users to believe that their view is the only accurate one, whereas those who disagree are biased or uninformed [50]. The latter, also called *selective exposure*, is the inclination to prefer (and receive) information which confirms existing views [39]. As a consequence, presenting factual information to correct false beliefs is usually unhelpful and may increase misperception [40].

Some studies also mention the importance of **social identity theory** [5] and **normative social influence** [4]; accordingly, users tend to perform actions which are socially safer, thus consuming and spreading information items that agree with the norms established within the community.

All these factors are related to a certain extent to the well-known **echo chamber** (or *filter bubble*) effect, which gives rise to the formation of homogeneous clusters where individuals are similar people, that share and discuss similar ideas. These groups are usually characterized by extremely polarized opinions as they are insulated from opposite views and contrary perspectives [67, 66, 42]; it has been shown that these close-knit communities are the primary driver of misinformation diffusion [10].

Social technologies amplify these phenomena as a result of **algorithmic bias**, as they promote personalized content based on the preferences of users with the unique goal of maximizing engagement [32, 14].

1.4 Effects on the real world

We can explain the explosive growth of attention on false news in light of a series of striking effects that the world has recently experienced.

Politics indeed accounts for most of the attention on false news, as highlighted in [76]. The 2016 US presidential elections have officially popularized the term *fake news* to the degree that it has been suggested that Donald Trump may not have been elected president were it not for the effects of false news (and the alleged interference of Russian trolls) [3]. Likewise, recent studies have shown that false news have also impacted 2016 UK Brexit referendum [26] and the 2017 France presidential elections [15].

Over and above we may recall the **finance** stock crisis caused by a false tweet concerning president

Obama [49], the **shootout** occurred in a restaurant as a consequence of the Pizzagate fake news [63] and the diffused mistrust towards **vaccines** during Ebola and Zika epidemics [16, 36].

1.5 Challenges

We mention here a few challenges which characterize the fight against false news on social media, as highlighted by recent research on the subject.

Firstly, false news are deliberately created to deceive the readers and to mimic traditional news outlets, resulting in an **adversarial scenario** where it is very hard to distinguish true news articles from false ones [63, 60].

Secondly, the **rate** and the **volumes** at which false news are produced overturn the possibility to fact-check and verify all items in a rigorous way, i.e. by sending articles to human experts for verification [60]. This also raises concern on developing tools for the **early detection** of false news as to prevent them from spreading in the network [33].

Finally, social media platforms impose limitations [61] on the **collection** of public data and as of today the community has produced very limited training datasets, which typically do not include all the information relative to false news.

1.6 Survey Focus

Aside from a few works appeared in 2015 and 2016 [8, 55, 54], we build our survey with a focus on the last two years, as most of the research on false news has developed in 2017 and 2018. Moreover, we concentrate on a few social networks which attracted most of the research focus: **Twitter**, **Facebook** and **Sina Weibo**[1]. This is mainly due to the public availability of data and the existence of proprietary application programming interfaces (API) which ease the burden of collecting data. As a final remark, we considered works covering solely the English language, as this is the prominent approach in the field.

Since our analysis is focused on the aforementioned social media, issues concerning false news on **collaborative platforms** such as Wikipedia and Yelp (namely fake reviews, spam detection, etc.) are out of the scope of this survey; we thus refer the reader to [30] for an overview of related research. We suggest [34] for a comprehensive review of the research that focuses, instead, on **rumors detection and resolution**, as we observed that many aspects are shared with our subject. **Automated fact-checking** is another related topic; it deals with verification rather than search of false news on social media, and we refer the interested reader to [69]. Finally, we suggest [16] to the readers who may be interested in the research on **social bots**.

[1]A popular Chinese microblogging website which is a hybrid between Facebook and Twitter.

2. PROBLEM FORMULATION AND METHODOLOGY

Our presentation of research about false news on social media is divided into three parts. We first describe a huge body of works whose objective is to detect false news, then we describe works that explain the models of diffusion of false news and finally works that attempt to mitigate their effects.

We start our survey by considering a variegated landscape of research contributions which focus on the **detection** of false news. Their taxonomy, presented in Table 1, is based on two aspects: employed technique and considered features.

The problem has been traditionally formulated as a supervised binary classification problem, starting with datasets consisting of labeled news articles, related tweets and Facebook posts which allow to capture different features, from content based ones (text, image, video) to those pertaining to the social context (diffusion networks, users' profile, metadata) and, in some cases, to external knowledge bases (Wikipedia, Google News). Labels carrying the classification into true and false news are typically obtained via fact-checking organizations or by manual verification of researchers themselves. Appendix A comparatively describes the datasets used as ground truth for false news classification.

For what concerns the classification method, a wide range of techniques are used, from traditional machine learning (Logistic Regression, Support Vector Machines, Random Forest) to deep learning (Convolutional and Recurrent Neural Networks) and to other models (Matrix Factorization, Bayesian Inference).

Section 4 describes the literature which focuses on the **characterization** of misinformation spreading on social media. This is achieved by reconstructing the diffusion networks pertaining to false news, as resulting from multiple users' interactions on the platforms.

Finally, Section 5 presents a few works which tackle the problem of **mitigation** against false news on social media, following recent announcements from major platforms to favor crowd-sourcing initiatives against malicious information [27].

3. FALSE NEWS DETECTION

We approach these methods by starting from those

	Machine Learning	Deep Learning	Other techniques
Content features	Wang et al (2017) [76] Horne et al. (2017) [24] Perez-Rosas et al. (2018) [46] Potthast et al. (2018) [44] Fairbanks et al. (2018) [12]	Baird et al. (2017) [6] Hanselowski et al. (2017) [20] Riedel et al. (2017) [50] Wang et al (2017) [76] Popat et al. (2018) [46]	Fairbanks et al. (2018) [12] Hosseinimotlagh et al. (2018) [25]
Context features	Tacchini et al. (2017) [67]	Volkova et al. (2017) [73] Wang et al. (2018) [77] Wu et al. (2018) [79] Liu et al. (2018) [32]	Tacchini et al. (2017) [67] Wang et al. (2018) [77] Yang et al. (2019) [80]
Content and context features	Shu et al. (2019) [63] Volkova et al. (2018) [72]	Ruchansky et al. (2017) [55] Volkova et al. (2018) [72]	Shu et al. (2019) [63]

Table 1. Comparative description of twenty studies for false news detection, in terms of method and considered features.

contributions which focus only on content-based features; we next describe contributions which consider only the social context and finally those that consider both aspects.

3.1 Content-based

In this section we consider research contributions which are content-based, meaning that they analyze solely the textual content of news articles, e.g. body, title, source.

Stance detection as a helpful first step towards fake news detection was introduced during the 2017 Fake News Challenge Stage 1[2] (FNC-1) organized by *D. Pomerleau et al. (2017)* [46] (cf. A.5). The goal was to classify the stance of an entire news article relative to its headline, i.e. document-level stance detection. Neural networks are employed by three top-performing systems, respectively Talos (*Baird et al. (2017)* [6]), Athene (*Hanselowski et al. (2017)* [20]) and UCL Machine Reading (*Baird et al. (2017)* [51]). These models rely on a combination of lexical features, including Bag-of-Words, topic modeling and word similarity features. An extensive analysis of these approaches, with experiments on their ability to generalize on unseen data, is provided by *Hanselowski et al. (2018)* [21].

Wang et al. (2017) [77] consider a multi-label classification task on the Liar dataset (cf. A.9), one of the first datasets introduced in the literature. This includes several textual and metadata features, such as the speaker affiliation or the source newspaper, and labels are based on the six degrees of truth provided by the PolitiFact[3] fact-checking organization. They solve the classification problem by considering several machine learning and deep learning methods, from logistic regression to convolutional and recurrent neural networks.

A deep textual analysis is carried out in *Horne at al. (2017)* [24], where authors examine the body and title (cf. A.1) of different categories of news articles (true, false and satire), extracting complexity, psychological and stylistic features. They highlight the relevance of each aspect in distinct classification tasks, using a linear Support Vector Machine (SVM), finally inferring that real news are substantially different from false news in title whereas satire and false news are similar in content. They also apply the Elaboration Likelihood Model [41] to news categories, and suggest that consuming false news requires little energy and cognition, making them more appealing to the readers.

A neural network model is also presented by *Popat et al. (2018)* [47], who build a framework to classify true and false claims, and also provide self-evidence for the credibility assessment. They evaluate their model against some state-of-the-art techniques on different collections of news articles (cf. A.3 and A.10) and they show examples of explainable results enabled by the *attention mechanism* embedded in the model, which highlights the words in the text that are more relevant for the classification outcome.

Perez-Rosas et al. (2018) [45] produce a dataset of false and true news articles (cf. A.4) and consider different sets of linguistic features (extracted from the body of news articles) namely ngrams, LIWC [44], punctuation, syntax and readability. On top of these features they train a linear SVM classifier, showing different performances depending on the considered feature. They suggest that computational linguistics can effectively aide in the process of automatic detection of false news.

The goal of *Potthast et al. (2018)* [48] is to assess the style similarity of several categories of news, notably hyper-partisan, mainstream, satire and false. The proposed methodology employs an algorithm called *unmasking* [29], which is a meta learning approach originally intended for authorship verifica-

[2] `http://www.fakenewschallenge.org`
[3] `https://www.politifact.com/`

tion. They carry out several experiments comparing topic and style-based features with a Random Forest classifier and they conclude that, while hyperpartisan, satire and mainstream news are well distinguished, a style-based analysis alone is not effective for detecting false news.

Fairbanks et al. (2018) [12] also aim to classify false and true news, using a collection of articles gathered from GDELT[4]); labels are manually crawled from a fact-checking website[5]. They compare two different models, a content-based one which uses a classifier on traditional textual features and a structural method that applies loopy belief propagation [38] on a graph built from the link structure of news articles. The conclusions indicate that by modeling just the text content of articles it is possible to detect bias, but it not possible to identify false news.

Hosseini et al. (2018) [25] tackle the problem of distinguishing different categories of false news (from satire to junk news), based only on the news content. They employ the Kaggle dataset (cf. A.8), where they consider up to six different labels. Their approach involves a tensor decomposition of documents which aims to capture latent relationships between articles and terms and the spatial/contextual relations between terms. They further use an ensemble method to leverage multiple decompositions in order to discover classes with higher homogeneity and lower outlier diversity. They outperform other state-of-the-art clustering techniques and are able to correctly identify all categories of fake news.

3.2 Context-based

Here we describe research contributions which are (social) context-based in the sense that they focus on information derived from social interactions between users, e.g. likes, comment and (re)tweets, as to detect fake content.

Tacchini et al. (2017) [68] propose a technique to identify false news on the basis of users who *liked* them on Facebook. They collect a set of posts and users from both conspiracy theories and scientific pages and they build a dataset where each feature vector represents the set of users who *liked* a page. They eventually compare logistic regression with a (boolean crowdsourcing) harmonic algorithm for showing that they are able to achieve high accuracy with a little percentage of the entire training data.

Volkova et al. (2017) [74] address the problem of predicting four sub-types of suspicious news: satire, hoaxes, click-bait and propaganda. They start from a (manually constructed) list of trusted and suspicious Twitter news accounts and they collect a set of tweets in the period of Brussels bombing in 2016. They incorporate tweet text, several linguistic cues (bias, subjectivity, moral foundations) and user interactions in a *fused* neural network model which is compared against ad-hoc baselines trained on the same features. They qualitatively analyze the characteristics of different categories of news observing the performances of the model.

Wang et al. (2018) [78] propose a multi-modal neural network model which extracts both textual and visual features from Twitter and Weibo conversations in order to detect false news items. Inspired by adversarial settings [18] they couple it with an *event discriminator*, which they claim is able to remove event-specific features and generalize to unseen scenarios, where the number of events is specified as a parameter. They evaluate the model on two custom datasets, but they compare it with ad-hoc baselines which are not conceived for false news detection.

Wu et al. (2018) [80] instead concentrate on modelling the propagation of messages carrying malicious items in social networks. Therefore they build a custom dataset, reflecting both true and false news, by leveraging the Twitter API and the fact-checking website Snopes[6]. They first infer embeddings for users from the social graph and in turn use a neural network model to classify news items. To this extent they provide a new model to embed a social network graph in a low-dimensional space and they construct a sequence classifier, using *Long Short-Term Memory* (LSTM) networks [23] to analyze propagation pathways of messages. They show that their model performs better than other state-of-the-art embedding techniques.

Propagation of news items is also taken into account by *Yu et al. (2018)* [33], who use a combination of convolutional and *Gated Recurrent Units* (GRU) [7] to model diffusion pathways as multivariate time series, where each point corresponds to the characteristics of the user retweeting the news, and perform early detection of false news. The method is evaluated on two real-world datasets of sharing cascades (cf. A.11) showing better performances than other state-of-the-art-techniques, which were nonetheless originally conceived for rumor resolution.

The first unsupervised approach to false news detection is provided in *Yang et al. (2019a)* [81], where veracity of news and users' credibility are treated as latent random variables in a Bayesian

[4] `https://www.gdeltproject.org/`

[5] `https://mediabiasfactcheck.com/`

[6] `https://www.snopes.com/`

network model, and the inference problem is solved by means of collapsed Gibbs sampling approach [53]. The method is evaluated on LIAR (cf. A.9) and BuzzFeedNews (cf. A.1) datasets, performing better than other general truth discovery algorithms, not explicitly designed for false news detection.

3.3 Content and Context-based

In this section we describe research contributions which consider both news content and the associated social (context) interactions as to detect malicious information items.

The contribution of *Ruchansky et al. (2017)* [56] is a neural network model which incorporates the text of (false and true) news articles, the responses they receive in social networks and the source users that promote them. The model is tested on Twitter and Weibo sharing cascades datasets (cf. A.11) and it is evaluated against other techniques conceived for rumor detection. They finally present an analysis of users behaviours in terms of lag and activity showing that the source is a promising feature for the detection.

In *Shu et al. (2017)* [64] a tri-relationship among publishers, news items and users is employed in order to detect false news. Overall, user-news interactions and publisher-news relations are embedded using non-negative matrix factorization [43] and users credibility scores. Several different classifiers are built on top of the resulting features and performances are evaluated on the FakeNewsNet dataset (cf. A.6) against other state-of-the-art information credibility algorithms. Results show that the social context could effectively be exploited to improve false news detection.

Volkova et al. (2018) [73] focus on inferring different deceptive strategies (misleading, falsification) and different types of deceptive news (propaganda, disinformation, hoaxes). Extending their previous work [74], they collect summaries, news pages and social media content (from Twitter) that refer to confirmed cases of disinformation. Besides traditional content-based features (syntax and style) they employ psycho-linguistic signals, e.g. biased language markers, moral foundations and connotations, to train different classifiers (from Random Forests to neural networks) in a multi-classification setting. Final results show that falsification strategies are easier to identify than misleading and that disinformation is harder to predict than propaganda or hoaxes.

3.4 Promising directions

Despite the vast amount of contributions discussed above, we believe that false news detection requires a deeper and more structured approach. Several works appear as academic exercises, not always compared to each other (and often not comparable). The main problem of most articles is that they achieve good performance when applied to given input dataset, but they do not generalize to unseen data. From our analysis, it seems that methods purely based upon content analysis work within a limited scope, whereas context analysis addresses generic actions (such as liking, commenting, propagating) that generalize more easily.

Here we highlight most promising approaches among works reviewed so far. They are also summarized in Table 2.

Among the articles from Section 3.1, focused on the content, we cite *Perez-Rosas et al. (2018)* [45] for its ability to consider a huge number of linguistic features, highlighting their different weights on the classification outcome. For such comprehensive approach, this work outstands on approaches based solely on news content; but the approach requires a considerable amount of annotated data (and thus manual efforts) which may hinder the setup of a real-world application.

Among the articles from Section 3.2, focused on the social context, we believe that *Liu et al. (2018)* [33] and *Wu et al. (2018)* [80] are most promising; they analyzed users' profiles and online news sharing cascades. Despite the inherent complexity of both techniques (and the limited datasets employed), we argue that a network-based approach focused on social responses might effectively detect deceptive information. They opened the way for new approaches that focus on the models of diffusion of false news on social media, where most of recent research advances stand as described next.

We finally cite *Volkova et al. (2018)* [73], among the articles from Section 3.3 based on both content and context-based features, for considering additional psycho-linguistic signals, e.g. biased language markers, moral foundations and connotations, and inspecting also social responses on Twitter as to infer different deceptive strategies and types of malicious information.

4. MODELS OF FALSE NEWS DIFFUSION

A first large-scale study on online misinformation is provided by *Del Vicario et al. (2016)* [10], who carry out a quantitative analysis on news consumption relatively to scientific and conspiracy theories news outlets on Facebook. They leverage the Facebook Graph API in order to collect a 5-year span

Reference	Task	Input Data	Methodology	Results
Perez-Rosas et al. (2018) [44]	Binary classification of false and true news articles.	News articles.	Linguistic features are extracted from the article body (LIWC, punctuation, syntax and readability) as to train a linear SVM classifier.	Accuracy up to 76%. Feature ablation study and evaluation of human performances at detecting false news.
Liu et al. (2018) [32]	Binary classification of false and true content.	Sharing cascades on Twitter and Weibo.	Propagation cascades are modeled as multivariate time series using users profile information, and neural networks based on GRU are used to classify items.	Other baselines are outperformed, with accuracy up to 92%. Early detection performance evaluation.
Wu et al. (2018) [79]	Binary classification of false and true content.	Sharing cascades on Twitter and Weibo.	The graph of social interactions (between users) is processed with a low-dimensional embedding and fed to an LSTM-based classifier.	Other baselines are outperformed with F1-score up to 91%. Performance evaluation in terms of different training/test size ratio.
Volkova et al. (2018) [72]	Multi-label classification of deceptive strategies (misleading, falsification) and types (propaganda, hoaxes, disinformation).	News articles. News summaries. Re-tweeting cascades.	Linguistic features (same as [44] plus word embeddings, biased language, moral foundations and connotations) are extracted to train several classifiers (from Logistic Regression to neural networks).	The combination of content and moral foundations and connotations is strongly predictive, neural networks outperform other classifiers.

Table 2: A summary of most promising directions for fake news detection.

of all the posts (and user interactions) which belong to the aforementioned categories. They analyze cascades (or *sharing trees*) in terms of lifetime, size and edge homogeneity (i.e. an indicator of the polarization of users involved) and they show that 1) the consumption patterns differ in the two categories and that 2) the *echo chambers* (or communities of interest) appear as the preferential drivers for the diffusion of content. On top of these results, they build a data driven percolation model which accounts for homogeneity and polarization and they simulate it in a small-world network reproducing the observed dynamics with high accuracy.

Similarly, a groundbreaking contribution is provided in *Vosoughi et al. (2018)* [76], where the entire Twitter universe is explored in order to track the diffusion of false and true news. Authors build a collection of links to fact-checking articles (from six different organizations) which correspond to true, false and mixed news stories and they accordingly investigate how these rumors spread on the Twitter network by gathering only tweets that explicitly contain the URLs of the articles. The resulting dataset contains approx. 126000 stories tweeted by 3 million users more than 4.5 million times. A series of measures are carried out including statistical and structural indicators of the retweeting networks along with sentiment analysis, topic distribution and novelty estimation of the different categories of news. The final results show that overall falsehood spread significantly faster, deeper, farther and broader than the truth in all categories of information, with a prominent weight on political news. Moreover, they observe that false news usually convey a higher degree of novelty and that novel information is more likely to be shared by users (although they cannot claim this is the only reason behind the "success" of misinformation).

A slightly diverse analysis is issued in *Shao et al. (2018a)* [61], where authors study the structural and dynamic characteristics of the core of the diffusion network on Twitter before and after the 2016 US Presidential Elections. They first illustrate the implementation and deployment of the Hoaxy platform [59] which is then employed to gather the data required for their analysis. They build different datasets (relative to a few months before and after the elections) which correspond to fact-checking and misinformation articles, i.e. the retweeting network of users that share URLs for related news items, and they perform a k-core decomposition analysis to investigate the role of both narratives in the network. They show that low-credibility articles prevail in the core, whereas fact-checking is almost relegated to the periphery of the network. They also carry out a network robustness analysis in order to analyze the role of most central nodes and guide possible different interventions of social platforms.

Same authors largely extend previous results in *Shao et al. (2018b)* [60], as they carry out a huge analysis on Twitter in a period of ten months in 2016 and 2017. They aim to find evidence of the considerable role of social bots in spreading low-credibility news articles. The Hoaxy [59] platform is leveraged once again and more than 14 million tweets, including fact-checking and misinformation sources, are collected. The *Botometer* algorithm [9] is used to assess the presence of social bots among Twitter users. The results show that bots are active especially in the first phase of the diffusion, i.e. a few seconds after articles are published, and that

although the majority of false articles goes unnoticed, a significant fraction tends to become viral. They also corroborate, to a certain extent, results provided by *Vosoughi et al. (2018)* [76]. Moving on, they highlight bot strategies for amplifying the impact of false news and they analyze the structural role of social bots in the network by means of a network dismantling procedure [2]. They finally conclude that curbing bots would be an effective strategy to reduce misinformation; using CAPTCHAs [75] is a simple tool to distinguish bots from humans, but with undesirable effects to the user experience of a platform.

Differently from previous works, a study of the agenda-setting [35] power of false news is instead accomplished in *Vargo et al. (2018)* [71], where authors focus on the online mediascape from 2014 to 2016. They leverage a few different agenda-setting models with a computational approach (collecting data from GDELT) in order to examine, among other targets, the influence of false news on real news reports, i.e. whether and to which extent false news have shifted journalistic attention in mainstream, partisan and fact-checking organizations. To this extent they gather news articles corresponding to partisan and mainstream news outlets as well as fact-checking organizations and false news websites; they refer to diverse references in the literature in order to manually construct the list. A network of different events and themes (as identified in the GDELT database) is built to relate distinct media and to model time series of (eigenvector) centrality scores [57] in order to carry out Granger causality tests and highlight potential correlations. Besides other results, they show that partisan media indeed appeared to be susceptible to the agendas of false news (probably because of the elections), but the agenda setting power of false news–the influence on mainstream and partisan outlets–is declining.

We described previous works in detail, as we strongly believe that they provided substantial research contributions to the phenomenon of false news spreading on social media, also due to the wide reach of large-scale experiments carried out in these studies. Overall, these approaches have shown that false news spread deeper, faster, broader and farther than the truth, with bots and echo chambers playing a primary role in (dis)information diffusion networks. They also cautiously suggest possible interventions which might be put in place by platform government bodies in order to curb this malicious phenomenon; nonetheless this can not be easily encouraged as it may raise ethical concerns about censorship. As aforementioned, we argue that future research should follow these directions and analyze, from a network perspective, how social communities react to online news as to identify malicious content.

5. FALSE NEWS MITIGATION

Finally, a few potential interventions have been proposed for reducing the spread of misinformation on social platforms, from curbing most active (and likely to be bots) users [60] to leveraging the users' flagging activity in coordination with fact-checking organizations. The latter approach is proposed as a first practical mitigation technique in [27] and [70], where the goal is to reduce the spread of misinformation leveraging users' flagging activity on Facebook.

Kim et al. (2018) [27] develop CURB, an algorithm to select the most effective stories to send for fact-checking as to efficiently reduce the spreading of non-credible news with theoretical guarantees; they formulate the problem in the context of temporal point processes [1] and stochastic differential equations and they use the Rumors datasets (A.11) to evaluate it in terms of precision and misinformation reduction (i.e. the fraction of prevented unverified exposures). They show that the algorithm accuracy is very sensitive to the ability of the crowd at spotting misinformation.

Tschiatschek et al. (2018) [70] also aim to select a small subset of news to send for verification and prevent misinformation from spreading; however, as they remark, with a few differences from the previous method respectively 1) they learn the accuracy of individual users rather than considering all of them equally reliable and 2) they develop an algorithm which is agnostic to the actual propagation of news in the network. Moreover, they carry out their experiments in a simulated Facebook environment where false and true news are generated by users in a probabilistic manner. They show that they are able at once to learn users' flagging behaviour and consider possible adversarial behaviour of spammer users who want to promote false news.

A different contribution is issued by *Vo et al. (2018)* [72], who are the first to examine active Twitter users who share fact-checking information in order to correct false news in online discussions. They incidentally propose a URL recommendation model to encourage these *guardians* (users) to engage in the spreading of credible information as to reduce the negative effects of misinformation. They use Hoaxy [59] (cf. A.7) to collect a large number of tweets referring to fact-checking organizations and they analyze several characteristics of

the users involved (activity, profile, topics discussed, etc). Finally, they compare their recommendation model, which takes into account the social structure, against state-of-the-art collaborative filtering algorithms.

Main social networking platforms, from Facebook to Twitter, have recently provided to their users tools to combat disinformation [27], an approach which seems reasonable enough to tackle the problem of disinformation without raising censorship alerts. Resorting to the *wisdom of the crowd*, as discussed above, can be effective at identifying malicious news items and prevent from misinformation spreading on social networks.

6. CONCLUSIONS

Despite the vast review of literature presented so far, in agreement with [32] we believe that there are only a few substantial research contributions, most of which specifically focus on characterizing the diffusion of misinformation on social media. It has been effectively shown that false news spread faster and more broadly than the truth on social media, and that social bots and echo chambers play an important role in the core of diffusion networks.

Although different psycho-linguistic signals derived from textual features are useful for false news detection, content alone may not be sufficient and other features, inferred from the social dimension, should be taken into account in order to distinguish false news from true news.

The lack of gold-standard agreed datasets and of research guidelines on the subject has favored the diffusion of ad-hoc data collections; the related detection techniques share several limitations, as they do not always compare with each other and do not explicitly discuss the impact and consequences of their results.

Nonetheless, the great number of contributions delivered in the last few years shows that the research community has promptly reacted to the issue, and that can successfully embody previous results to advance further in the combat against false news.

Besides the existing challenges highlighted in the introductory section, we believe that: 1) in light of recent contributions on the characterization of disinformation diffusion networks, more insights into false news detection should be gained from a network perspective; 2) in general, the research community should coordinate efforts originating from different areas (from psychology to journalism to computer science) in a more structured fashion; 3) future contributions should favor the development of real-world applications for providing effective help in the fight against false news.

7. ACKNOWLEDGEMENTS

F.P. and S.C. are supported by the PRIN grant HOPE (FP6, Italian Ministry of Education). S.C. is partially supported by ERC Advanced Grant 693174.

8. REFERENCES

[1] O. Aalen, O. Borgan, and H. Gjessing. *Survival and event history analysis: a process point of view*. Springer Science & Business Media, 2008.

[2] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378, 2000.

[3] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36, 2017.

[4] S. E. Asch and H. Guetzkow. Effects of group pressure upon the modification and distortion of judgments. *Groups, leadership, and men*, pages 222–236, 1951.

[5] B. E. Ashforth and F. Mael. Social identity theory and the organization. *Academy of management review*, 14(1):20–39, 1989.

[6] S. Baird, D. Sibley, and Y. Pan. Talos targets disinformation with fake news challenge victory. *Fake News Challenge*, 2017.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] N. J. Conroy, V. L. Rubin, and Y. Chen. Automatic deception detection: Methods for finding fake news. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, page 82. American Society for Information Science, 2015.

[9] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 273–274. International World Wide Web Conferences Steering Committee, 2016.

[10] M. Del Vicario, A. Bessi, F. Zollo, F. Petroni, A. Scala, G. Caldarelli, H. E. Stanley, and W. Quattrociocchi. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, 113(3):554–559, 2016.

[11] L. Derczynski, K. Bontcheva, M. Liakata, R. Procter, G. W. S. Hoi, and A. Zubiaga. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76, 2017.

[12] J. Fairbanks et al. Credibility assessment in the news: Do we need to read? In *Proc. of the MIS2 Workshop held in conjuction with 11th Int. Conf. on Web Search and Data Mining. 799800.*, 2018.

[13] D. Fallis. A conceptual analysis of disinformation. *iConference*, 2009.

[14] M. Fernandez and H. Alani. Online misinformation: Challenges and future directions. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 595–602. International World Wide Web Conferences Steering Committee, 2018.

[15] E. Ferrara. Disinformation and social bot operations in the run up to the 2017 french presidential election. *First Monday*, 22(8), 2017.

[16] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[17] W. Ferreira and A. Vlachos. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1168, 2016.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[19] J. Gottfried and E. Shearer. *News Use Across Social Medial Platforms 2016*. Pew Research Center, 2016.

[20] A. Hanselowski, P. Avinesh, B. Schiller, and F. Caspelherr. Description of the system developed by team athene in the fnc-1. *Fake News Challenge*, 2017.

[21] A. Hanselowski, P. Avinesh, B. Schiller, F. Caspelherr, D. Chaudhuri, C. M. Meyer, and I. Gurevych. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1859–1874, 2018.

[22] P. Hernon. Disinformation and misinformation through the internet: Findings of an exploratory study. *Government Information Quarterly*, 12(2):133–139, 1995.

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] B. D. Horne and S. Adali. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *arXiv preprint arXiv:1703.09398*, 2017.

[25] S. Hosseinimotlagh and E. E. Papalexakis. Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. In *Proc. of the MIS2 Workshop held in conjuction with 11th Int. Conf. on Web Search and Data Mining. 799800.*, 2018.

[26] P. N. Howard and B. Kollanyi. Bots,# strongerin, and# brexit: computational propaganda during the uk-eu referendum. *arXiv preprint arXiv:1606.06356*, 2016.

[27] J. Kim, B. Tabibian, A. Oh, B. Schölkopf, and M. Gomez-Rodriguez. Leveraging the crowd to detect and reduce the spread of fake news and misinformation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 324–332. ACM, 2018.

[28] B. Kollanyi, P. N. Howard, et al. The junk news aggregator: Examining junk news posted on facebook, starting with the 2018 us midterm elections. *arXiv preprint arXiv:1901.07920*, 2019.

[29] M. Koppel, J. Schler, and E. Bonchek-Dokow. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8(Jun):1261–1276, 2007.

[30] S. Kumar and N. Shah. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559, To appear in the book titled Social Media Analytics: Advances and Applications, by CRC press, 2018*, 2018.

[31] S. Kumar, R. West, and J. Leskovec. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web*, pages 591–602. International World Wide Web Conferences Steering Committee, 2016.

[32] D. M. J. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer,

M. J. Metzger, B. Nyhan, G. Pennycook, D. Rothschild, M. Schudson, S. A. Sloman, C. R. Sunstein, E. A. Thorson, D. J. Watts, and J. L. Zittrain. The science of fake news. *Science*, 359(6380):1094–1096, 2018.

[33] Y. Liu and Y.-F. Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. *AAAI Conference on Artificial Intelligence*, 2018.

[34] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3818–3824. AAAI Press, 2016.

[35] M. McCombs. *Setting the agenda: Mass media and public opinion*. John Wiley & Sons, 2018.

[36] J. Millman. The inevitable rise of ebola conspiracy theories. *The Washington Post*, 2014.

[37] S. Mukherjee and G. Weikum. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 353–362. ACM, 2015.

[38] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

[39] R. S. Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology*, 2(2):175, 1998.

[40] B. Nyhan and J. Reifler. When corrections fail: The persistence of political misperceptions. *Political Behavior*, 32(2):303–330, 2010.

[41] D. J. O'Keefe. Elaboration likelihood model. *The international encyclopedia of communication*, 2008.

[42] E. Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[43] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons. Text mining using non-negative matrix factorizations. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 452–456. SIAM, 2004.

[44] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn. The development and psychometric properties of LIWC2015. Technical report, 2015.

[45] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401. Association for Computational Linguistics, 2018.

[46] D. Pomerleau and D. Rao. Fake news challenge. http://www.fakenewschallenge.org, 2017.

[47] K. Popat, S. Mukherjee, A. Yates, and G. Weikum. Declare: Debunking fake news and false claims using evidence-aware deep learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 22–32, 2018.

[48] M. Potthast, J. Kiesel, K. Reinartz, J. Bevendorff, and B. Stein. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 231–240. Association for Computational Linguistics, 2018.

[49] K. Rapoza. Can 'fake news' impact the stock market? *Forbes*, 2017.

[50] E. S. Reed, E. Turiel, and T. Brown. Naive realism in everyday life: Implications for social conflict and misunderstanding. In *Values and knowledge*, pages 113–146. Psychology Press, 2013.

[51] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*, 2017.

[52] M. Risdal. Fake news dataset. https://www.kaggle.com/mrisdal/fake-news. 2017.

[53] C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[54] V. Rubin, N. Conroy, Y. Chen, and S. Cornwell. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, 2016.

[55] V. L. Rubin, Y. Chen, and N. J. Conroy. Deception detection for news: three types of fakes. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with*

Impact: Research in and for the Community, page 83. American Society for Information Science, 2015.

[56] N. Ruchansky, S. Seo, and Y. Liu. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806. ACM, 2017.

[57] B. Ruhnau. Eigenvector centrality: a node centrality? *Social networks*, 22(4):357–365, 2000.

[58] G. Santia and J. Williams. Buzzface: A news veracity dataset with facebook user commentary and egos. *International AAAI Conference on Web and Social Media*, 2018.

[59] C. Shao, G. L. Ciampaglia, A. Flammini, and F. Menczer. Hoaxy: A platform for tracking online misinformation. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 745–750, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[60] C. Shao, G. L. Ciampaglia, O. Varol, K.-C. Yang, A. Flammini, and F. Menczer. The spread of low-credibility content by social bots. *Nature communications*, 9(1):4787, 2018.

[61] C. Shao, P.-M. Hui, L. Wang, X. Jiang, A. Flammini, F. Menczer, and G. L. Ciampaglia. Anatomy of an online misinformation network. *PLOS ONE*, 13(4):1–23, 04 2018.

[62] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018.

[63] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, Sept. 2017.

[64] K. Shu, S. Wang, and H. Liu. Beyond news contents: The role of social context for fake news detection. *arXiv preprint arXiv:1712.07709 (2017), to appear in Proceedings of 12th ACM International Conference on Web Search and Data Mining (WSDM 2019)*.

[65] C. Silverman. This analysis shows how fake election news stories outperformed real news on facebook. BuzzFeed, `https://zenodo.org/record/1239675`, 2016.

[66] C. Sunstein. *On Rumors: How Falsehoods Spread, Why We Believe Them, What Can Be Done*. New Haven: Yale University Press. Stowe, 2007.

[67] C. R. Sunstein. *Echo chambers: Bush v. Gore, impeachment, and beyond*. Princeton University Press, 2001.

[68] E. Tacchini, G. Ballarin, M. L. Della Vedova, S. Moret, and L. de Alfaro. Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*, 2017.

[69] J. Thorne and A. Vlachos. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, 2018.

[70] S. Tschiatschek, A. Singla, M. Gomez Rodriguez, A. Merchant, and A. Krause. Fake news detection in social networks via crowd signals. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 517–524. International World Wide Web Conferences Steering Committee, 2018.

[71] C. J. Vargo, L. Guo, and M. A. Amazeen. The agenda-setting power of fake news: A big data analysis of the online media landscape from 2014 to 2016. *New Media & Society*, 20(5):2028–2049, 2018.

[72] N. Vo and K. Lee. The rise of guardians: Fact-checking url recommendation to combat fake news. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 275–284, New York, NY, USA, 2018. ACM.

[73] S. Volkova and J. Y. Jang. Misleading or falsification: Inferring deceptive strategies and types in online news and social media. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 575–583, 2018.

[74] S. Volkova, K. Shaffer, J. Y. Jang, and N. Hodas. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 647–653, 2017.

[75] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *International Conference on the Theory and Applications of Cryptographic*

Techniques, pages 294–311. Springer, 2003.
[76] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.
[77] W. Y. Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 422–426, 2017.
[78] Y. Wang, F. Ma, Z. Jin, Y. Yuan, G. Xun, K. Jha, L. Su, and J. Gao. Eann: Event adversarial neural networks for multi-modal fake news detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 849–857. ACM, 2018.
[79] S. C. Woolley and P. N. Howard. *Computational Propaganda: Political Parties, Politicians, and Political Manipulation on Social Media*. Oxford University Press, 2018.
[80] L. Wu and H. Liu. Tracing fake-news footprints: Characterizing social media messages by how they propagate. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 637–645. ACM, 2018.
[81] S. Yang, K. Shu, S. Wang, R. Gu, F. Wu, and H. Liu. Unsupervised fake news detection on social media: A generative approach. In *Proceedings of 33rd AAAI Conference on Artificial Intelligence*, 2019.
[82] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51(2):32:1–32:36, Feb. 2018.

APPENDIX

A. DATASETS

The research community has produced a rich but heterogeneous ensemble of data collections for fact checking, often conceived for similar objectives and for slightly different tasks. We first introduce the datasets which are referenced in this survey along with a short description, the source and the main references; their features are summarized in Table 2. Next, we present some other interesting datasets.

A.1 BuzzFeedNews

BuzzFeed[7] News journalists have produced different collections of verified false and true news, shared by both hyperpartisan and mainstream news media on Facebook in 2016 and 2017; two of them, introduced by *Silverman (2016)* [65], consist of title and source of news items and they are used in [81, 24, 58]

A.2 BuzzFeed-Webis

This collection extends the previous one as it also contains the full content of shared articles with attached multimedia; it is employed in [48].

A.3 DeClare

This dataset contains several articles from Snopes, PolitiFact and NewsTrust [37] corresponding to both true and false claims; it is proposed in [47] and used for false news detection.

A.4 FakeNewsAMT

This collection contains some legitimate articles from mainstream news, some false news generated by Amazon Mechanical Turk workers and some false and true claims from GossipCop[8] (a celebrity fact-checking website); it is introduced in [45] for false news detection.

A.5 FakeNewsChallenge

This dataset was proposed for the 2017 Fake News Challenge Stage 1 [46]; it contains thousands of headlines and documents which have to be classified in a document-based stance detection task using 4 different labels (Agree, Discuss, Disagree, Unrelated). It was inspired by [17] where stance detection is instead applied at the level of single sentences. It is employed in [20, 6, 51]; an additional analysis is provided in [21].

A.6 FakeNewsNet

This dataset contains both news content (source, body, multimedia) and social context information (user profile, followers/followee) regarding false and true articles, collected from Snopes and BuzzFeed and shared on Twitter; it was presented in [62] and employed in [64].

A.7 Hoaxy

The Hoaxy platform[9] has been first introduced in [59] and employed in several studies [72, 60, 61] for different goals; it is continuously monitoring the diffusion network (on Twitter, since 2016) of news articles from both disinformation and fact-checking websites and it allows to generate custom data collections.

[7] `https://www.buzzfeed.com`
[8] `https://www.gossipcop.com`
[9] `https://hoaxy.iuni.iu.edu`

	Content Features	Social Context Features	Size	Labeling	Platform	Reference
BuzzFeedNews	Article title and source	Engagement ratings	10^2	BuzzFeed	Facebook	[65]
BuzzFeedWebis	Full Article	-	10^3	BuzzFeed	Facebook	[48]
DeClare	Fact-checking post	-	10^5	NewsTrust PolitiFact Snopes	-	[47]
FakeNewsAMT	Article text only	-	10^3	Manual GossipCop	-	[45]
FakeNewsChallenge	Full article	-	10^3	Manual	-	[46]
FakeNewsNet	Full article	Users metadata	10^3	BuzzFeed PolitiFact	Twitter	[62]
Hoaxy	Full article	Diffusion network Temporal trends Bot score (for users)	$> 10^6$	-	Twitter	[59]
Kaggle	Article text and metadata	-	10^4	BS Detector	-	[52]
Liar	Short statement	-	10^4	PolitiFact	-	[77]
SemEval-2017 Task8	Full article Wikipedia articles	Threads (tweets, replies)	10^4	Manual	Twitter	[11]
Rumors	Fact-checking title	Diffusion network (Twitter) Original message, replies (Weibo)	10^4	Snopes Weibo	Twitter Sina Weibo	[34]

Table 3. Comparative description of the datasets referenced in this survey.

A.8 Kaggle

This dataset was conceived for a Kaggle false news detection competition [52] which contains text and metadata from websites indicated in the BS Detector[10]; it is employed in [25].

A.9 Liar

This is a collection of short labeled statements from political contexts, collected from PolitiFact, which serve for false news classification; it first appeared in [77] and it is employed in [81].

A.10 SemEval-2017 Task8

This data collection, composed of tweets and replies which form specific *conversations*, was designed for the specific tasks of stance and veracity resolution of social media content on Twitter; it is described in [11] and used in [47].

A.11 Rumors

This dataset was originally conceived for rumor detection and resolution in Twitter and Sina Weibo; introduced in [34], it contains retweet and discussion cascades corresponding to rumors/non-rumors and it is employed for false news detection and mitigation in [56, 27, 33].

A.12 Others

BuzzFace is a novel data collection composed of annotated news stories that appeared on Facebook during September 2016; it extends previous BuzzFeed dataset(s) (cf. A.1) with comments and the web-page content associated to each news article; itjunknews is introduced in [58].

As a complement to Hoaxy (cf. A.7), **JunkNewsAggregator** is a platform that tracks the spread of disinformation on Facebook pages; it is described in [28].

Other datasets point to relevant organizations in the context of false news: [71] contains a list of false news outlets as indicated by different fact-checking organizations, whereas the list of signatories[11] of the International Fact Checking Network's code of

[10] https://github.com/bs-detector/bs-detector

[11] `https://ifcncodeofprinciples.poynter.org/signatories`

principles is a collector of the main fact-checking organizations which operate in different countries. Finally, [3] provides a set of the most shared false articles identified on Facebook during 2016 US elections.

Build your own SQL-on-Hadoop Query Engine

A Report on a Term Project in a Master-level Database Course

Stefanie Scherzinger
Ostbayerische Technische Hochschule Regensburg
Regensburg, Germany
stefanie.scherzinger@oth-regensburg.de

ABSTRACT

This is a report on a course taught at OTH Regensburg in the summer term of 2018. The students in this course built their own SQL-on-Hadoop engine as a term project in just 8 weeks. *miniHive* is written in Python and compiles SQL queries into MapReduce workflows. These are then executed on Hadoop. *miniHive* performs generic query optimizations (selection and projection pushdown, or cost-based join reordering), as well as MapReduce-specific optimizations.

The course was taught in English, using a flipped classroom model. The course material was mainly compiled from third-party teaching videos. This report describes the course setup, the *miniHive* milestones, and gives a short review of the most successful student projects.

1. MOTIVATION

When taking a big data course at an applied university of sciences such as OTH Regensburg, students expect a hands-on, coding-intensive experience. Since the majority of our students pursues a career in industry, they expect to get in touch with technology that will be an immediate asset for their CVs. Currently, this seem to be the Apache projects HDFS, Hadoop, Hive, and Spark.

Yet to the first-time user, interacting with HDFS may just feel like a simple file system. "Teaching HiveQL" is tricky, too: For the student fluent in SQL, writing first HiveQL queries seems unspectacular. Of course, these first impressions are treacherous, as there are language features in HiveQL that require a deeper understanding of the MapReduce data flow (e.g., SORT-BY versus ORDER-BY).

In designing her Master-level course titled "Modern Database Concepts", the author of this report wanted to teach the ideas behind engines like Hive, as well as the design decisions regarding query language constructs.

The students were therefore asked to build *miniHive*, an SQL-on-Hadoop engine for compiling SQL queries into MapReduce workflows. *miniHive* was designed according to the original presentation of Hive as a VLDB demo [14] in 2009: This version of Hive supported no updates, used an internal algebra to represent query plans, and could perform common logical optimizations (in particular, selection and projection pushdown). Back then, all physical operators were implemented as MapReduce jobs. As a MapReduce-specific optimization, Hive merges jobs using a technique called chain folding [11].

Among the 60 students taking the final exam, 25 students built a working SQL-on-Hadoop engine, which compiles SQL queries, performs generic logical optimizations, and executes them on Apache Hadoop. This is impressive insofar as the term project was optional, and stretched over just 8 weeks. Moreover, 11 submissions of *miniHive* also implemented chain folding among further optimizations.

Structure. In the following, we describe the course and its term project. Section 2 outlines the development of *miniHive* in four milestones, as well as how the students then perceived working with Apache Hive and Apache Spark. Section 3 describes the testbed and evaluation. Section 4 concludes.

2. THE FOUR MILESTONES

In "flipping" the course, the instructor relied on students to prepare the required theory on their own. Each week, they were assigned videos or book chapters. While studying the material, they answered a set of questions and submitted their answers online, prior to class. Since the students were allowed to take these notes into the final exam as reference material, they were motivated to diligently compile their answers.

During the weekly classroom sessions, the instructor and the students revised the prepared notes together, and worked on paper-based exercises to practice and apply the material.

During the lab sessions and in the students' own time, *miniHive* was built with Python 3.6 in four successive milestones. The deadlines for submitting

the milestones were spaced two weeks apart, which admittedly, is a sporty pace. Milestone specifications came with unit tests that submissions had to pass. Successful submissions were awarded bonus points that counted towards the exam.[1]

We now describe the scope of each milestone, the required material for self-study, and the coding challenges, in turn. We then report on our observations how students interacted with Apache Hive and Spark, having already built *miniHive*.

2.1 Compiling SQL to Relational Algebra

Scope: In the first milestone, the students implemented the canonical translation of conjunctive SQL queries into relational algebra. In particular, we support the fragment of queries of the form

```
SELECT DISTINCT ⟨ list of attributes to select ⟩
FROM   ⟨ list of relation names ⟩
[ WHERE  ⟨ condition ⟩ ]
```

where the condition is a conjunction of atomic equality conditions. Different from Hive, *miniHive* does not support nested relations. The translation into an equivalent relational algebra expression was intended as a warm-up exercise for students new to Python. For instance, the following query over Jennifer Widom's pizza scenario [7] produces the ages of all persons who eat mushroom pizza:

```
SELECT DISTINCT P.age
FROM Person P, Eats E
WHERE P.name = E.name AND E.pizza = 'mushroom'
```

The compilation of this query into relational algebra is by the book. Below, we make use of the straightforward syntax of the `radb` interpreter for relational algebra [15]. This interactive interpreter was written by Jun Yang from *Duke University* and is a great teaching tool.

```
\project_{P.age}
 \select_{P.name = E.name and E.pizza = 'mushroom'}
 (\rename_{P:*}(Person) \cross \rename_{E:*}(Eats))
```

Several MapReduce-specific algebras have been proposed that provide powerful operators, e.g. [13]. However, this author chose to settle with traditional relational algebra, which is taught as part of the undergraduate database course at OTH Regensburg.

Independent Study: In advance, students taught themselves Python with a free course offered on the Udacity MOOC platform [10]. Moreover, they watched Jennifer Widom's video lectures for a refresher on relational algebra, offered on Stanford's MOOC platform Lagunita [7]. This teaching unit comes with interactive exercises that use the very same `radb` syntax as above.

The video lectures were to be completed over the first four weeks of the semester. The author took great care in choosing appealing material. Indeed, in the course evaluation, several students stated that they very much enjoyed taking these altogether excellent online courses.

Coding: The students then set out to compile SQL. As a query parser, they used the Python module `sqlparse` [1]. With over 50 contributors and over 1,600 stars on GitHub, this is a popular SQL parser.

As datastructures to represent relational algebra queries, the students simply used the Python classes declared within the `radb` source code [15].

2.2 Selection Pushdown

Scope: In the second milestone, the students performed selection pushing on the relational algebra queries, and translated cross products into joins, where possible. For this milestone, a data dictionary was provided. Selection pushing is also a key feature in first public release of Hive, as described in [14], whereas projection pushing, also included in the first release of Hive, was left as an optional feature for the final milestone.

In the example from before, this yields the following equivalent query in `radb` syntax:

```
\project_{P.age}
 (\rename_{P:*}(Person) \join_{P.name = E.name}
  (\select_{E.pizza = 'mushroom'}
    \rename_{E:*}(Eats)))
```

Independent Study: For the theory on logical query optimization (selection and projection pushdown, as well as cost-based join reordering), the students followed parts of Jens Dittrich's flipped database course, which comes complete with in-class quizzes and exercises [3]. This material covers more than what is necessary for milestone 3, but is also a basis for the final milestone, where students could choose which optimizations to implement.

Coding: Coding for this milestone mainly involved recursive rewriting of the relational algebra trees. This gave the students the opportunity to familiarize themselves with the `radb` module.

2.3 A First Physical Query Plan

Scope: In the third milestone, logical operators were mapped to physical, MapReduce-based operators. The output is a tree-shaped workflow of

[1]Examination regulations at OTH Regensburg for this course require that the final grade is determined by a written exam. They further prohibit that the final grade is earned in part by an assignment. Thus, bonus points are our incentive for students to write code.

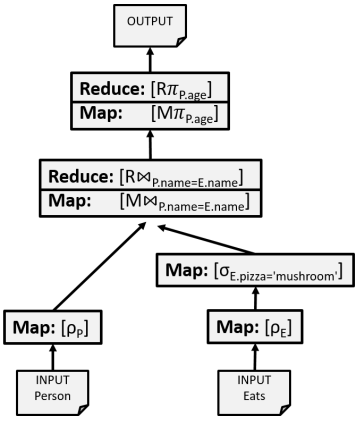

Figure 1: Naive physical query plan: Each node implements a single operator.

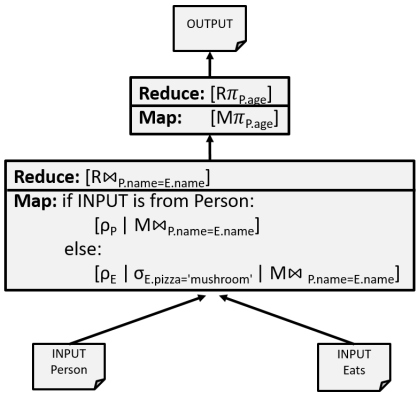

Figure 2: Plan after chain folding.

MapReduce jobs. Figure 1 shows the physical query plan for our running example. The data flow is from bottom to top. Renaming and selection can be realized as Map-only jobs. In the syntax used in this figure, this is denoted as "**Map:** $[\rho_{\mathsf{E}}]$" and "**Map:** $[\sigma_{\mathsf{E.pizza='mushroom'}}]$" respectively, where we first specify the type of the function (either **Map** or **Reduce**), and then state the relational algebra operator implemented in brackets.

In contrast, join and relational projection (due to duplicate elimination) require a full MapReduce job. Let us consider the final MapReduce job implementing the projection. The implementation of the Map-job, which we denote as "**Map:** $[\mathsf{M}\pi_{\mathsf{P.age}}]$", will emit key-value pairs where the key is the person's age. The Reduce-job "**Reduce:** $[\mathsf{R}\pi_{\mathsf{P.age}}]$" simply outputs the unique key from its input.[2]

The resulting workflow can be immediately executed on Hadoop, to naively evaluate SQL queries. Like in Hive, the intermediary results of the physical operators in the query plan are stored in HDFS.

Independent Study: The students familiarized themselves with MapReduce processing by taking an online course offered by Cloudera [9]. This course also uses Python and contains a set of introductory MapReduce coding exercises. MapReduce jobs can be run on Hadoop on a Linux-based virtual machine, the Cloudera Quickstart VM[3].

Implementing relational algebra operators with MapReduce is comprehensively described in the book "Mining Massive Datasets" [5].

Coding: The students were started off with skeleton code that translates `radb`-encoded relational algebra expressions to a `luigi`-managed workflow of MapReduce jobs. `luigi` [8] is a Python package for building pipelines of long-running batch processes. `luigi` supports several execution platforms, and among them, Hadoop. The students were already provided with this code. Thus, they could focus on implementing the code stubs for realizing selection, projection, renaming, and join.

Figure 3 shows the skeleton code for a `luigi`-task that implements the relational selection-operator as a Map-only job on Hadoop. All that remains for the students to do is to flesh out lines 25 through 29, having ready access to the selection predicate (see line 22) represented with `radb` datastructures. The `mapper`-function is invoked once for each line of input, which is then parsed into a key-value pair.

The input is formatted as shown in Figure 4, consisting of the relation name as the key and the JSON-encoded tuple as its value. Like in the original Hive, all intermediate results are stored in HDFS before they are processed by the next operator.

The code skeleton supports three mode of operandi: (1) Reading and writing to main memory only, and merely mocking a cluster-based execution environment. This makes unit testing easy, hermetic, and fast. (2) Reading and writing to local disk, rather than HDFS, and again mocking a cluster-based environment. This is the development mode, with quick turnaround times and the option to inspect all intermediate data in local files. Moreover, the development mode does not require an HDFS or Hadoop installation. Finally, (3) reading and writing to HDFS, and running on a Hadoop cluster in the Cloudera Quickstart VM. This was the intended production mode.

Switching between these modes with a runtime flag, the students experienced the pain of debug-

[2]We chose this involved syntax in preparation for chain folding, as shown in Figure 2, where code from different jobs is merged into stages. This allows us to track which parts of the code go where.

[3]Available at `https://www.cloudera.com/downloads/quickstart_vms/5-13.html`.

```
# SelectTask implements selection as a Map-only job.
class SelectTask(luigi.contrib.hadoop.JobTask):

  # The radb-encoded relational algebra expression
  # that this operator evaluates, serialized as a string
  # of the form "\select_{cond}(R)".
  qs = luigi.StringParameter()

  # Omitting some luigi/workflow-specific code.
  ...

  def mapper(self, line):
    # Parses the input line into a key-value pair,
    # where the key is the relation name and the
    # tab-separated value is a JSON-encoded tuple.
    relation, tuple = line.split('\t')
    json_tuple = json.loads(tuple)

    # Deserializes the query string into an radb expression
    # to gain access to the selection condition.
    query = radb.parse.one_statement_from_string(self.qs)
    condition = query.cond


    ''' .... fill in your code below ....'''

    yield("foo", "bar") # To be replaced with your code.

    ''' .... fill in your code above .... '''
```

Figure 3: The `luigi` skeleton code for the selection operator from relational algebra.

```
Person  {"name": "Amy", "age": 16, "gender": "female"}
Person  {"name": "Ben", "age": 21, "gender": "male"}
...     ...
```

```
Eats    {"name": "Amy", "pizza": "pepperoni"}
Eats    {"name": "Amy", "pizza": "mushroom"}
Eats    {"name": "Ben", "pizza": "pepperoni"}
Eats    {"name": "Ben", "pizza": "cheese"}
...     ...
```

Figure 4: The pizza data instance [7] encoded as key-value pairs. The key is the relation name, the value is a JSON-encoded tuple.

ging in a distributed environment: Just because the unit tests passed and everything worked fine in development mode is no guarantee that their implementation succeeds in the production environment (often due to careless use of global variables, or proprietary packages not available in the production environment). Digging through the logs and troubleshooting Hadoop turned out to be cumbersome, which in itself is a good learning experience.

2.4 Beyond Selection Pushdown

Scope: With the third milestone, the students had already built a working SQL-on-Hadoop engine. Yet since each MapReduce stage only evaluates a single relational algebra operator, and reads its input from HDFS, the runtimes are unnecessarily high.

Thus, in the fourth milestone, the students were asked to optimize their query engine. While the earlier milestones came with tight specifications, the students could now decide for themselves which optimizations to implement. As an incentive, the top ranking solutions would receive extra points.

It was stated as a requirement that for a predefined set of queries over TPC-H data, the students had to beat their milestone 3 implementation in 75% of the cases. These queries had been engineered such that the students would see the benefits of the optimization techniques discussed in class. The students were further provided with the cardinality estimates from the official TPC-H benchmark. For instance, execution of the following query

```
SELECT DISTINCT CUSTOMER.C_CUSTKEY
FROM CUSTOMER, NATION, REGION
WHERE CUSTOMER.C_NATIONKEY = NATION.N_NATIONKEY
  AND NATION.N_REGIONKEY = REGION.R_REGIONKEY
```

benefits from projection pushdown (provided the pushed projections do not remove duplicates), in combination with chain folding. Moreover, reordering the joins, so that the relations with smaller cardinalities (`REGION` and `NATION` have only 5 and 25 tuples respectively, whereas the `CUSTOMER` relation contains a multiple of 150K tuples, depending on the scale factor chosen when generating the data) are joined first. This effectively reduces the costs for storing intermediary results in HDFS.

As a practical means for capturing the effects of optimization, we measured the amount of intermediate data stored in HDFS. Of course, it would have been great to actually benchmark the students' solutions on a Hadoop cluster. Yet since the course was taught without any supporting staff, the instructor had to find a way to limit the administrative overhead for validation: In the development mode of *miniHive*, where all data is stored as local files on disk, measuring the data temporarily stored in HDFS can be realized with basic shell script commands. Also, the metric chosen is roughly aligned with the communication costs introduced in [5].

Independent Study: In preparation to the final milestone, several optimizations, some of them MapReduce-specific, were addressed:

(1) *Chain Folding.* The most "bang" for one's money was to be gained with rewriting the workflow of MapReduce jobs by merging several jobs into multi-functional stages. This approach is sketched in the original Hive paper [14], and has meanwhile been explored systematically in academic research, e.g. also motivated and described in [13] and benchmarked in [6]. This is considered a generic MapReduce design pattern also among practitioners [11].

We go by the terminology of [11] and refer to this strategy as *chain folding*. By chain folding, which can be as simple as collapsing sequences of Map-only jobs, we need to store fewer temporary files in HDFS. This evidently reduces the overall communication costs, and accordingly, the elapsed wall-clock time. In Figure 2, we show the physical query plan for our running example after chain folding. Now, renaming, selection and join are evaluated within a single stage (symbolized by the Unix pipe operator).

(2) *Projection Pushdown.* Projection pushdown only makes sense in combination with chain folding, provided that the pushed projections do not eliminate duplicates and therefore can be implemented as Map-only jobs. These can be merged in subsequent chain folding. Otherwise, adding blocking Reduce jobs drives up the communication costs.

(3) *Multi-way Joins.* Besides Reduce-side joins, we further discussed multi-way joins, covered in [5].

Coding: For the final milestone, the students were provided with a list of queries over TPC-H data, together with the cardinality estimates for this data model. The most successful student submission implemented optimizations (1) through (3) from above, as well as cost-based join reordering, heuristically joining the relations with lower cardinalities first.

2.5 Moving from *miniHive* to Apache Hive

Included in Cloudera Quickstart VM is an installation of Apache Hive, as well as Spark. Towards the end of the term, students interacted with these systems. By then, they had gained an appreciation for the scalability of Hive. Moreover, the students could now make sense of the output of Hive's EXPLAIN statements, and had an easier time understanding certain design decisions, such as Hive trying to avoid MapReduce jobs in query compilation. For instance, a simple exploratory query like "`SELECT * FROM Person LIMIT 10`" can be evaluated without spinning up MapReduce jobs, just by scanning a single chunk of the input file on HDFS.

Having implemented eager query evaluation in *miniHive*, the students now understood how lazy evaluation, as implemented in Spark, can make for a great interactive user experience.

3. TESTBED AND EVALUATION

Figure 5 summarizes the number of submissions. In total, 60 students participated in the final exam. The majority of these students also submitted a solution to milestone 1. A Python script was used to unpack the submitted `zip` files, run the unit tests, and to cross-check solutions with `pycode-similar`,

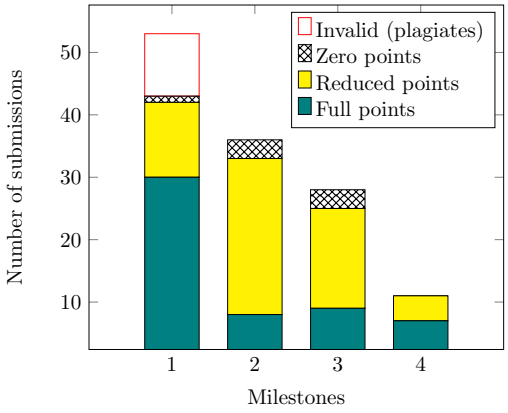

Figure 5: Submissions and points earned for the milestones in a class of 60 students.

Rank	Improvement
Top 1	4.6×
Top 2	2.9×
Top 3	2.8×

Figure 6: Improvements in milestone 4 over milestone 3 by the top-3 solutions.

a simple plagiarism checker.[4] This immediately revealed 10 submissions as plagiates, which their authors also admitted to.[5] In consequence, these students received no points for their submissions.

One milestone 1 submission was awarded no points, since the majority of unit tests had failed. A total of 30 submissions received full points, having passed all unit tests. 12 submissions further received reduced points, since non-public unit tests had failed. These tests checked for simple syntactic variations of the provided queries, and revealed submissions where students had not tested their code diligently. The students had been made aware that their submissions would undergo non-public unit tests.

Fewer students made submissions for the later milestones, which is owed to the fact that submissions were time-intensive, optional, and awarded only with bonus points. Nevertheless, the enthusiasm of the participating students remained high, and with milestone 3, still 40% of the students submitted a working SQL-on-Hadoop engine.

Figure 6 lists the improvements achieved by the top 3 submissions to milestone 4. As described earlier, we compare the size of data written into tem-

[4] `https://github.com/fyrestone/pycode_similar`.

[5] Apparently, code plagiarism is quite common in computer science education, a topic also noted by the press, e.g. [2]. It is a debate whether this problem is specific to our field, or merely revealed by tool-based checks.

porary files into HDFS against the milestone 3 implementation. As the specifications for milestone 3 were tight, all successful milestone 3 submissions had the same baseline regarding costs. The best *miniHive* engine achieved a 4.6× improvement, using all optimizations discussed in Section 2.4.

4. SUMMARY AND OUTLOOK

In their anonymous course feedback, the students described *miniHive* as work-intensive. When asked about the time spent on the individual milestones, the students reported anything between five and forty hours per milestone. It is up to debate whether this may be seen as evidence of the "factor 10 coder" effect, a controversial theory among developers stating that some coders are more effective by a factor 10. At the same time, the majority of students who participated in the project stated that they believed they had learned a lot. The students who completed *miniHive* where literally ecstatic about the improvements that they had achieved.

While the term project was not mandatory, the students who did well with *miniHive* also excelled in the final exam, since they had acquired a solid understanding of MapReduce processing and the associated communication costs. It is one thing to learn about the theory of query optimization, it's another thing to watch your own code perform the magic.

As a future feature for *miniHive*, it would be very instructive to add *partition pruning*, as also implemented in the first version of Hive from 2009: Provided that a Hive table is partitioned into several HDFS folders, based on attribute values (similar to building a traditional cluster index), Hive can ignore irrelevant folders in evaluating selection predicates. Indexing for Hadoop processing has, of course, also been explored in research, e.g. [12], so this would be an opportunity to integrate more recent research results into class. Experiencing the speedups achievable by indexing would be a further valuable learning experience.

The *miniHive* material for students, including the assignment descriptions as well as skeleton code and unit tests, is available at: `https://github.com/miniHive/assignment`.

To instructors, the complete course material, including a prototype and selected student solutions, can be made available upon request.

Acknowledgements: In 2006, my PhD advisor Christoph Koch taught a database systems course at *Saarland University*, where students built a native XML database that could evaluate a practical fragment of XQuery [4]. The *miniHive* term project is inspired by this experience.

5. REFERENCES

[1] A. Albrecht. python-sqlparse - Parse SQL statements, 2019. A Python package, available as open source at `https://github.com/andialbrecht/sqlparse`.

[2] J. Bidgood and J. B. Merrill. As Computer Coding Classes Swell, So Does Cheating. New York Times, May 29, 2017.

[3] J. Dittrich. *Patterns in Data Management: A Flipped Textbook*. CreateSpace Independent Publishing Platform, 2016.

[4] C. Koch, D. Olteanu, and S. Scherzinger. Building a native XML-DBMS as a term project in a database systems course. In *Proceedings of XIME-P'06*, 2006.

[5] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press, 2014.

[6] H. Lim, H. Herodotou, and S. Babu. Stubby: A Transformation-based Optimizer for MapReduce Workflows. *Proc. VLDB Endow.*, 5(11):1196–1207, July 2012.

[7] Jennifer Widom. Database Mini-Courses, 2014. Online course, available at `https://lagunita.stanford.edu/courses/DB/2014/SelfPaced/about`.

[8] Spotify AB. luigi, 2018. A Python package, available as open source at `https://github.com/spotify/luigi`.

[9] Udacity Inc. Intro to MapReduce and Hadoop, 2018. Online course, available at `https://classroom.udacity.com/courses/ud617`.

[10] Udacity Inc. Introduction to Python Programming, 2018. Online course, available at `https://classroom.udacity.com/courses/ud1110`.

[11] D. Miner and A. Shook. *MapReduce Design Patterns*. O'Reilly Media, Inc., 2012.

[12] S. Richter, J.-A. Quiané-Ruiz, S. Schuh, and J. Dittrich. Towards Zero-overhead Static and Adaptive Indexing in Hadoop. *The VLDB Journal*, 23(3):469–494, June 2014.

[13] C. Sauer and T. Härder. Compilation of Query Languages into MapReduce. *Datenbank-Spektrum*, 13(1):5–15, 2013.

[14] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, et al. Hive: A Warehousing Solution over a Map-Reduce Framework. *Proc. VLDB Endow.*, 2(2):1626–1629, Aug. 2009.

[15] J. Yang. RA (radb): A relational algebra interpreter over relational databases, 2019. A Python package, available as open source at `https://github.com/junyang/radb`.

Richard Hipp Speaks Out on SQLite

Marianne Winslett and Vanessa Braganholo

Richard Hipp

http://www.hwaci.com/drh/index.html

Welcome to ACM SIGMOD Record Series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at the 2017 SIGMOD and PODS Conference in Chicago. I have here with me, Richard Hipp, who won the 2017 SIGMOD Systems Award and the 2005 Google O'Reilly Open Source Award for SQLite. Richard has his own consulting firm, Hwaci, and his Ph.D. is from Duke University.

So, Richard, welcome!

Thank you for having me here.

It's a pleasure. Can you take off your glasses for a moment and show off what you have there?

Oh, you mean, this?

This black eye! It must have been a real battle to win that SIGMOD Systems Award!

Oh, no, this is nothing. You should see the other guy. No, no! There was no fight or anything. This is actually a farming accident.

A farming accident?

I went to visit my parents last week. I was helping dad put up some new fencing on the farm, and I was cutting through the fence, and when I cut the last wire the whole thing sprung up, and a piece of wire hit me right there above the eye. By the grace of God, it missed my eyeball and didn't put an eye out. But I'm fine; everything is fine now. But this just goes to show that programmers should stick to programming computers and should not try to grow their own food. Leave farming to professionals!

All right, that's probably easy advice to swallow for our audience.

So, SQLite is the most widely deployed database engine in the world! Take that Oracle! SQLite is in just about every modern phone, PC browser, car, you name it, it might even be the single most widely deployed software component of any type! My first question: How did you happen to write such an insanely popular piece of software?

It was entirely by accident. I was working as a consultant doing some really interesting problems, and I wrote a product for my client and it had to pull the data off of Informix. And that worked great. Except for sometimes, the Informix server would go down for reasons that were out of my control, and then I'd pop-up a dialog box that says, "Can't access the database." And, of course, it was not my fault but I got the support call because it was my software painting the dialog box.

So, I thought, well, why can't I have a database that just reads directly off the disc? And I looked around and there were none available. I thought, "oh, I'll just write my own, how hard can that be?" Well, it turns out to be harder than you might think at first, but I didn't know that at the time. But we got it out there and I just put it out as open source. And before long, I started getting these phone calls from the big tech companies of the day, like Motorola and AOL, and, "Hey, can you support this?", and "Sure!" And it's like, wow, you can make money by supporting open source software? Who knew?

So, I built a small team and we've been doing that for about 10 or 12 years now. So, it was an accidental database, I didn't intend to take over the world.

The accidental database… You know you're in big company with that? Because Mike Stonebraker, when he first got started he had the exact same thought, he said, "How hard can this be?" They say that knowledge is power, but I think often ignorance is power.

Right. If I'd known how hard it would be I probably never would've have written it.

So, why is it so popular?

You know, I don't really know. I mean, I can guess. Well, one, it's really easy to use. I mean it comes as a single file source code, and so people just plop the single file of source down into their application and recompile, and they have a complete SQL stack.

The database is one file on disc. So, it's really easy to email to colleagues. It's fast. It's really easy to use. It's small, got a small footprint. That's why it's real popular on cell phones and things, because especially in the early days they were really interested in saving every byte of memory they could.

How small is it?

Compiled for size, it's less than a half megabyte.

Ooh, and how many lines of code, vaguely?

I think it's about 120,000 source lines of code; 200,000 if it includes the comments, all in one big file. We don't edit that one big file, okay, we actually have a bunch of little files that we developed and then there's a build process that stacks them all together.

But that makes it very easy to deploy because it's just one file that you drop in the middle of your application. And so, if you go look at popular applications like Chromium or Firefox or these open source things, you will find sqlite3.c in the tree which just gets linked in with the rest of their product and then they've got a complete stack. They don't have any external dependencies, and it just works.

What were the main technical challenges you faced in creating SQLite?

Well, a technical challenge is apart from the fact that I didn't know what I was doing? Well, so, SQLite doesn't have a server, you know, it's just a library. And so, you make a call into the library and then it returns, and there's not a thread or a process hanging around behind to take care of all the housekeeping details normally associated with a relational database, like concurrency control, rolling up LSM trees, or feeding the write-ahead log back into the database. These are all normally taken care of by some background thread. We don't have a background thread. So, we have to handle all of that in the foreground without causing unnecessary latency on the library calls. And we just don't have something persistent there to remember the state of things. And it causes you to have to think about the design of the engine very differently from when you're working on a client-server install.

Wouldn't it be great if SQL or something like that were extended in a way that you could have relations, you could have graphs, you could have arrays, you could have JSON, and it all worked together seamlessly?

Most developers are using SQLite as just a key-value store. Should we be trying to convince them to take more advantage of SQL?

I think so. I think that's the whole point. Well, I see so many developers that think that SQL is just the wire protocol for talking to the database engine. They don't really grasp that the whole concept of the declarative language, it's gonna do lots of really cool things for them. Once people get that, it makes a huge difference to them, they become so much more productive.

But, you think about it, the whole programming world has become so complicated. I mean, when I started in this business all these decades ago, my first computer had 4k of RAM, total. And that included the video RAM. And so, I could know everything that was happening in my computer, but that's no longer possible. And now we get people and we're trying to teach them how to program in four years? And there's just so much to learn and so much has to be left out, and I think that the whole declarative idea is being completely omitted. And people come out and, you know, it's easy to kind of think about things in a key-value store, that's a very familiar concept. But nobody's ever really taught the value of a declarative language and how much work that can really save. How many thousands of lines of code in your application you can save by specifying the join in SQL, rather than pulling all the information in and doing the join in your application, which is what we see a lot of people doing in actual applications. Yeah.

So, yeah, I think that about 90 percent of the use of SQLite is a key-value type thing. But that other 10 percent, people are really using it to the max, they're really putting these complex queries in there. And I see some of those queries sometimes and I think, wow, is my database computing this, that's amazing! But that's the way it's intended to be used.

If you had the ability to redesign SQLite from scratch, what would you change?

Mostly just piddly details. I mean, because SQLite is used in so many millions of applications, we cannot fix mistakes that I've made in the interface. So, for example, the database file format uses Big Endian numbers. And now all the processors are Little Endian. And so, we could save a few cycles here and there if the database stored everything as Little Endian.

And there are a few quirks in the language… If you say "integer primary key" that means one thing, but if you say "integer" and then put "primary key" into column name it works slightly differently and it's just wrong. But we can't change that now because there's a lot of databases out there that do make use of that distinction and that would break compatibility. And we're all about preserving compatibility to the hundreds of billions of instances that are out there already.

So, yeah, we could do a few tweaks like that to improve the performance, but I'm really kind of happy with the overall design and how it worked out. I wish I could claim that I had this brilliant insight and foresight and was able to predict that that's the way it was gonna work out (it was dumb luck). But it worked out well in the end.

You say that SQLite has aviation grade testing. What does that mean?

So, we follow a design process that's inspired by DO-178B, which is a Spec followed by the FAA, actually I think it's been superseded now with DO-178C, but I'm told it's not much different. And it's a Spec that the FAA requires for safety-critical systems, safety-critical software systems in the aircraft. It's a very detailed design Spec about the processes you do in developing the software. And the key point is that you have to test

it to 100 percent modified condition/decision coverage (100% MC/DC). Which, basically, means you have to test the machine level such that every branch instruction has been taken and falls through at least once. It's an incredible amount of testing.

This started because years ago I had this idea that, oh, I'll come up with this test suite and then I'll sell it to people and make money. That didn't really play out. But what we found is when we did this and spent a year developing all these tests and, of course, tons of time since then maintaining them, is that the number of bugs just dropped dramatically. And it will amaze you how many bugs pop up when your software is deployed on two billion cell phones. And, yeah, I used to think that I could write bug-free software and it got put on cell phones and then, no…

But once we got that and got this aviation grade testing in place, the number of bugs just dropped to a trickle. Now we still do have bugs but the aviation grade testing allows us to move fast, which is important because in this business you either move fast or you're disrupted. So, we're able to make major changes to the structure of the code that we deliver and be confident that we're not breaking things because we had these intense tests. Probably half the time we spend is actually writing new tests, we're constantly writing new tests. And over the 17-year history, we have amassed a huge suite of tests which we run constantly.

Other database engines don't do this; don't have this level of testing. But they're still high quality, I mean, I noticed in particular, PostgreSQL is a very high-quality database engine, they don't have many bugs. I went to the PostgreSQL and ask them "how do you prevent the bugs"? We talked about this for a while. What I came away with was they've got a very elaborate peer review process, and if they've got code that has worked for 10 years they just don't mess with it, leave it alone, it works. Whereas we change our code fearlessly, and we have a much smaller team and we don't have the peer review process. So, that's the basic difference. People hear aviation grade testing, that means it must be bug-free. Not really. It's low bug, but there are bugs. The key benefit is that it allows us to move fast and aggressively and make big changes to the code without fear of breaking things.

Now, why do you want to make big changes to the code if you've already written something that clearly is working for almost everybody?

Well, because, you know, you can always make improvements to the query planner. The query planner is an AI, and so there's no perfect solution. And so, no matter what we do there's gonna be somebody come along and find some query that it comes up with a bad plan for. And then a lot of times we have to make pretty radical changes to the query planner in order to come up with a good query plan for some bizarre bit of SQL. Of course, the first question we always ask is, why did you want to do this? Who would ever issue such a query? And it's often machine-generated queries.

It was an accidental database, I didn't intend to take over the world.

So, it occurs to me that one factor in your success is the fact that only 10 percent of the billions of installations are doing the SQL queries. Because if all those key-value store people were actually making full use of the abilities of the SQL language, heaven knows what kinds of strange queries they'd be asking…

Well, I'd suppose but, no, 10 percent of a billion is still a lot!

It is.

And the people who are doing the elaborate queries are companies that use it really heavily. Customers! I wish I could name them for you.

That's fine. We don't wanna know, we don't wanna know. So, if commercial database engines don't undergo nearly that amount of testing, why do you work so hard at testing an open source product that you give away for free?

Well, that's a good question. You know, the intense testing allows us to keep a really small team. The product is free, and so, we make all our money from support. And we've been following of the open source world and people are saying "you can't make money selling support." And they're basically right, you can't make a lot of money. So, a typical start-up would be 30 to 50 guys in an office building in San Francisco. And you cannot make enough money selling support to support that operation. But we only have three developers, including me. And we're a distributed company, we don't have office space, everybody works from home. We keep our overhead really low. And we only have three people. And so, with only three people this high level of testing allows us to produce a quality product without having a lot of eyeballs on it.

Wait, it almost suggests that a giant company could also get by with three people on some major product. If they did mega testing.

Possibly so. You know, the DO-178B Spec is very detailed. And what I've seen is a lot of companies trying to implement it and they get very legalistic about it. And, I mean, it's a really great thing if you do it well, but if you don't implement it carefully it can just become bureaucratic overhead. And then it actually multiplies the number of people rather than reducing them.

Okay. Got it.

It's a tricky thing; it's a knife edge there.

People use SQLite in some places where you really don't want a hacker to be able to get at the data. So, beyond aviation grade testing, are there other considerations you give to the security?

Yes, we do. Well, so the MC/DC testing is great for verifying that sensible queries that normal programmers would write actually work and give the correct answer. But to prevent attacks we use fuzz testing. And there are a number of great products out there that do this for us.

I think a lot of our audience might not know what fuzz testing is.

Fuzz testing is when you take a library or a product and you start pounding it with just seemingly random inputs and trying to get it to break. You're not trying to see if it gets the right or wrong answer, you just wanna break it, you wanna get a segmentation fault error or something like that. And that's an opportunity to break into the system.

And so, there are some researchers at Google, and what they've done is they have these fuzz testers where they start with a random input. But they also instrument the source code, or the object code. And they monitor the path through the object code that that test case took and it finds new behaviors, it learns from that. When it finds a new path it says, "oh, I'm gonna keep changing that mutation" and it finds new paths to the codes. It's a very powerful thing. It's only been out for two to three years. And it will find an amazing number of the bugs, even in very well tested software.

So, an example of this is they took a jpeg library and they started it with an empty file and started fuzzing it. And the fuzzer actually discovered valid .jpeg files. That's the power of the system. So, we feed these sorts of things into SQLite. Because it's a library, it's very easy to do this and we can give it thousands and thousands of queries per second, fuzzing them, trying to find bugs.

And when we first started doing that we did find a dozen or two dozen ways of crashing it. But since then, and we continue to test, we've fixed those and we haven't had any more problems.

So, SQLite can be safely used as, for example, a file format transferring data from across the internet. And you can receive a SQLite file from an untrusted source and bring it up and be confident that it's not going to –

Be a trojan horse…

Exactly.

So, what's the name of the Google tool for fuzz testing?

The first one was American Fuzzy Lop. And then the following one to that was OSS-Fuzz.

(Laughs) I laughed because the first name American Fuzzy Lop definitely sounds like the name of a kind of funny rabbit.

That's where he got the name. I won't try to pronounce the developer's name, it's Michael[1] and he's Polish. I will totally mispronounce his name so I won't try to pronounce it. But he wrote it and he picked that name because of the pun.

So, vaguely, how long would it take, when you've made some changes to your source, to run through your entire set of tests? Just vaguely.

Well, on one platform, because we run it on multiple platforms.

No, just one.

If it's a fast work station we can, running on multiple cores, we can do a complete set of tests in about 12 hours. But then we also run on multiple platforms, including some slow ones, like phones and that sort of thing. Some antique hardware, like ancient Mac Books that are still using Power PC processors, so we can test that it runs on Big Endian as well as the Low Endian. And so, it normally takes us about three or four days to do a complete test. But we can do full coverage testing, the 100 percent MC/DC testing in about three minutes. And so, after any change, we always do that. But then we have these long soak tests that do a lot of additional testing, and that's what takes a large amount of time.

[1] Editor's note: His name is Michal Zalewski.

Why did you choose the license for SQLite to be public domain rather than BSD or MIT?

Well, of course, technically, public domain is not a license, but the absence of one. Actually, SQLite Version 1 in 2000 used GDBM[2] as the storage engine. And that's a key-value storage thing. And it's GPL, and so SQLite Version 1 was GPL, it had to be because it was linking against the GPL library.

But GDBM is only key-value, I can't do range queries with it. Then I said, "I'm gonna write my own B-tree layer". And so, I wrote my own and at that point it became all my code and I thought, well, what license shall I do? And I thought, well, you know, BSD, MIT. And I thought, well, what's the point, why not just say it is public domain? And so, I put it out there as public domain. People can do whatever they want, I have disavowed all copywrite to it.

I would love to see people spend more time researching query languages as opposed to storage engines

And that was cool and that worked great. What I later learned is that the ability to put something in the public domain is kind of unique to countries that follow British common law. Other countries don't allow people to disavow their rights to the code. And so, it's a problem in that sense.

But it's to tradition and we've stuck with it. It's worked well and people understand it, and it makes people confident that they can use it their products and not have to worry about legal repercussions. Some companies still do worry about that because, I mean, anybody can grab some piece of code from somebody and put it out in the internet and say, oh, this public domain, you know.

So, another way that we do make money is we actually sell licenses for it. Actually, it's not a license, it's a warranty of title. It's an official document that says we do have the right to place this in the public domain and we will accept legal responsibility if anybody comes forward and accuses you of stealing it. And companies send us money for this piece of paper. And that's what we use to fund the development.

Okay. Anything to keep those lawyers satisfied. And if you were starting again today, would you still go the public domain route?

It's hard to say. I mean, it's fun to be unique in that, but practically speaking the two-clause BSD license or MIT license, accomplishes the same thing, and doesn't run into the problems that are in non-British common law jurisdictions. So, that would probably be a better choice.

After they graduate, most computer science grad students go to the university, or a big company or a start-up. So, what led you to be a consultant instead?

Oh, that's a long story. Because I've temperamentally ill-suited for working in the –

Corporate life?

Yeah. I don't play well with others.

Oh! The small team is just a matter of preference!

It is. I really like working for myself. And one reason we haven't grown to a big team is that there's no way in the world that I could manage a big team. And the people I have working for me are great and we get along well, and I think trying to manage 30 to 50 people would really be a disaster for me. I really enjoy working for myself.

Back in 1992, when I took my Ph.D., there were 500 applicants for any tenure track position. And I decided "I'm not gonna get in that rat race." And so, I started the consulting thing and it's worked out really well. I get to set my own hours, you know because when you work for yourself you can work any 80 hours of the week you want, work from home, work in projects that I wanna work on. And it's really worked out well. It's been a dream job. All of us on the team have really enjoyed working on SQLite.

Yeah, it sounds like if you graduated in a year when it was a seller's market, buyer's market, I'm not sure which direction, you might've taken a completely different path!

Well, I may have. But I guess it's providential that it worked out as well as it did.

Yeah! For everybody.

[2] https://www.gnu.org.ua/software/gdbm/gdbm.html

Yeah. I'm very happy.

But for someone who's graduating today, what considerations do you think should make them choose a consulting path?

Well, you get to choose your own destiny. You don't have to deal with the big company or the big university bureaucracy and the politics that comes with that. Maybe I'm sort of an urban prepper you might think of, in the sense that I wanna live off grid. Not really! Not really! It's just that I wanna do my own thing. No, I actually enjoy very much having the conveniences of modern city life. I don't wanna go off grid. But the concept is similar in that I want to decide for myself what I'm working on. I want the freedom to move around from lots of different customers.

You know, one really great thing about working in this is I've had the opportunity to visit so many different businesses and see the really different cultures there are in all these different places all over the world. And I can't really explain it unless you've experienced it, but you go to one company and see how they operate and how many people interact, and you go to a different company and have a totally different culture there. And then you step back and look and that culture kind of comes out in their products, and you can see it after you've visited their engineering facilities.

Interesting.

It's very interesting. And I don't know of any way of describing that other than just say go there and see for yourself and then you'll understand.

So, even inside the banks in Charlotte where you live, all those big banks, can you see the difference in culture in the products that come out of these giant banks?

A little bit. I don't have a whole lot of insight into the banks there in Charlotte. I'm thinking of like visiting, well, companies that you know well like Facebook and Google and Apple.

Oh, computer companies?

Yes. Motorola, AOL, these sorts of things. I get opportunities to go and visit and talk, and give talks there, and meet with the people and see their environment. And it's really exciting to be able to see all of these differences, coming in, not have to be a part of that company, but get a glimpse into their culture and see how they're all very different. You'd think that a bunch of companies all in the Bay Area would be very similar, but they're not, they're really very different. And when you travel to other countries and do this, it gets really, really different.

Yes, definitely. So, you don't see an evolution toward a single culture at computer companies? There's no convergence, it sounds like. If anything, it's divergence.

No, everybody pretty much copied Google's idea of giving the people free food, okay. That was a big win, apparently, and everybody copied that. So, there are some ideas that get around. And so, maybe you're right, maybe there was a lot more diversity a decade ago than there is today. But still, even today you can see a big difference in the different companies.

> ***SQL is the language that everybody loves to hate.***

Cool. Among all your past accomplishments, do you have a favorite piece of work other than SQLite?

I'm kind of proud of the version control system that we wrote specifically for SQLite called Fossil[3]. So, SQLite started out using CVS because back then that's what everybody used. But CVS is great if you had to use what came before CVS, you know. Some people bad mouth CVS, I know that those people never had to use what came before. But it has its limitations, and so we wanted to go with something better, and the other options weren't really making it for me. So, I wrote my own and called it Fossil.

And it's interesting in that Fossil stores all of its data in an SQLite database, and SQLite is controlled by Fossil, so we have this recursion here – if something breaks the whole project sort of collapses. It's a house of cards. But it works really well and it's been really flattering that thousands of other people have picked it up and started using it. So, it's a distributed version control system like Git. Nowhere near as many users, orders of magnitude fewer. But those who do use it are really enthusiastic about it.

And it keeps me in touch with the application development side of things. Because if you're developing just the database engine for so long, you get heads down and you don't see what's happening. But I can go out and work on Fossil and it's helping so many people, and then I also get to work on an actual application that uses SQLite and understand the pain of

[3]https://fossil-scm.org/home/doc/trunk/www/index.wiki

the users who have to deal with the interfaces that I design. And that's very good.

Eating your own dog food.

Yes. We're very much into dog food in SQLite.

If you magically had enough extra time to do one additional thing at work that you're not doing now, what would it be?

Oh, I've got a long list. But my #1 thing right now I think would be a new version control system which the working name is Fit, it's a combination of Fossil with Git. Uses the Fossil user interface but it uses the low-level file format of Git. So that then you can work with Fossil's interface but push and pull to legacy Git users. And I think that would be huge.

Why? What's better about the Fossil interface?

Well, it's not the perfect interface, but all the users say "oh, the Fossil interface is great, it's so wonderful, I hate having to use Git." But then they're compelled to use Git because everybody else in the world is and they wanna collaborate. So, I think that by merging these two producing Fit, Fossil plus Git, that would be a really big win for a lot of people.

Sounds like it.

It sounds like a lot of work, too.

Maybe… Do you have any words of advice for fledgling or mid-career database people?

I would love to see people spend more time researching query languages as opposed to storage engines. I mean, storage engines are a very important thing, we need to work on that, but it's like query languages are ignored.

Do you mean like changing SQL or do you mean like…

Enhancing SQL. SQL is the language that everybody loves to hate. And of course, I guess, everybody in the database world has at one point or another tried to come up with a better SQL. I know I've tried multiple times myself with indifferent results.

But the relational model is great and it'll represent anything, but there're some problems that just work out better with like a graph model or an array, or something like that, or JSON. Wouldn't it be great if SQL or something like that were extended in a way that you could have relations, you could have graphs, you could have arrays, you could have JSON, and it all worked together seamlessly? And that would be really, really amazing.

If you could change one thing about yourself as a computer scientist what would it be?

Well, I think we'd all like to be smarter, right? The ability to communicate better. I don't know. I've had such a blessed career, truly. I mean, I fell into this, I didn't plan to be a database guy, that was never in my career plans it just happened, but it's been an enormous amount of fun. And if I were to design it I'd mess it up. So, I'm just gonna go with what we've got.

Sounds good. Thank you very much for talking with us today.

Thank you for your time.

The SIGMOD 2019 Research Track Reviewing System

Anastasia Ailamaki[†‡] Periklis Chrysogelos[†] Amol Deshpande[¶] Tim Kraska[§]

[†]EPFL, [‡]RAW Labs SA
{firstname}.{lastname}@epfl.ch

[¶]University of Maryland
amol@cs.umd.edu

[§]MIT
kraska@mit.edu

1. INTRODUCTION

While organizing the submission evaluation process for the SIGMOD 2019 research track, we aim at maximizing the value of the reviews while minimizing the probability of misunderstandings due to factual errors, thereby valorizing impactful ideas. The objective is an educating and rewarding experience for both the authors and the reviewers.

The actionable goals are:

1. Maximize review depth and breadth. For depth, optimizing the assignment of papers to reviewers is of key importance; "low confidence" reviews should be few to none, in order for reviewers to provide extensive and useful comments to the authors. To cover the breadth and to address controversial issues, recruit as many reviewers as needed to converge to a unanimous set of comments.

2. Ensure that all submissions are treated equally fairly by experts in the respective domains.

3. Obtain as much input from the authors as possible during the process. Enabling author feedback is the key step in the process.

4. Allow re-evaluation of papers with non-critical flaws through revisions.

The better part of the infrastructure is devoted to optimizing reviewer assignments and to orchestrating the first set of reviews and discussions, both of which are critical parts in the entire process. The rest of this paper describes the process we set forth and our experiences from running it.

2. REVIEWER ASSIGNMENT

We address two issues in the reviewing process.

PC member load and responsibilities. Reviewer load and responsibilities must be clear and predictable. Some research areas are at times more popular than others; when selecting reviewers, we make sure that the PC composition reflects topic popularity, albeit not always perfectly. Trying to achieve uniform load across all reviewers therefore inevitably results in non-expert reviews. In addition, after reviews are submitted, a discussion starts which ends at the very end of the process with a decision which is summarized in a meta-review and is supported by final, polished reviews. Orchestrating the discussion and ensuring final review quality is a major responsibility and a critical step toward an unanimous and educated decision. Top database conferences employ a two-level reviewer hierarchy: A large set of reviewers who provide the reviews and a smaller set of meta-reviewers who act as area chairs for a subset of papers (typically four times as many as the average reviewer load). The advantage is that the work is distributed and the quality of the final result is increased. Nevertheless, the meta-reviewers can only help resolve a disagreement if they have read the paper in question themselves, which is only partly possible and typically happens under severe time constraints (after the end of the review period).

Reviewer assignment. The typical method for reviewer assignment relies on topic relevance: reviewers of a paper with intellectual contributions in a certain topic provide useful comments if they work on the same or a related topic. Feedback from recent conferences, however, shows that the evaluation methods play a significant role in the appreciation of a paper's contributions. The evaluation of a paper's contributions is typically based on theoretical proofs or on observations from experiments with implemented systems. Experience shows that a paper is evaluated most fairly when at least a subset of its reviewers are experienced on the evaluation methods the paper uses.

The rest of this section describes how we structure the program committee work in order to resolve the aforementioned issues.

2.1 PC member load and responsibilities

The number of assigned papers may vary across program committee members – this is due to unexpectedly disproportional number of submissions in a subset of topics. This is a fact that reviewers need to be alerted about. Reviewers also need to be prepared to provide impromptu reviews in severe conflict cases.

A subset of reviewers form the **core committee** and are responsible for (a) providing reviews for their assigned papers, monitoring discussions, ensuring high review quality, and writing meta-reviews as needed. Core committee members (aka meta-reviewers) are assigned about double the number of papers that **regular committee** members are assigned; the latter are responsible for providing reviews for their assigned papers and for participating in discussions. Reviewer invitations are clear about load and responsibilities. In SIGMOD 2019, about 25% of the reviewers are core committee members.

For review assignments, we collect a number of attributes that describe reviewers and submissions to enrich the data provided by the conference management tool and provide better assignments.

2.2 Reviewer assignment preparation

The goal of assigning papers to reviewers is to maximize the average reviewer expertise per paper, and minimize the number of non-expert reviews. To maximize the level of expertise per paper we characterize papers and reviewers with respect to the style of evaluation and contributions.

Paper characterization. The SIGMOD 2019 paper submission form asks authors to characterize their paper as either **systems** or **non-systems**. The call for papers describes **systems papers** as:

- Papers that describe an entire new system, covering, e.g., the system architecture and design issues or experiences learned from building the system.
- Papers that extend an existing (open-source) system with new or more efficient functionality. Such papers may add new functionality to Spark, Hadoop, PostgreSQL, etc. The motivation may be to better support new applications.
- Papers that concern specific aspects of a system or systems. Such papers may concern storage management, query processing, indexing, transaction management, access control, authentication, etc.
- Papers that concern systems support for new hardware, e.g., multi core, SIMD, NUMA, HTM, SGX, GPU, FPGA.
- Papers that analyze system performance.

The systems/non-systems annotation helps ensure that system papers are assigned not only to reviewers who are experts on the topic of the papers but also have extensive expertise in writing and evaluating systems papers.

Reviewer categorization. Similarly to papers, each program committee member is annotated as "systems" or "non-systems" according to their own published work. As a result, each of the core and regular program committees is further divided into systems and non-systems subcommittees. (To preserve reviewer anonymity, the systems annotation is not announced on the website.)

To match reviewer and paper research areas, we use both the Conference Management Toolkit (CMT) [4] and the Toronto Paper Matching System (TPMS) [3]. Upon acceptance of the invitation to the program committee and prior to abstract submission, all reviewers mark their research areas on CMT and upload a set of their own representative publications on TPMS. After paper submission, all submissions are uploaded to TPMS that outputs baseline matching scores for each reviewer-paper combination.

Using TPMS scores as a similarity measure. TPMS analyzes the submitted papers and the reviewers' uploaded list of papers and produces a score based on their topics similarity, extracted based on word counts and LDA [2].

Preferred reviewer similarity measures. We solicit proposals for reviewers from both paper authors and core committee members as follows. At submission time, authors can optionally propose "preferred" reviewers (inside or outside the program committee) for their submissions, which we use opportunistically as an additional indicator of the quality of a reviewer-paper combination. Similarly, at the first submission cycle, we asked meta-reviewers to suggest reviewers for each of their assignments. Handling meta-reviewer input, however, was manual and incurred more overhead than its associated benefit, so we dropped this option in the second submission cycle. Nevertheless, meta-reviewer input can prove useful if the suggestions are incorporated into the process automatically.

Conflicts of interest. We augment the conflicts of interest inserted by authors and reviewers through CMT with additional ones based on the mined collaboration graphs provided by AMiner [1]. This was

a useful step as none of the COI lists was a superset of the other: AMiner is not aware of ongoing collaborations and conflicts outside of what can be inferred from published work or employment status, while authors may forget to register some conflicts.

2.3 Meta-reviewer assignment

We assign meta-reviewers to submissions using an expanded variant of the integer program proposed by Taylor [7] and reused in TPMS [3]:

$$\begin{aligned} \underset{y_{rp}}{\text{maximize}} \quad & \sum_r \sum_p \hat{s}_{rp} y_{rp} \\ \text{subject to} \quad & y_{rp} \in \{0,1\}, && \forall r,p \\ & \sum_r y_{rp} = R, && \forall p \\ & y_{rp} = 0, && \forall (r,p) \in COI \\ & \sum_p y_{rp} \leq S_{max}, && \forall r \\ & S_{min} \leq \sum_p y_{rp}, && \forall r \end{aligned} \tag{1}$$

where, y_{rp} is constrained to 0-1, with $y_{rp} = 1$ if and only if reviewer r is assigned to paper p, R is the number of required reviews per paper and S_{min}, S_{max} is the minimum and maximum number of assignments per reviewer, $\hat{s}_{rp}$ is the gain from matching reviewer r to paper p and COI is the set of conflicts of interest. We observe that for large values of S_{max} and as there is no lower bound, two groups of reviewers are created: the "misunderstood" ones who are assigned a very small number of papers and the "rockstars" that are assigned almost S_{max} papers, with almost no one in between. To avoid bipolar groups, we augment Taylor's formulation by setting a lower bound S_{min} for the number of assignments per reviewer. For meta-reviewer assignments, r is simply restricted to the set of meta-reviewers and, as we assign only one meta-reviewer per submission, $R = 1$.

For SIGMOD 2019, we use:

$$\hat{s}_{rp} = s_{rp} w_{rp} + refinement_{rp} \tag{2}$$

where s_{rp} is the TPMS score, $refinement_{rp}$ is added to allow penalizing or embracing matches after manual inspection of the results and w_{rp} is a factor we calculate based on the collected information:

$$w_{rp} = 1 + \alpha f\left(Rank_{rp}\right) - \beta T_{rp} - \gamma P_{rp} - \delta A_{rp} \tag{3}$$

where $f\left(Rank_{rp}\right)$ depends on the position, if any, of reviewer r in the list of proposed reviewers by the authors and T_{rp}, P_{rp}, A_{rp} are indicator variables described in Table 1.

We select constants α, β, γ and δ by trying different combinations and subjectively evaluating the results based on our experience, until we find a set that produces overall good assignments. (Ideally, however, these parameters should be tuned using data collected from previous editions of the conference and community feedback. The same holds for selecting a linear model for combining the factors in Equation 2: we use a simple model, but collecting previous conference data allows for more informed decisions.) Using Equation 1, we produce the meta-reviewer assignments and then go over them to detect and correct corner cases. For the meta-reviewer assignments we only consider reviewers from core program committee.

Symbol	is 1 if r and p do ... otherwise it is 0
T_{rp}	differ in system/non-system trait
P_{rp}	differ in registered primary areas
A_{rp}	have no common registered areas

Table 1: Indicator variables used in modeling

2.4 Reviewer assignments

For reviewer assignments, we use a similar process as for meta-reviewer assignments but now we consider both core and regular program committee members. For starters, we mark meta-reviewers also as reviewers of their assigned papers. For regular reviewers, we solve once again Equation 1 but this time with r going over the set of light reviewers and R set to the target number of reviews per paper (minus one, as the meta-reviewer is providing one of the reviews). Also, for regular program committee members, S_{min} and S_{max} change due to the higher number of regular program committee members, compared to core ones.

As in the first submission cycle we solicited reviewer suggestions from the meta-reviewers, an extra factor in the calculation of w_{rp} incorporates the suggestions into the formula. Finally, we readjust the weights to tune the relative importance of the features. As both the area and the system traits are input to the process, the set of reviewers per paper may be a mix of both systems and non-systems reviewers. For example, a paper on data cleaning which evaluates the performance of a system will be assigned to both systems and non-systems reviewers who work in the field of data cleaning.

2.5 Timeline

The reviewing process of each submission cycle is divided into four phases as shown in the Gantt chart of Figure 1.

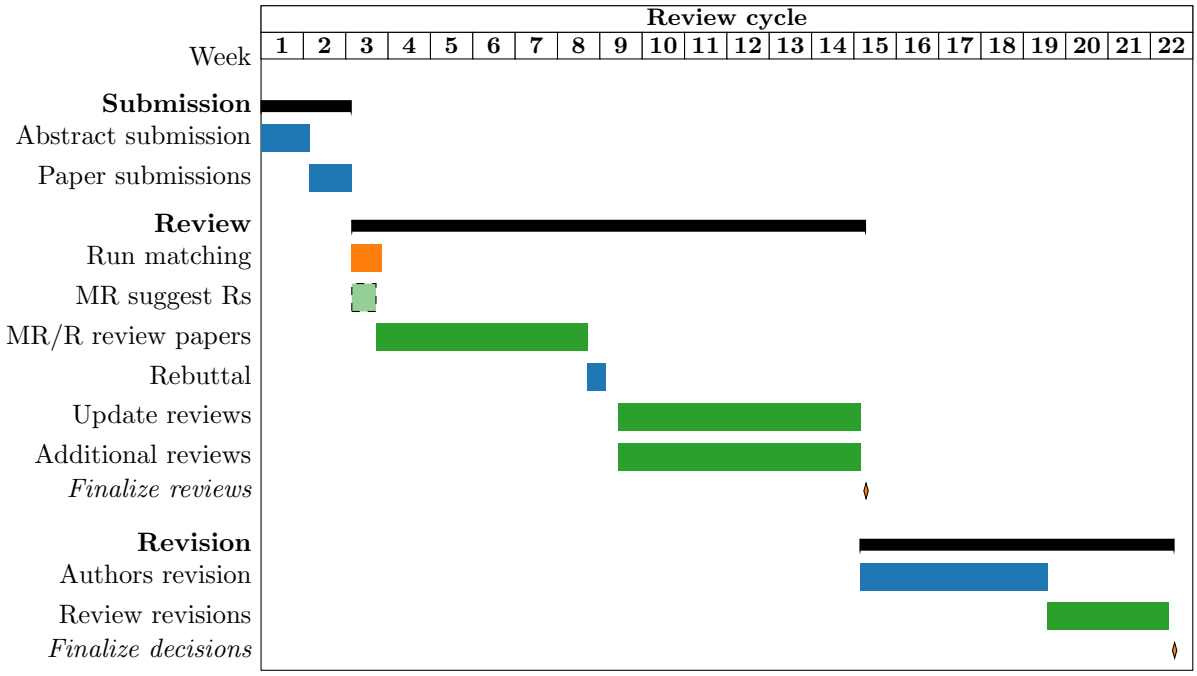

Figure 1: Approximate timeline of reviewing the submissions of each research submission cycle

We augment the submission form with a mandatory field for the authors to characterize their paper as system or non-system, and with an optional field to supply a list of suggested reviewers. Before the submission deadline, we ask the reviewers to upload a sample of their work on TPMS and we manually annotate reviewers as systems or non-systems.

After the submission deadline, we run TPMS to obtain the scores and solve the optimization problem 1, using the collected information to create the meta-reviewer assignments. Then, we manually review the assignments to find sub-optimal assignments and fix them using author suggestions, or secondary matches generated from TPMS. If no reviewers are found, we solicit suggestions from the research community. Afterwards, we upload final reviewer assignments to CMT and launch the review phase. We allow two days for reviewers to express concerns about their eligibility to review papers, and reassign the (typically very few) papers.

Author feedback. Review time is three to four weeks. During that time each reviewer (including meta-reviewers) reads all papers assigned to them and writes their reviews independently. At the end of the reviewing period (plus a few days waiting for late reviews to come in), meta-reviewers initiate discussions. At the same time, we invite author feedback in order to avoid misunderstandings of the text and clarify factual errors. We allow 72 hours for authors to read the (draft) reviews and provide short feedback on factual errors; the time is more than sufficient considering that clarifying any factual errors in the reviews should not take more than a few hours (author feedback is not a revision). In addition, we allow authors to express sensitive issues about the reviews through a new field in the feedback form for specific reviewer complaints which is only visible by the chairs. The meta-reviewers (and chairs) ensure that author feedback is taken into account in the discussions, and the chairs inject themselves into the discussions when a red flag is raised from the authors (through the feedback or via email) or from the reviewers, or when there is a significant disagreement among the reviewers (typically chairs are alerted to such cases by the meta-reviewers).

Assigning additional reviews. If a paper is headed for rejection and at least one reviewer has rated it "weak accept" or more, we invite two additional reviews. We select the additional reviewers as before, i.e., using author suggestions, or secondary matches generated from TPMS, or suggestions from the research community. Reviewers have a one-to-two week time period to review the (typically very few, if any) additional papers.

Review refinement. After all reviews are submitted, reviewers discuss and converge to accept or reject the paper or invite a revision, and the meta-

reviewer summarizes the rationale for the decision and revision points (if applicable). An important step before releasing the reviews and meta-review is to carefully polish the reviews to consider author feedback and discussion points, as well as to ensure that all reviewers agree with the meta-review.

Revision. Authors of papers that have been invited for a revision have approximately a month to submit the revised manuscripts along with a letter detailing the main revision points. The reviewers have a couple of weeks to ensure that the revised papers address all points in a satisfactory way. The final decision is to accept or reject the paper. If a paper addresses the revision points but there are still some less significant issues to be fixed, we accept the paper with shepherding (with one of the reviewers acting as a shepherd and communicating with the authors directly). The authors send an updated version of paper to the shepherd a few days before the camera-ready deadline, and the shepherd ensures that all additional points have been addressed and clears the paper for inclusion in the proceedings.

Submission cycles. There are two submission cycles in SIGMOD 2019, one with deadline in July and the other with deadline in November. The described process was repeated for each cycle.

2.6 Implementation

We use CMT as the conference management tool and both the reviewers and authors use this platform. To apply our reviewer assignment process, we extract the required information from CMT and submit our queries for TPMS scores using CMT-TPMS integration.

The integration with our implementation is done by downloading the corresponding files from CMT and incorporating our own files, such as the AMiner generated COI list and our systems/non-systems annotated list of reviewers. We query files with multiple formats in order to generate the $\hat{s}_{rp}$ matrix, as CMT exports almost each part of the data in a different format: xml, tsv, csv, lists per tuple (e.g., list of authors in a submission) and semi-structured text (e.g., author suggestions for reviewers). Then, we solve Equation 1 using CVX [5] and convert the solution to assignments. Finally, we create different views of the assignments and share them with program chairs to inspect a subset of the assignments, excluding from their views and redistributing among them conflicting papers. For simplicity, we flush views into spreadsheets.

Any changes are introduced back to our tool. We use $\mathit{refinement}_{rp}$ to lock assignments across iterations as well as replace some of them. As allowing the integer optimization program to re-run after small manual changes of the weights can produce ripple effects and thus require to recheck every assignment, currently we only use $\mathit{refinement}_{rp}$ for adding and removing assignments rather than changing a weight.

The final assignments are converted into XML files and uploaded to CMT.

3. REWARDING OUTCOMES

Setting up and operating the SIGMOD 2019 reviewing infrastructure for the research track was an end-to-end exciting and rewarding experience. Here we briefly report on the rewarding outcomes and some issues which would make future realizations of our methods a lot easier.

- The number of reviews and average expertise level per paper were improved. We had 3.36 reviews per paper on average (4.64 reviews per rejected paper with at least one weak-accept). Average expertise level[1] per paper was 67.4%.
- Systems and non-systems papers had the same acceptance rate (22.6% vs 20.1%, respectively; calculated over papers of the same trait).
- Reportedly, reviews were of generally higher-quality. Figure 2 shows that 77% of the reviews were made by knowledgeable (50%) or expert (27%), on the topic, reviewers. Only 2% of the reviews had low confidence. Lastly, the distribution of expertise level across system and non-system papers was almost identical, (within a 5% difference).
- Having meta-reviewers actually read and review the papers ensures that the core committee members are aware of the technical details and are involved in discussions more actively than if they are engaged at the end of the review period.
- Establishing author feedback, revisions, and shepherding improves two-way communication between reviewers and authors significantly.
- Allowing authors to express complaints confidentially prevents and defuses subsequent author-reviewer conflicts.

[1]Where 0%, 33%, 66% and 100% correspond to reviewers claiming to be unfamiliar, somewhat familiar, knowledgeable and experts, respectively, with the paper's topic

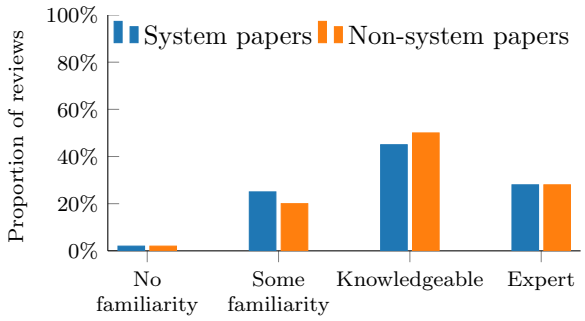

Figure 2: Distribution of review expertise level, as reported by reviewers, over papers in the same author-annotated category (system/non-system). Non-system papers were twice as many as system papers.

4. POINTS FOR IMPROVEMENT

We list the issues which need to be addressed in future editions of the process.

- The most serious time sink was the solicitation of late reviews and trying to reach non-responsive reviewers, through multiple reminders sent via CMT and personal email. Even a handful of late reviewers throws the process off, while non-responsiveness is extremely problematic.
- Some papers are borderline between systems and non-systems; some authors mis-classified their papers. Maybe an option to select "in-between" would have mitigated the problem.
- People change affiliations and thus emails often, which causes problems as CMT uses email addresses as IDs.
- We had to ask CMT many how-to questions. CMT responds swiftly within one to two days. Nevertheless, updated CMT documentation and a more accessible API would facilitate tasks enormously.
- Reviewers need to be educated to provide full citations and to be careful with asking for comparisons with non-peer-reviewed publications such as ArXiv papers. Reviewers should definitely alert authors to such publications, but asking for a head-to-head comparison against recent, non-peer-reviewed work is typically unfair to authors and needs strong justification.
- Re-submissions: If a reviewer receives the same submission that they reviewed for a prior conference, they should be able to mark the paper as a re-submission (visible only to the Chairs to avoid biasing other reviewers). This flag gives the chairs options to appropriately act on it. For example, the chairs might (1) assign an additional (un-)biased reviewer or, (2) in case the paper really has not changed, they may allow the reviewer to resubmit the same review. Another idea is to allow the author to submit the previous reviews with the paper including a description of how they addressed the reviews. This might help papers that are highly controversial and might give the reviewers additional points of view.
- TPMS scores generally worked well, but there were definitely papers and reviewers for which those scores were way off. More specifically, we observed that:
  - The program chairs had to manually check all the reviewer-provided papers in TPMS and manually upload papers for reviewers who did not do it on time. The very helpful CMT and TPMS teams make it easy to send papers from CMT to TPMS and retrieve scores and proposed assignments. Nevertheless, synchronizing the reviewer emails between the two systems and checking whether reviewers have uploaded papers is done through email correspondence with the TPMS team.
  - For some reviewers, the scores result repeatedly in clearly sub-optimal assignments. TPMS works off of a set of representative papers that are uploaded by the reviewers. However, some reviewers upload papers which misrepresent their expertise. We had at least 2 cases for which the assigned papers were far outside the expertise because of that issue. We fixed those cases manually.
  - TPMS logic is counter-intuitive at times: a senior researcher who works on five different topics in depth may upload two papers for each topic. By contrast, a junior researcher who works on one of these topics uploads ten papers on that topic. TPMS appears to give a higher matching score to the junior researcher. The consequences are only mitigated through manual check and adjustment.
  - TPMS can be gamed through rare keywords and it seems trivial to build a tool that tries to significantly increase the

chances of getting certain reviewers from the committee even without their knowledge. While this shouldn't be problematic in practice as it is very similar to explicitly listening "preferred" reviewers, it is something to be aware of.

- The formula we used to combine the "systems" flag and TPMS scores sometimes results in a sub-optimal assignment. For example, consider a systems paper on semantic data integration and two reviewers, one working on the theoretical aspects of semantic data integration and another building systems focused on data cleaning (a topic relevant to, but different than data integration). The first reviewer scores high on topic relevance and the second scores high on the system trait. It is not clear which of these two reviewers is preferable and it becomes even worse considering that the "relevance" and the research area aspects are neither discrete nor well-defined values. This is unfortunately a fundamental issue – it is just not clear how these two indicators should be properly combined to generate a quantifiable "appropriateness" measure. We resolved the issues by checking the assignments ourselves visually as well as asking the reviewers for confirmation on the appropriateness of the assignments and correcting as needed.

- Additional reviews were solicited manually by the chairs and this was a huge time sink, especially when some reviewers refused to take on the additional assignment. The additional review solicitation needs to be automated and reviewer expectations need to be set appropriately beforehand.

- The chairs discovered low-confidence reviews manually; such reviews, however, should be flagged automatically to allow for immediate action.

- A continuous automated analysis of the reviews as they come in to spot problematic text, low-confidence reviews, and poorly attended discussions, could all dramatically alleviate the overhead that chairs and meta-reviewers endure while trying to detect the problem cases manually.

- The meta-reviewers can help suggest appropriate reviewers for papers, but this is only efficient if it is automated.

5. CONCLUSIONS

Overall, the experiment of SIGMOD 2019 with all its changes (TPMS, no reviewer bidding, reviewer suggestions by authors, automatic additional reviewers for papers with one "weak accept", etc.) was a success. Like with every large conference we did receive some complaints, but those were significantly fewer than other years. Furthermore, systems and non-systems papers were treated exactly the same and neither were at a disadvantage. This indicates that SIGMOD should remain a single conference and not be split up as recently suggested [6].

However, there is of course also room for improvements for future conferences and more potential things to try out to improve the overall quality of the reviewing process and with it the quality of the papers.

Automatically adding reviewers for every paper with one "weak accept" is extremely rewarding. In many cases it significantly helped to increase the reviewing quality. Furthermore, reviewing effort is not wasted on papers which are clear rejects/accepts. However, assigning additional reviews has side effects: for instance, the additional reviews may cause the paper to be rejected while without the additional reviewers the paper might have been accepted. This happened with a couple of SIGMOD 2019 submissions and while authors expressed concerns, careful monitoring of the process revealed that the final decision was the correct one: a solid paper can withstand thorough reviewing. As discussed earlier, this process needs to be more streamlined and better integrated into the reviewing workflows.

Preferred Reviewers: Authors have the best understanding of their work and know their peers. As a result they are the experts to resolve misconceptions and propose reviewers for their work. Therefore, increasing author feedback improves the quality of the conference while asking them to suggest reviewers for their work typically provides excellent reviewing matches. Of course, as reviewer identities should be hidden and authors may misuse the feature, great care has to be taken on how the author suggestions are used. Our approach gives low weight to the authors' suggestions during the automatic assignments and increases the weight when chairs are looking for additional reviewers; this reduces the probability of leaking reviewer identities and using malevolent suggestions.

Open reviews: Some conferences have already moved to the open-review concept. While obvious reservations exist (e.g., that a reviewer of a re-submission looks at the reviews of the original rejected submission), open reviewing constitutes a

strong incentive to create stronger, more thorough reviews. Thus, testing open reviews for SIGMOD would be an interesting experiment.

Feedback: A rewarding decision is to allow authors to complain about specific reviews. The chairs considered every complaint carefully, investigated them personally, and followed up by discussing with the reviewers and by inviting additional reviews. Several authors expressed their appreciation about the additional steps the SIGMOD organization took to alleviate all misunderstandings.

Recruiting and tracking reviewers: Recruiting reviewers is a manual and slow process, the lack of constructive feedback towards the reviewers and of proper guidelines in writing reviews is still an issue, and external reviewer load and unresponsiveness can have a ripple effect on increasing the load of other PC members. This entire process has to be reconsidered and automated as much as possible.

Reviewer delays Reviewers are typically expected to complete their assignment over several weeks and stage the work. In practice, however, few reviewers are organized enough to distribute their work evenly throughout the review period; most review their entire set of assigned papers in the last few days before the deadline. As is expected, when the deadline arrives several papers are missing reviews and time is wasted trying to hunt down delinquent reviewers and obtain guarantees for delivery of reviews. The reviewing system can provide an optional parameter setting to enable the reviewers to plan their work. The parameter can enable fine-grained deadlines for reviews. For example, if a reviewer is responsible for 20 papers, they can set the deadline for the first five papers in two weeks, the next five after four weeks and so on, and receive automatic reminders. The PC chair may enforce such fine-grained deadlines and receive input on progress. Based on the input, the chair can re-balance load among reviewers as needed.

Plug-in Structure: If existing conference management tools were providing APIs for integrating external plugins/tools and potentially a sandbox environment, many tasks we had to perform would be much easier and faster to do.

Evaluation: The conference system can provide custom success metrics defined based on the procedures in place, and collect data accordingly. For example, it would have been great to know how many reviewers changed their rating after the discussion or after reading author feedback, and how they changed it (favorably or not).

To implement the suggestions above it is important that the PC Chairs are engaged two years before the conference (one year before the first deadline) so that they have time to work with the conference system to put all processes in place.

6. ACKNOWLEDGMENTS

We wholeheartedly thank the authors of SIGMOD 2019 submissions as well as the reviewers for their impeccable collaboration and timely response during the process. We also thank the SIGMOD 2019 executive committee for their support and feedback.

7. AUTHORS AND ROLES

The four authors worked together from December 2017 until June 2019. Anastasia Ailamaki is the SIGMOD 2019 Program Chair, and Amol Deshpande and Tim Kraska are the vice Program Chairs for the SIGMOD 2019 Research Track. The three collaborated with the reviewers in shaping the program of SIGMOD 2019. Periklis Chrysogelos supported the entire reviewing assignment system by designing the formulas, implementing the scripts and tuning and interacting with CMT as needed.

8. REFERENCES

[1] Aminer. `https://aminer.org/`.
[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
[3] L. Charlin and R. Zemel. The toronto paper matching system: an automated paper-reviewer assignment system. *Proceedings of the 30th International Conference on Machine Learning*, 2013.
[4] Conference management toolkit. `https://cmt3.research.microsoft.com`.
[5] M. Grant and S. Boyd. Cvx: Matlab software for disciplined convex programming, 2014.
[6] M. Stonebraker. My top ten fears about the DBMS field. In *34th IEEE International Conference on Data Engineering*, 2018.
[7] C. J. Taylor. On the optimal assignment of conference papers to reviewers. *Technical Report*, 2008.