Filtering News from Document Streams: Evaluation Aspects and Modeled Stream Utility

by

Gaurav Makhon Baruah

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirements for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Events like hurricanes, earthquakes, or accidents can impact a large number of people. Not only are people in the immediate vicinity of the event affected, but concerns about their well-being are shared by the local government and well-wishers across the world. The latest information about news events could be of use to government and aid agencies in order to make informed decisions on providing necessary support, security and relief. The general public avails of news updates via dedicated news feeds or broadcasts, and lately, via social media services like Facebook or Twitter. Retrieving the latest information about newsworthy events from the world-wide web is thus of importance to a large section of society.

As new content on a multitude of topics is continuously being published on the web, specific event related information needs to be filtered from the resulting *stream* of documents. We present in this thesis, a user-centric evaluation measure for evaluating systems that filter news related information from document streams. Our proposed evaluation measure, Modeled Stream Utility (MSU), models users accessing information from a stream of sentences produced by a news update filtering system. The user model allows for simulating a large number of users with different characteristic stream browsing behavior. Through simulation, MSU estimates the utility of a system for an average user browsing a stream of sentences. Our results show that system performance is sensitive to a user population's stream browsing behavior and that existing evaluation metrics correspond to very specific types of user behavior.

To evaluate systems that filter sentences from a document stream, we need a set of judged sentences. This judged set is a subset of all the sentences returned by all systems, and is typically constructed by pooling together the highest quality sentences, as determined by respective system assigned scores for each sentence. Sentences in the pool are manually assessed and the resulting set of judged sentences is then used to compute system performance metrics. In this thesis, we investigate the effect of including duplicates of judged sentences, into the judged set, on system performance evaluation. We also develop an alternative pooling methodology, that given the MSU user model, selects sentences for pooling based on the probability of a sentences being read by modeled users.

Our research lays the foundation for interesting future work for utilizing user-models in different aspects of evaluation of stream filtering systems. The MSU measure enables incorporation of different user models. Furthermore, the applicability of MSU could be extended through calibration based on user behavior.

Acknowledgements

This PhD was an interesting and enlightening journey, a journey that challenged me on all fronts intellectual, professional, physical and spiritual. I was privileged to have Prof. Mark D. Smucker as my supervisor and mentor for my PhD; his advice was invaluable all throughout. I must accord my heartfelt thanks to Prof. Mark D. Smucker for his encouragement, patience and understanding for all matters that affected my PhD. I was also privileged to have worked with leading Information Retrieval (IR) scientists: Prof. Charles L. A. Clarke, my co-supervisor who is always quick to help and develop ideas; Prof. Jimmy Lin, who is always encouraging of ideas and is ever-ready to contribute to and take research projects forward; and Prof. Olga Vechtomova, who collaborated with me on two related research projects.

I was fortunate to have a PhD Examination committee that consisted of many IR luminaries: Prof. Jay Aslam, whose Temporal Summarization research initiative formed the bedrock of my thesis; Prof. Lukasz Golab, who helped me cement my understanding of fundamentals; Prof. Gordon Cormack, who helped me to communicate research ideas more effectively; Prof. Jimmy Lin, who cleared pathways to future work; Prof. Charles Clarke, who advised on the core chapter of my thesis; and Prof. Mark Smucker, whose advice helped make the thesis possible. My defense was a tremendous learning experience for me; for this I sincerely thank my examination committee. I do humbly and happily realize that the world of IR research is rich with opportunity; I sincerely hope that I am able to apply the skills I learned for furthering the understanding of Information Retrieval.

An undertaking such as a PhD cannot really come to fruition without the help and support of colleagues, friends and family. I would like to thank my colleagues and friends: Aiman Al-Harbi, who had the same advisors as I did, and with whom I shared my office, and we both kept each other going through tough times personal or professional during the PhD; Adam Roegiest, with whom I co-authored two papers and whose advice was helpful in navigating many issues be it research or otherwise; Maheedhar Kolla, whose advice on what to do and what not to do was eye opening during the whole PhD journey and beyond; Rakesh Guttikonda, for his collaboration on a research project that led into my PhD thesis; Bahareh Sarrafzadeh, Adriel Dean-Hall, Ashif Harji, Haotian Zhang, Le Li, Alexandra Vtyurina and Luchen Tan, all provided fascinating world views and wonderful discussions on research and other topics all throughout my PhD.

I would like to especially thank Akshay K. Singh for being a great and wonderful friend and Linda Schryer for sharing her pragmatic advice and life experience. Both were excellent interlocutors for matters professional and personal, and they kept me grounded and sane

when I needed it the most. For this I am eternally grateful to them. I thank my Indian friend circle in Waterloo in helping me to partake in joy even when the going got tough at times.

I would like to thank my mother, Nirupama Baruah for her unflinching and unwavering support throughout my PhD, even as she experienced various travails in life, and my father, Makhon Prasad Baruah who provided inspiring examples and anecdotes in order to keep me motivated during my darkest hours. My parents have far exceeded their responsibilities with regards to my education by supporting me financially or emotionally as need be and I hope I have lived up to their expectations through this PhD. My wife, Hemangi Rai, was always there; I thank her for accompanying me on my academic journey and for contributing in whichever way she could. I thank both my in-laws Anirudh Muktinath Rai and Minakshi Rai who were supportive of my decisions to educate myself; my father-in-law greatly helped me during my admission process and helped to ensure that things were in order back home. I must also thank my brother Anirudh, my extended family, and all the friends I have had the pleasure to meet during my lifetime for all their prayers and well wishes.

I must also thank those no longer with us: my grandmother, Sundarabai Sahadu Jadhav, a respected teacher and an inspiration to many, who not only cared for me but her advice and blessings inspired me to pursue the quest for knowledge and aspire to the noble profession of teaching; Mr. Menino Silveira taught me how to play the guitar well and he also taught me a great many lessons for life, chief among which was instilling discipline through the beats of a metronome.

Life is an ongoing learning experience; a PhD however, is the last recognized academic milestone and I would not have reached here without the encouragements of teachers/mentors at the various schools/organizations that I have attended. I would like to acknowledge and thank teachers at respective alma-maters: the many respected teachers, house masters and matrons at the Sanjeewan Vidyalaya, Panchgani; the lecturers at the Bharati Vidyapeeth's College of Engineering, Pune; the esteemed professors at the Indian Institute of Technology (IIT), Guwahati; and the exceptional faculty at the Cheriton School of Computer Science, University of Waterloo.

I would like to accord heartfelt thanks to Ganesh Bhutkar, Assistant Professor, Vishwakarma Institute of Technology, Pune—then at the Bharati Vidyapeeth's College of Engineering, Pune—who allowed us to pursue a research oriented final year undergraduate project even when the overall trend and overwhelming suggestions were to do a software engineering project. Prof. G. Sajith and Prof. Diganta Goswami, my advisors at the Indian Institute of Technology, Guwahati, were instrumental in shaping my outlook towards

research as a career choice; I am highly grateful for their advice and thankful to them as well as to Prof. Sukumar Nandi who motivated me to pursue a PhD.

I would also like to thank Prof. B. Ravi, Institute Chair Professor, Mechanical Engineering Department, IIT, Bombay, who gave me my first break and working at the IIT first exposed me to research in an academic setting. Dr. Shailesh Deshpande, Founder and CEO at Intellection Software and Technologies, was an exceptional mentor; at Intellection, I learned how to run a growing startup, translate research into workable solutions and more importantly how be humble and resolute in the face of adversity in the professional as well as personal spheres of life. Dr. Deshpande, in particular, was highly supportive of my pursing higher education. Phaneendra Kumar D., Technical Lead at Cisco—then at Geodesic Limited, Bangalore—was also a great mentor who not only guided me in managing a team of engineers but was also highly encouraging of my PhD aspirations.

A graduate student not only works on research but also partakes in the day to day activities of the university. I would like to thank Olga Zorin from the Instructional Support Group, for her advice, understanding and professionalism that she accorded to me during my years as a Teaching Assistant and as an Instructional Apprentice for the CS 251 Computer Organization and Design course at the Cheriton School of Computer Science. I would like to thank: Margaret Towell and her team at the CS Graduate Office for all their support and quick resolution of administrative tasks; Gordon Boerke who facilitated quick resolutions of IT related problems; Helen Jardine, and other administrative staff whom I had the pleasure of meeting at the Cheriton School of Computer Science office; and the extremely competent International Student Experience team at the University of Waterloo who greatly helped simplify resolution of immigration related issues in a way that eased the PhD process throughout.

Finally, I would like to extend my gratitude and acknowledge the various organizations and agencies that provided facilities, infrastructure, support and funding that enabled me to work on the research presented in my thesis. The research work for my thesis was made possible by the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET) and Compute/Calcul Canada, and the facilities of the David R. Cheriton School of Computer Science at the University of Waterloo, and was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Google Founders Grant, and in part by the Graphics, Animation and New Media Network of Centres of Excellence (GRAND NCE), and in part by an Amazon AWS in Education Research Grant, and in part by the University of Waterloo.

I would at the very end like to thank all those whom I may have missed in this acknowledgement.

Dedication

Dedicated to all my Teachers (Gurus) — who create knowledge, sustain knowledge and dispel ignorance.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information about recent noteworthy events is reported in the news. Significant news events could be sensational, historic, groundbreaking or unpredictable. In many cases, news events transpire over a period of time, for instance, a sporting event like the Olympics, or a natural calamity like a hurricane, or the aftermath of an earthquake. Given a time ordered document stream, such as the World Wide Web (Web)—wherein content is continuously published via social media, blogs, or news reports—news updates for an event of interest can be filtered/retrieved from the document stream as the event evolves over time.

In this thesis we focus on the evaluation of IR systems that retrieve or filter news updates over a defined period of interest for a news event. Our main contribution is the development of a user-model based evaluation measure for evaluating such systems. We also investigate related aspects of system evaluation which could affect the performance measurement of news filtering systems.

1.1 Motivation

The sudden onset of unpredictable events can generate related information needs amongst affected people, first-responders, government agencies, aid providers, observer groups, diaspora around the world and other interested people. Such events could be unexpected like earthquakes or accidents, holding public interest for varying periods of time. On the other hand sporting events or elections often hold public interest for the duration of event, and possibly, during the days leading up to the event. In both cases, as time progresses, more information about the event may come to light, resulting in the events taking on a dynamic (or evolving) form. For instance a shooting incident is a sudden occurrence however it sets into motion a chain of events such as law enforcement, government response, citizen reactions, safety updates, post event investigations, and analysis. These events that follow in the aftermath of the main event, are themselves newsworthy and can contain information of interest to various users. Some events, like hurricanes or epidemics, are inherently evolving events.

The news about events may have an impact on possibly large sections of society. Following an event, the Web as a publishing medium, can expect the addition of documents or articles about the event, all generated in the hours/days following an event. Documents relating to the event may come from social media (microblogs and social networks), news services, personal blogs and other media. Together, these sources of information effectively form an *event-stream* that contains content about an event. From an information retrieval perspective, the event-stream is obtained by filtering an *aggregate-stream* of all documents that are being generated and published over the Web in the same time period that contains

the event.

A concerned user on searching the web via a search engine, might expect to find documents from the event-stream to gain more information about the event. When consuming documents about such events, users might need to know as many facts as possible at the earliest. A concerned user may check back multiple times, in order to find out new information or important facts about the event. A user might realistically expect that the latest information is novel and is returned with low latency. By low latency, we mean that the time lag between actual occurrence of the event and the time at which the information is presented to the user is as small as possible. Furthermore, shorter and to the point information could be most helpful for the user.

In reality, users could be affected in different ways by an evolving news story. Users in the government or health care, or users in the immediate vicinity of the event may need extremely low latency information updates. Periodic or urgent updates may be more desirable for long ranging events like hurricanes or epidemics. Moreover, interested users may only be able to check back for updates as their time constraints allow.

1.1.1 Temporal Summarization at TREC

The Temporal Summarization Track (TST) (Aslam et al., 2013; Aslam et al., 2014) at the Text Retrieval Conference (TREC), in particular, fosters the development and the evaluation of systems that retrieve updates regarding news events over a specified duration of time. The topics for the track in 2013 were news events of the type accident, bombing, earthquake, shooting and storm. Later iterations of the track also included events of the

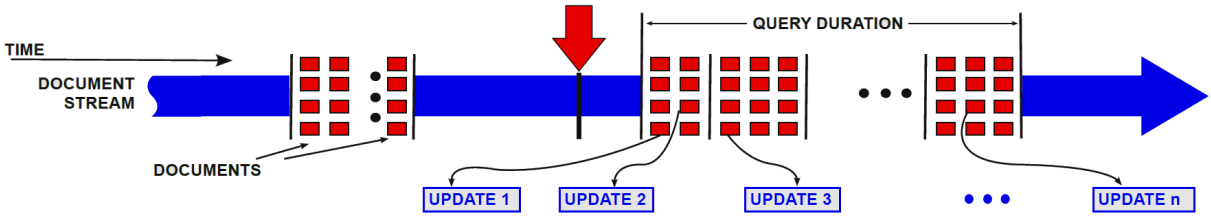


Figure 1.1: The Temporal Summarization Task at TREC. The red arrow indicates the time at which the event occurred. A system returns a temporal summary by emitting updates at various instants during the query duration.

type protest, impact event, hostage, riot and conflict.

The TST topics mainly comprise of a query string for an event and a specified query duration (period of interest) following the event. Participating teams at the track were required to submit *runs* (system results), consisting of topically related sentences filtered from a time-ordered document stream, with each sentence having a timestamp indicating the instant of retrieval. The set of sentences returned at various instants across the period of interest forms a temporal summary of updates for the event (Figure 1.1). The returned set of sentences were evaluated for their *relevance*—did the sentence contain a topically relevant fact, *latency*—was the sentence returned with minimal delay following the actual occurrence of the fact, and *verbosity*—does the sentence contain minimal extraneous content. Relevant sentences having low latency and low verbosity were preferred over other submitted sentences.

The track utilized a nugget based evaluation framework. Important facts (i.e. *nuggets*) about each news event were extracted from Wikipedia[1]. Each nugget was associated with the timestamp at which the nugget first appeared in the event's Wikipedia article's edit

[1] https://en.wikipedia.org/

history. Thus, an update (the returned sentence) is relevant if it contains a nugget. The latency of the update is the time difference between the contained nugget's timestamp and the update's timestamp. The verbosity of the update is the amount of non-nugget content in the update (Section 3.1.2). Gain experienced on reading a relevant update is discounted for latency. The track defines a precision analogous measure Expected Latency Gain (ELG), that measures the latency discounted gain per verbosity normalized update. The track also defines a recall analogous measure Latency Comprehensiveness (LC), that measures the recall of the nuggets for the topic, given the latency discounted gain.

We participated in the Temporal Summarization Track at TREC 2013. Chapter 3 discusses the track in detail and presents an example of performing the TST task. Our participation experience led to interesting directions of research. Specifically, we explored various factors for the evaluation of temporal summarization systems, leading up to the development of, a user-model oriented evaluation measure, for systems that produce streams of updates for news events.

1.2 Overview and Contributions

In this thesis, Chapter 2 highlights related prior research on filtering systems and their evaluation, different kinds of information streams for which information retrieval techniques have been applied, such as news articles, documents and microblogs, as well as, various applications of filtering such as topic detection and tracking (Allan et al., 1998), tracking epidemics (Culotta, 2010), along with the retrieval/filtering methods and evaluations used thereof. In Chapter 3, we describe the temporal summarization track, the track's evaluation

measures, the participating teams' approaches to temporal summarization as well as our participation attempt at TST 2013.

1.2.1 Evaluation in the Presence of Duplicates

Following our participation at TST 2013, we observed a large number of duplicate sentences in the track's prescribed document collection, the KBA Stream corpus 2013 (Frank et al., 2014). We hypothesized that duplicate sentences with earlier timestamps could help improve the measured performance of the participating systems as the latency of returned updates would reduce. We first explored the proliferation of duplicates of judged sentences from the TST 2013 evaluation pool in the KBA stream corpus 2013. We then investigated the effect of expanding the judged set of sentences with the found duplicates on the evaluation of submitted participant systems at the track (Baruah, Roegiest, and Smucker, 2014). Although the ranking of systems does not significantly change, adding duplicates allows for more accurate computation of performance scores for systems (Chapter 4).

1.2.2 Modeled Stream Utility

The primary contribution of this thesis is the development of Modeled Stream Utility (MSU) (Chapter 5), a user-model based evaluation measure for systems that produce event related updates by filtering an aggregate stream (Baruah, Smucker, and Clarke, 2015). Essentially, MSU utilizes a simple user model that simulates users accessing information from a stream and measures *gain* based on updates read by modeled users, for evaluating a system. The gain is a function of the number of relevant information nuggets read by a

user from a given system's response; the average gain across all modeled users for a system is the system's modeled stream utility. MSU primarily builds on the time well spent work by Clarke and Smucker (2014), and likewise, incorporates models of user-behavior over a user-interface for effective user performance measurement (Clarke et al., 2013).

For TST 2013, there is a large variation in the volume of content returned by the participating systems (Table 3.7). The number of updates returned across participating systems, numbers from less than 200 to more than 2.8 million. The evaluation measures of the Temporal Summarization track are analogous to precision (ELG) and recall (LC). Systems that returned a higher number of sentences, scored higher on LC. Returning fewer sentences resulted in higher ELG.

The best run as per ELG returned only 197 sentences (21.9 sentences per topic). However, the specified query duration was 10 days, i.e. the run returned approximately 2 updates per day to the user, with the total relevant content being 6.89 nuggets. However the identified number of nuggets per event averages 119 nuggets per topic (Table 5.3). We wondered if a user would be satisfied to receive less than 7 nuggets for a news topic when the actual amount of relevant content could be much higher.

At the other end of the spectrum, the most updates returned by a run numbered 312,863 per topic over the 10 day query duration. Though, the expected recall is higher in this case with 37.8 nuggets found, it is unlikely that the user would read 31,286.3 updates per day. These examples lead us to the interesting questions of how a user might consume the stream of updates? How many updates would a user prefer to read?

Our experiments show that the development and evaluation of systems that generate

streams of updates, would benefit by a cognizance of target user behavior. Accordingly, for real world applications, user models would need calibration by observing real users.

1.2.3 Simulation-based Pooling

For the development of the MSU measure we utilized the pooled and judged set of sentences from TST 2013. The evaluation pool for TST was constructed using a variant of depth-pooling (Aslam et al., 2013). Standard depth-pooling selects a specified number of top-scoring sentences from each participating run to construct the evaluation pool. On simulating 10,000 users generated via the MSU user model (Section 5.1), we found that very few relevant updates, as judged for TST 2013, were read by simulated users. In other words, a large number of simulated users encountered very few relevant sentences in the set of updates that they read. Almost a third of the relevant updates were read by less than 1% of the simulated users (Figure 6.4). The set of judged relevant sentences was found to have little overlap with the set of sentences read by the simulated users.

Under evaluation by MSU, it may be beneficial for systems to have the updates that are actually read by users, to be labeled by assessors. A user model for streaming information access allows for recording which updates are actually read by users. With the knowledge of which update is read by which modeled users, we are in a position to estimate the probability of an update being read (Baruah, Roegiest, and Smucker, 2015). Given the probabilities of updates being read, we propose two pooling methods: depth-pooling with the probability as the selection criteria, and pooling based on probability mass cover (Chapter 6).

To know how effective (or adequate) the probability based pooling methodologies are, we need to acquire relevance judgements for the probability based pools. Analyses on the quality of resulting pools, such as those done by Cormack and Lynam (2007) and Buckley et al. (2007), is an interesting area for future work.

1.3 List of Contributions

1. We found that there are a large number of duplicate sentences in the KBA Stream Corpus 2013; duplicates of judged sentences number 9,034,179, almost 1,000 times the number of judged sentences, 9,113 (Chapter 4).

2. On including the duplicates into the judged set of sentences we found that, the absolute scores for half (13 of 26) the participating systems show significant changes over the track's evaluation measures (Chapter 4).

3. The inclusion of duplicate sentences, however, does not affect the relative performance of the systems significantly. The Kendall's τ rank correlation between system rankings induced by the track's relevance judgements, and the system rankings induced by the expanded set of judgements, remains high at 0.899 for ELG, and 0.942 for LC (Chapter 4).

4. We developed a user model for *streaming information access*. We model the behavior of a user reading updates from a stream. The model simulates a user alternating between, (a) spending time reading updates from the filtered stream, and (b) spending

time away from the system. Depending on the user's interest and time constraints, the durations of (a) and (b) would vary (Chapter 5).

5. Given the user model for streaming information access, we develop an evaluation measure (MSU) that measures how many nuggets are read on average by a user. The method involves simulating a number of users given the user model with specified parameters, and noting the relevant updates, and thereby the nuggets, read by each user. Then, MSU is the number of nuggets read on average by a user from the simulated user population (Chapter 5).

6. We demonstrate MSU using the test collection and participating systems of the Temporal Summarization track at TREC 2013. Systems returned likely relevant sentences as updates for a topic at various points in time. These updates produced at various instants across the period of interest can be considered to be a stream of likely relevant updates for a topic. Our experiments show that for a reasonably interested user (who checks back for about 2 minutes every 3 hours on average), the relative performance of systems does not correlate well with the track's measures, ELG and LC (Chapter 5).

7. On exploration of the parameter space for MSU's user model, we find that parameter values that induce a system ranking that is closest to the ranking produced by ELG, represent a user population that reads updates for about 1 minute per day on average. Such users would seem to be highly constrained for time, or, they lack interest in the event. In other words, ELG caters to highly time constrained or selective users (Chapter 5).

8. On simulating many different user populations, by varying the MSU user model parameters, we find that gain derived when using a system is sensitive to different kinds of user behavior. Model parameters characterize user behavior, which determines amount of content consumed, which in turn determines system performance (Chapter 5).

9. We propose novel pooling methodologies based on the probability of a sentence being read, given a user model for streaming information access. We look at alternative formulations to compute the probability of an update being read (Chapter 6).

10. We compare standard depth-pooling based on system assigned scores with probability based depth-pooling. The respective pools have less than 45% overlap and are thus quite different (Chapter 6).

11. Given probabilities, we can also pool updates based on *probability mass cover*. For instance, for a specified value of probability mass, one could select updates into the pool from a given system, till the contributed probability mass—as per the probabilities of the updates selected into the pool—equals the specified probability mass. The resulting pool would then contain the updates that are most likely to be read for each of the contributing runs, up to the specified probability mass cover. Depending on the desired probability mass cover and the size of the contributing runs, the size of the pool and hence the assessing effort could differ. This is different from standard depth-pooling in which each system contributes the top-k sentences to the pool.

Chapter 2

Related Research

This thesis builds upon 3 major themes of ongoing and previous research in information retrieval and filtering:

- Information retrieval and evaluation in a temporal setting, and related applications (Section 2.1).
- User-behavior driven evaluation (Section 2.2).
- Construction of test collections, and evaluation measures (Section 2.3).

Research in this thesis lies at the intersections of these broad areas of information retrieval. We discuss in this chapter the salient research findings on which we build this thesis. We also connect our research with related work and compare our methods with similar approaches.

2.1 Temporal IR: Systems and Evaluations

Research in Temporal Information Retrieval (TIR) has seen growing impetus in recent times (Kanhabua, Blanco, and Nørvåg, 2015; Berberich et al., 2015; Spaniol, Masanès, and Baeza-Yates, 2014; Aslam et al., 2014). Although, Information Retrieval (IR) research in a temporal context has been ongoing for over 2 decades (Voorhees, Harman, et al., 2005; Allan, 2002; Soboroff, 2004), the problem has seen growing interest due to the much larger scale of data as well as change in nature of the IR tasks and applications (Frank et al., 2013; Aslam et al., 2013; Lin et al., 2014). Challenges and applications for temporal information retrieval include, processing a dynamic—ever growing—corpus (Kanhabua, Blanco, and Nørvåg, 2015), time aware rankings (Li and Croft, 2003; Dong et al., 2010; Metzler et al., 2009), understanding temporality of query intents (Kulkarni et al., 2011; Jones and Diaz, 2004), detecting events and trends (Allan et al., 1998; Zhu and Shasha, 2003), summarizing the evolution of a topic over time (Swan and Allan, 2000; McCreadie, Macdonald, and Ounis, 2014), monitoring epidemics (Ginsberg et al., 2009; Paul and Dredze, 2011; Culotta, 2010), and disaster awareness (Imran et al., 2013; Earle et al., 2010; Rogstadius et al., 2013; Vieweg et al., 2010).

Belkin and Croft (1992) first compare information filtering and information retrieval problems and highlight that there is little difference between them. They point out that timeliness of retrieved information is an important consideration, however temporal aspects can be context dependent. They acknowledge that filtering could be concerned with long-standing information needs. IR systems with temporal aspects have been researched at TREC through various tracks over several years (Voorhees, Harman, et al., 2005). However,

there has been an exponential growth in the scale of document collections over time. In recent years, the temporal summarization track at TREC looks at filtering information from a stream of documents. The microblog track has in recent years also worked on filtering (Lin et al., 2015) as well as the generation of timeline overview summaries (Lin et al., 2014).

Apart from TREC, temporal IR has also seen support from evaluation workshops like NTCIR and CLEF, as well as dedicated workshops running alongside major conferences. The Temporalia track at NTCIR (Joho, Jatowt, and Blanco, 2014; Joho et al., 2014) focusses on temporal query intent and temporal information retrieval. The CLEF NewsREEL task (Hopfgartner et al., 2014) focuses on news recommendations evaluated in a living-labs (Balog et al., 2014) setting. The Time-aware Information Access (TAIA) workshop (Diaz et al., 2012; Diaz et al., 2013; Diaz et al., 2014) has been running for 4 years alongside the SIGIR conference, with a focus on time aware rankings, temporal summarization, real-time search, event detection, as well as evaluation methodologies for time-aware systems. The Temporal Web Analytics Workshop (TempWeb) has been running for 5 years (Baeza-Yates, Masanès, and Spaniol, 2012; Baeza-Yates, Masanès, and Spaniol, 2013; Spaniol, Masanès, and Baeza-Yates, 2014) alongside the WWW conference, with a similar research focus as the TAIA workshop, as well as temporal data mining and predictive applications.

Diaz (2014), builds a case for a test collection for crisis informatics, an apt application for temporal IR. The temporal summarization track (TST) at TREC can be considered to be a step in this direction. The TST also works with a web-scale time ordered corpus containing over a billion documents spanning 2 years (Frank et al., 2014).

In the following sections, we discuss the evaluation methodologies developed for temporally inclined IR tasks. Specifically, we follow the evolution of IR tasks that relate to filtering/retrieving documents/information from a temporally ordered stream of documents.

2.1.1 Filtering

The TREC Filtering track ran for 7 years (Voorhees, Harman, et al., 2005). Various iterations of the track focused on *batch* filtering and *adaptive* filtering, along with some variations. The primary filtering task was to process a stream of documents and find documents relevant to a user *profile*. The user profile essentially represented a topic, however, the profile also contained additional information such as a query for the information need, appropriate thresholds for the filter, and kept track of already acquired information like feedback from the user. Relevance judgments for filtered documents were instantaneously made available to the system.

The filtering track differs from topic detection and tracking (Section 2.1.2) in that there is no detection component to the problem. On the other hand, the filtering track resembles TST (Chapter 3) in its use-case simulation, with some notable differences. TST works on a web-scale dataset and TST does not incorporate concurrent judgments as provided by the filtering track's simulation. Instead, for TST, systems emit (filter) updates and assign a confidence score to each emitted update. Updates from submitted systems are pooled and then judged for relevancy after the fact. Essentially, there is no user-feedback aspect to the profile for TST. Another difference is that the Filtering track requires that a system

should take the decision to filter the document as soon as it is processed. TST does not have this constraint, i.e. the system can decide to emit any number of updates at any time deemed appropriate by the system.

The evaluation of the filtering track was based on a notion of utility,

$$Utility = AR^{+} + BN^{+} \tag{2.1}$$

where A is a positive constant, B is a negative constant, R^{+} is the set of retrieved-relevant documents and N^{+} is the set of retrieved-non-relevant documents. TREC evaluation for filtering later evolved into using a precision oriented F-measure. Both the KBA track (Section 2.1.5) and the TST 2014 track (Section 3.2) evaluation measures currently have a formulation of an F-measure that combines precision-like and recall-like measures for evaluation of stream filtering systems.

Appropriate thresholding is a major issue for filtering systems (Voorhees, Harman, et al., 2005; Robertson, 2002) and inappropriately chosen thresholds for a filtering decision can affect system performance. This problem also manifests itself for recent temporal summarization tasks (McCreadie, Macdonald, and Ounis, 2014).

2.1.2 Topic Detection and Tracking

The Topic Detection and Tracking (TDT) research encompassed topic detection, topic tracking, link detection, first story detection and story segmentation from news related media (Fiscus and Doddington, 2002). TDT research also led to the construction of 4

TDT corpora. The corpora consisted of news articles spanning multiple languages. TDT systems detected and returned news stories that are relevant to the topic. TDT evaluation centered around missed detection rate and false alarm rate. A missed detection occurs when a news story is not detected as being relevant. A false alarm occurs when a relevant news story is categorized as being non-relevant. To measure the performance of systems, a detection cost function and a decision error trade-off curves were employed. Both measures utilize the probabilities of missed detections and false alarms.

Allan, Gupta, and Khandelwal (2001) first formulate the problem of updating the user with sentences containing information about an evolving news story. They were inspired by prior topic detection and tracking research (Allan et al., 1998) which relates to tracking documents about a running news story. They define evaluation measures that measure *usefulness* (of event sentences), *novelty* (non redundant sentence content) and *size* of the summary produced by a system. A summary is typically a set of sentences. The TST uses similar evaluation criteria for evaluating the results returned by participating systems (Section 3.1.2).

2.1.3 Streams

Kleinberg (2006) summarizes the state of the art (circa 2006) for methods and technologies that work on the temporal dynamics of information. He puts forth the view that there exists data that can be perceived as a stream of information rather than a static collection. Examples of data conforming to such a viewpoint are emails, scientific publications, patents, news, etc. Such information streams can be characterized as being (i)

bursty (episodic), such that topics can grow (or reduce) in intensity over a specified interval of time, and (ii) the topics (or sub-topics) could be the subject of temporally co-located documents, however, the user only sees the *merged (braided)* stream. A system that filters information from a stream must therefore address these complexities.

Kleinberg (2006) highlights topic detection and tracking (Section 2.1.2) as being one of the first attempts that aims to address problems posed while working with text information streams. Typically the TDT methods tend to be based on (i) thresholding: characterizing features occurring with more frequency than average (Swan and Allan, 2000), (ii) state-based: finite-automata methods with states representing low/high intensities of topics (Kleinberg, 2002), or (iii) trend-based: frequency of co-occurring terms/features. Apart from TDT, these methods may find use in the processing of blogs, search engine query logs and may also help gauge user-behavior via implicit feedback techniques.

Finally, Kleinberg (2006) outlines various issues in dealing with temporal information streams that are arguably still valid today. The foremost of these are that the prediction of bursts is hard especially in real-time deployments of a stream filtering system. For example, for a news-stream, interest may quickly peak following a natural disaster, whereas, for sporting-events or elections, there is usually a slow build-up followed by a quick decay in traffic. Another problem is the temporal alignment of multiple information streams.

Most work on streams typically addresses the problem of burst detection/prediction (Zhu and Shasha, 2003; Kifer, Ben-David, and Gehrke, 2004), or classification/filtering of streams (Katakis, Tsoumakas, and Vlahavas, 2006; Hong et al., 2011). In recent years the dataset of choice seems to be Twitter (Lee and Chien, 2013; Pozdnoukhov and Kaiser,

2011) as tweets essentially form an information stream containing user generated content that can be mined for various applications. As such, the evaluation for such methods tends to be typical of machine learning or data mining methods, e.g. Accuracy, Area Under Curve (for Precision and Recall) and/or significant improvements over an existing baseline.

Techniques to enhance *Situational Awareness*, i.e., understanding the overall view during critical situations like natural disasters, is a closely related research area. The use of Twitter and social media (Vieweg et al., 2010; Rogstadius et al., 2013; Imran et al., 2013; Earle et al., 2010) for improving situational awareness is an ongoing area of research. Twitter has also been used for predicting the spread of epidemics and general population health monitoring (Culotta, 2010; Paul and Dredze, 2011), which could be considered as slowly evolving news events.

2.1.4 Temporal Summarization

The core temporal summarization task as detailed in Chapter 3 is first explored by Guo, Diaz, and Yom-Tov (2013). They focus on updating users for time-critical events. The scale of the data and urgency of the information requires the system to make an on-line decision about filtering a likely relevant update. That is to say, the summarization is not "after the fact" and takes place with minimal delay as more information becomes available.

Guo, Diaz, and Yom-Tov (2013) primarily focus on performing temporal summarization for rapidly evolving news events for which the information need is urgent (or time-critical). They consider an event to be composed of a number of subtopics. Their system filters updates from a time ordered stream of documents. Each update may contain one or more

subtopics. Accordingly, precision and recall for each update may be computed given the subtopics of the update and the subtopics of the event. They define measures *expected precision* and *expected recall* of an event's subtopics over the complete set of updates.

Evaluation measures for the temporal summarization track were based on the relevancy of sentences as was done by Allan, Gupta, and Khandelwal (2001) and Guo, Diaz, and Yom-Tov (2013). They both have notions of set-based evaluation, i.e., the metrics are analogous to Precision and Recall. Expected Precision and Expected Recall as described in Guo, Diaz, and Yom-Tov (2013), are precursors to Expected Latency Gain (ELG) and Latency Comprehensiveness (LC) measures of TST, respectively. However ELG and LC move on to a nugget based evaluation (Aslam et al., 2013), where a nugget replaces the notion of subtopics as described by Guo, Diaz, and Yom-Tov (2013). A nugget identifies a key relevant fact or information that may be represented in many forms across sentences returned by systems. TST adds a notion of latency discounting, i.e., gain from an update is penalized for being late.

McCreadie, Macdonald, and Ounis (2014) work on incremental update summarization, where the focus is on producing relevant and novel sentences as updates over time. They investigated adaptive filtering techniques and utilized the macro F_1 of ELG and LC for evaluation of their incremental update summarization system. For TST 2014, Aslam et al. (2014) utilize an F_1 measure of normalized ELG and LC for evaluation.

2.1.5 Other TREC Tracks with Temporal Leanings

Microblog Track

The Microblog track at TREC has seen 5 iterations since its inception in 2011 (Ounis et al., 2011). The real-time ad-hoc search task at the microblog track required systems to return recent and relevant tweets. The evaluation measure for this task was P@30; P@k is the precision of the top-k documents (in this case, top-30 tweets) returned by a system (Büttcher, Clarke, and Cormack, 2010). The 2012 iteration of the track ran a real-time filtering pilot task (Soboroff et al., 2012). Microblog track 2014 (Lin et al., 2014) introduced a temporally anchored version of real-time ad-hoc search. Systems were required to return 1000 likely relevant tweets up to a specific time instant. The evaluation was done using MAP; however, P@30 was also reported.

Microblog track 2014 also introduced a tweet-timeline generation task, requiring systems to return relevant tweets that constitute a summary about a topic. The summary was required to be a list of chronologically ordered tweets up to the time at which the query was executed. For evaluation, two measures, cluster precision and cluster recall, were formulated. Tweets containing the same information were grouped into semantic clusters. The system was evaluated on how many clusters were represented by their retrieved tweets. The task is essentially one of generating a retrospective summary for a topic using a chronologically ordered set of tweets. Wang et al. (2015) further analyzed the semantic clustering method of evaluation and found that although assessors may differ in their judgement of which tweets constitute a semantic cluster, the differences do not affect relative performance measurement.

Microblog track 2015 introduced the real-time filtering task, requiring the systems to (a) push likely relevant tweets to the user (at most 10 per day), or (b) compile a digest of 100 tweets relating to a topic. The push task is quite similar in nature to the TST except that the former is more stringent in its requirement of updates per day. Indeed, the push task utilizes the TST metric of Expected Latency Gain with graded relevance for the returned tweets. For the digest task, the evaluation was done using NDCG@k; NDCG@k is the normalized discounted cumulative gain for the top-k documents returned by a system (Büttcher, Clarke, and Cormack, 2010).

Novelty Track

The Novelty track at TREC (Soboroff and Harman, 2003) has some similarities with the temporal summarization track. The main task for the novelty track was to find relevant and novel sentences from an ordered document stream. A key difference between the novelty track and TST is the scale of the corpus in TST and that latency was not accounted for in the novelty track. The topics for the novelty track were relating to events or opinions. The organizers provided relevant documents to participants from which the relevant and novel sentences were to be identified. The 2004 iteration of the novelty track (Soboroff, 2004) supplied a number of non-relevant documents along with relevant documents to participants. For evaluation, the novelty track matched system returned relevant/novel sentences against assessor identified relevant/novel sentences. The primary evaluation metric was the F_1 of precision and recall of a system's returned set of sentences.

Knowledge Base Acceleration Track

The Knowledge Base Acceleration (KBA) track promotes research on building a knowledge base for a given entity. Specifically, for a given entity, KBA requires systems to return content relating to the entity with the passage of time. For instance the cumulative citation recommendation task at KBA (Frank et al., 2013) requires systems to return pages from a time ordered corpus that are citation worthy in a given entity's Wikipedia article. Other tasks include finding changes in values for key entity properties or attributes (streaming slot filling task). The KBA track also utilizes the F_1 measure as its primary evaluation metric.

2.2 User-behavior Modeling and Simulation for Evaluation

An important research area in IR is the study of how evaluation measures of IR tasks are reflective of human performance. As per the Cranfield paradigm (Cleverdon, 1967), an IR system can be evaluated by utilizing (i) a document collection, (ii) a set of topics, (iii) relevance assessments for each document in the collection with respect to the set of topics. The retrieved results from systems for each topic can then be evaluated against the respective relevance judgements.

As IR tasks and corresponding user interfaces evolve, it is important to understand how a user might use a system. There have been various studies on whether simpler measures like precision and recall, and their derivative measures like average precision,

can adequately reflect user performance. Such studies typically aim to determine if real users actually find benefit in using a system A that performs better on, say precision, than another system B. Al-Maskari et al. (2008) show that system performance does correlate with change in human performance, and that users can distinguish between systems that are slightly different. Smucker and Jethani (2010) further substantiate this finding and point out that users change their behavior when using systems having different precisions for retrieved results. They further note that, Cranfield-style evaluation works well when the IR task and user interface are affine to the corresponding Cranfield-style metrics. For instance browsing down a ranked list of documents (no summaries), and assuming that each document takes the same amount of time to read, then the precision of the list of documents is indeed a valid metric to measure performance. However, a user's behavior might change if reading longer or shorter documents, or if reading a highly relevant document after a non-relevant document, or when reading snippets that hint on relevancy only to find contradicting content in the documents. Evaluation measures that are sensitive to such user considerations may help to better measure system performance in terms of the system's utility for a user performing an IR task.

2.2.1 User-models and Derived Measures Evolution

One of the major changes in system evaluation was the shift to graded relevance judgements and the conception of the Discounted Cumulative Gain (DCG) measure (Järvelin and Kekäläinen, 2002). DCG evaluates systems based on their ability to return highly relevant documents at high ranks. A discount is applied when higher grade documents

are found to appear below lower grade documents in the ranked list produced by the system. A normalized version of DCG (nDCG) scores the DCG of a system relative to the ideal ranking's DCG for the system. The normalization enables comparison of system's performance across topics. From a user perspective, DCG models a user's dissatisfaction of finding a relevant document lower in the ranked list (or "later" in time), by discounting the gain obtained on reading a document as a function of the rank at which the document appears.

Moffat and Zobel (2008) introduce Rank-biased Precision (RBP), an evaluation measure that utilizes a simple user model of a user examining the next document in a ranked list with probability p, or exiting the search with probability $1 - p$, to compute the average utility gained by a user while reading a ranked list of documents. They point out that using identified relevant documents in TREC-style pooling evaluation with low pooling depths can result in Average Precision (AP) to be over-estimated. RBP, on the other hand, does not require the knowledge of the number of relevant documents in the collection and only observes the quality of the results returned by the system that are read by the user.

Chapelle et al. (2009) developed the Expected Reciprocal Rank (ERR) measure that improves upon RBP as well as DCG. They point out that the DCG follows a *position*-based model, in that the browsing behavior of the user solely depends on the current depth of the ranked list that the user has reached. This model, however, fails to consider that a user's probability of reading a document at a given rank can depend on the quality of the document at the previous rank (a *cascade*-based model). ERR effectively computes the inverse of the expected effort required to satisfy the information need of a user. ERR was shown to correlate well with data derived from click-logs of a web search engine, and as

such, is well suited to reflect user performance.

There have been many investigations conducted into observing users and understanding the underlying models of user search behavior. Of late, Interactive Information Retrieval (IIR) and its evaluation (Kelly, 2009) has received growing interest from researchers. The user-interface of a system has emerged as a key aspect that influences user performance. The simple model of a user browsing down a ranked list of documents is not necessarily reflective of actual user behavior. Search interfaces can be faceted allowing for browsing sublists from a ranked list (He et al., 2015). Search interfaces could also suggest similar articles to browse, enabling the users to guide systems to return similar documents to known relevant ones (Smucker and Allan, 2006). These interfaces are significantly different from the "user browses down a ranked list" paradigm.

It is thus crucial to understand, what the user might actually do, in terms of interactions with the system interface and retrieved results, in order to estimate a performance score reflective of user performance. Clarke et al. (2013) and Azzopardi and Zuccon (2015) collate excellent compendiums of research on modeling user behavior for evaluation. Clarke et al. (2013) outline and discuss procedures for system evaluation using simulation of user-behavior over a system's interface. Azzopardi and Zuccon (2015) discuss and demonstrate what underlying models drive the search processes a user might follow, and how they can be used for understanding the user behavior over a given search system. For instance Azzopardi (2014) examines how a user's information search strategies align with economics theory, and tries to predict optimal search behavior given a search system (retrieval method in combination with the user interface).

2.2.2 Grounding User-Oriented Evaluation in Time

Smucker and Clarke (2012d) proffer the view that performance evaluation measures such as nDCG, RBP, ERR, can be expressed in an abstract form as

$$\frac{1}{\mathcal{N}} \sum_{k=0}^{\infty} g_k d_k \tag{2.2}$$

where, k is the rank, g_k is the gain experienced after reading document at rank k, and d_k is the discount associated with the document at rank k. $\mathcal{N}$ is the normalization factor, usually a value that constrains the final evaluation score to the interval [0,1]. The sum is usually limited to a specified maximum rank depth K such that $1 \leq k \leq K$. For instance, for nDCG@10, one would set $K = 10$, g_k would depend on the relevance grade of the document, d_k is $1/log_2(1+k)$, and $\mathcal{N}$ is the ideal value of gain achievable at rank K.

Smucker and Clarke (2012d) point out that these measures, though they incorporate a model of user behavior, the model assumes that users take a uniform amount of time to read each successive document in a presented ranked list. In reality, a user goes through a sequences of events: reading a document surrogate (snippet), deciding if browsing the document could be worthwhile, reading the document, going back to the ranked list if document did not satisfy the information need, and repeating the process starting at the next snippet. Thus, as the user interacts with the system, the user has the navigate through multiple states (each having an associated cost) and each transition from state to state has an associated probability (Fuhr, 2008).

Smucker and Clarke (2012d) develop the Time Biased Gain (TBG) measure for evaluat-

ing system performance, with the insight that the cost associate with each user interaction can be represented with time, e.g. the time taken to read a snippet, read a document and interact with the interface, in order to complete the search process. They go further, in fact, modeling the time at which a user will stop the search process; done via estimating the probability that a user will continue to search until time t, represented by a decay function $D(k)$. A reformulation of Equation 2.2 gives us

$$\frac{1}{\mathcal{N}} \int_{k=0}^{\infty} \frac{dG}{dt} D(t) dt \tag{2.3}$$

where, dG/dt models the probability of gain being considered in full with the passage of time. For instance, a user may not necessarily find it satisfactory to read one moderately relevant document after reading 20 non-relevant ones.

Smucker and Clarke (2012d) acknowledge that measuring instantaneous gain could be a hard endeavor and they suggest the more practical approximation

$$\frac{1}{\mathcal{N}} \sum_{k=0}^{\infty} g_k D(T(k)) \tag{2.4}$$

where, $T(k)$ is the expected time required by a user to read documents up to rank k, and g_k is assumed to be uniform across all documents encountered. g_k represents the benefit or gain experienced on reading the document at rank k. Thus, TBG measures for an evaluating system, its performance in terms of expected number of relevant documents that can found by users.

Undoubtedly, TBG will change depending on the IR task, the system's user interface,

and the associated user behavior. A key aspect for evaluation with TBG is the calibration of the metric for the underlying user-model. Once calibrated, the TBG can return an expected value of gain per user. The advantage with TBG is that calibration need only be done for the amount of time that users spend for various sub-tasks (interactions, reading retuned content) involved in using the search system.

Smucker and Clarke (2012b) first describe the user-model of a user browsing a ranked list of summaries that link to documents. They then, calibrate (estimate costs of states, and probabilities of transitions between states) this model of user behavior by analyzing data from conducted user-studies. They finally estimate TBG via stochastic simulation of the user model. Smucker and Clarke (2012a) extend this idea further; they posit that modeling a whole population of users using multiple user models allows us to get a distribution of expected gain (TBG values) for the population of users. This is advantageous because, given how individual users use search systems differently, we can estimate how users with *different* search behaviors might experience gain while using a system. The underlying user model need not change, however, some users may only read summaries, some others may search for a lesser time, some users may read slower (or faster) than others. Modeling user variance allows system designers to observe how different classes of users might benefit when using their system.

Clarke and Smucker (2014) move beyond the ranked list interface (and the corresponding user-model) for TBG. They develop Time Well Spent (TWS), an evaluation measure that essentially measures the time spent reading relevant content. Building on the prior work, they also develop methods to enable pairwise comparisons between systems, given the user population's performance distribution. A key contribution of their work was the

development of a reading speed model for a user population. Utilizing a model of reading speed within the TWS/TBG evaluation framework, allows the extension as well as the application of this evaluation approach for other IR tasks. Chapter 5, develops a model for user behavior for information access over a stream of updates produced by a system, and demonstrates the use of this model to evaluate the performance of systems that produce a stream of news updates.

A potential advantage in using the TBG/TWS evaluation framework is measurement of expected gain for a given *population* of users. Given that most user interaction data is only available to proprietary organizations, Balog et al. (2014) discussed the role that living labs could play for system evaluation and observation of user behavior. Azzopardi and Balog (2011) illustrate how living labs may benefit research groups by providing a common platform to observe user interactions on IR systems, for a variety of tasks. Although the benefits of a living labs evaluation cannot be under-stated, the logistics of the setup such an evaluation can be challenging. The TWS paradigm provides an alternative, albeit approximate, evaluation framework that can simulate populations of users working with competing systems. The respective user-models and their calibration is a key requirement for estimating user performance accurately using the TBG/TWS evaluation framework.

Grounding Evaluation in Text

The TBG/TWS evaluation framework grounds the evaluation in terms of time spent by a user for a search task. An alternative approach is to ground the evaluation in term of the *text* read by a user. Sakai and Dou (2013) propose a unified framework for evaluation of

sessions, summaries, ranked lists, all based on the text content that is read by a user. The set of text strings, all concatenated in order that a user reads them, is termed as a *trailtext*. A trailtext can contain, snippets, summaries, portions of documents, and even span across sessions. Based on the amount and position of relevant content in the trail-text T_{trail}, a system's response is evaluated using U-measure as

$$\frac{1}{\mathcal{N}} \sum_{p=0}^{|T_{trail}|} g(p)D(p) \tag{2.5}$$

where, p is the index of a string in T_{trail}, $g(p)$ is the gain experienced when reading the string at index p, and $D(p)$ computes a discount that increases with p.

The U-measure incorporates a model of user-behavior to determine the text content that a user might read. The authors describe methods to estimate trailtext for different IR tasks and they acknowledge that eye-tracking of users might give a better estimate of trailtext. Clarke and Smucker (2014) point out that although the U-measure is able to incorporate models of user-behavior to estimate the text that is read, it does not account for the cost of user interactions with the system's interface, and also, it assumes that all users read at the same pace. In contrast, the TBG/TWS evaluation framework subsumes the cost of reading content as well as system interaction costs using time as the grounding unit for the measurement of user performance.

2.2.3 User-behavior and Temporal IR

User behavior analysis for temporal IR tasks is relevant to the research done in this thesis. Although there is no known model of user-behavior for streaming information access (to the best of our knowledge), behavioral analysis of temporal IR tasks can help guide us towards the development of an adequate approximation (cf. 5.1). Although not all discussed research is directly applicable to our work, most of the discussed research potentially aids in the development of future work.

Analysis of user behavior for news information access was previously undertaken over news portals, such as the work done by Dezsö et al. (2006) wherein the authors analysed visitation patterns of users over articles published on a news portal. They found that users visit a particular news article with a power law distribution. Yom-Tov and Diaz (2011) show that geographic distance from an event, as well as social separation, both affect the information need of users for the topic. Using the query-logs from Yahoo for 3 major events, they tried to model the user preferences for information related to the event. Joho, Jatowt, and Blanco (2015) utilized the NTCIR Temporalia TIR task data, and observed past recent and future information needs of participants in a user study. They found that current search engines work best for recent, then past, and worst for future information needs. Users add temporal expressions to queries although the benefit of doing so varies across topics. They identified areas for improvement in the detection temporal intent, vocabulary evolution as topics evolve, future oriented searches such as predictions, shopping or travel itineraries. Data like this may find use for calibrating MSU (Section 7.3.1).

Teevan et al. (2013) provide some insight about the latency requirements of users while performing search tasks and they construct a probability distribution over the time that a user might wait for T minutes before giving up searching. The distribution has exponential decay over time. This distribution is reminiscent of the decay function modeled in TBG. Crescenzi, Kelly, and Azzopardi (2015) show that users under time constraints query more, and view/examine fewer documents, than those not under time constraints. Understanding user behavior and system responses in temporal contexts such as these, is key to developing an evaluation measure predictive of user performance.

Azzopardi (2009) describes a stream of documents as an alternative perspective to the ranked list presentations of the IR process; the documents that a user views during the search process constitutes a stream. Such a perspective generalizes to different kinds of interactive IR tasks. They lay the ground work for usage based evaluation measures in temporal contexts. They identify the requirements for a stream-centric view of evaluation; what exactly constitutes a stream, dealing with unjudged documents in the stream, comparing streams of different lengths, how to model and predict user performance; the necessity of observing users to estimate the model of usage-patterns. They explored measures based on batches, sliding windows, defined periods of time or sessions. They propose a *relevance frequency* measure: the rate of encountering relevant documents in time. Interestingly, the idea that the user views a stream of documents is similar to the trailtext idea from Sakai and Dou (2013) (wherein, the "documents" are strings that a user reads).

Sessions

Baskaya, Keskustalo, and Järvelin (2012) note that IR tasks are usually conducted in sessions, and, the search interface and user search strategies, both influence the user performance. They identify and measure the costs (in terms of time) involved for each sub-task (e.g. scan result list, click on result, reformulate query) of a search session. Costs could be higher for re-querying on a mobile device and thus users may reformulate less in order to keep session costs lower. They show that search strategies change with the availability of time. Also, a higher querying costs is correlated with longer result list scans.

A metric similar to MSU is the Expected Global Utility (EGU) measure developed by Yang and Lad (2009). They present an algorithm and an evaluation method for information distillation with the goal of optimizing utility of ranked lists over multiple sessions. The novelty in EGU was that user interaction patterns over a ranked list of documents were modeled, for computing the expected utility of a system by estimating probabilities over users' browsing paths. Kanoulas et al. (2011) further address evaluating multi-query sessions and extend the simulation models of EGU, as well as the session-nDCG measure (Järvelin et al., 2008). Baskaya, Keskustalo, and Järvelin (2013) further investigate the modeling of query reformulation for multi-session evaluation.

The sessions for EGU were 12 days apart in a 4 month period as per the TDT4 corpus, and for each session, a ranked list was presented to the uses. TST on the other hand deals with updates closer to real-time and the evaluation measure considers the quality of the returned content as a whole. MSU is closer in spirit to EGU, however, MSU adopts a simpler user model (of a user reading updates in reverse chronological order at every

session), and the measure is designed to simulate a whole population of users who may not necessarily access the system periodically.

Summarization

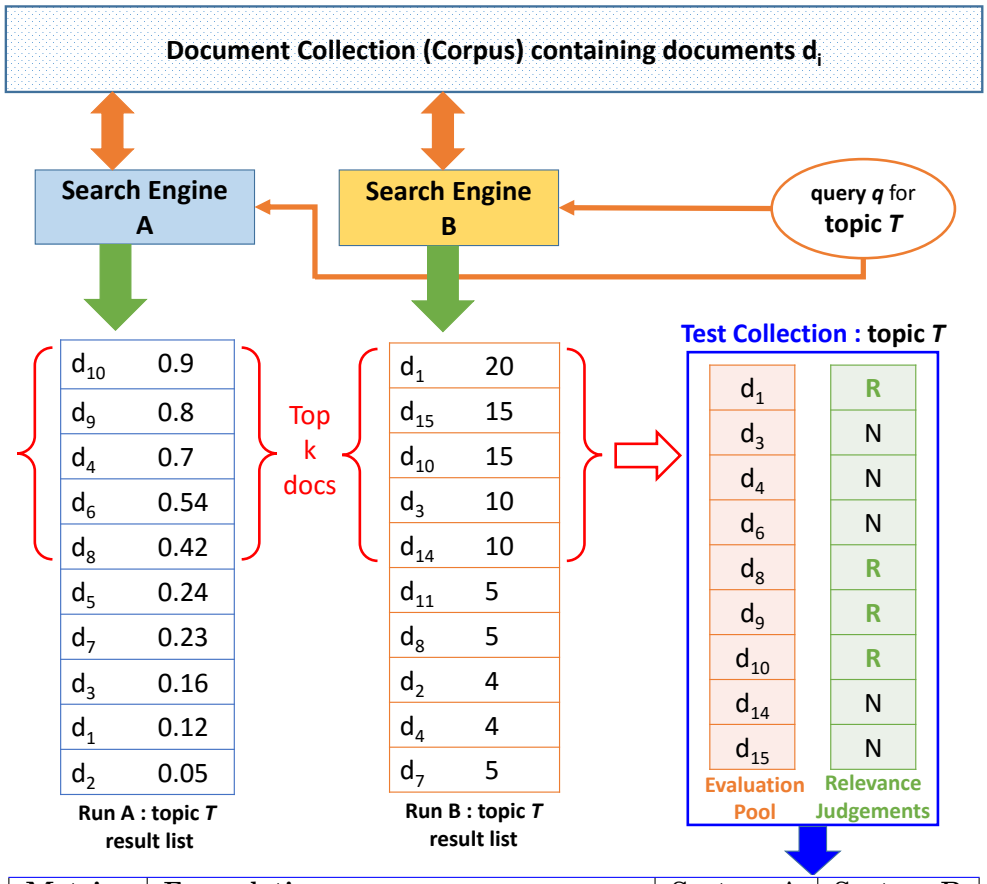
Summarization systems are generally evaluated using the ROUGE evaluation framework (Lin, 2004) where system generated summaries are compared to a human generated gold standard. TST builds a temporal summary over time (period of interest), and it may be possible to have a human construct a gold standard summary after the fact. An interesting approach to evaluate temporal summaries was put forth by Kedzie, McKeown, and Diaz (2015). They construct a gold standard summary by concatenating all nuggets for a TST topic. However, ROUGE requires that the model summary and the system generated summary have the same number of sentences. They, therefore, sampled with replacement system generated updates numbering equal to the number of sentences in the gold standard summary. The final evaluation score was the average ROUGE score over 1000 sampled summaries.

2.3 IR Evaluation: Test Collections and Evaluation Measures

To create a test collection, the Cranfield paradigm of evaluation advocates the use of (i) a document collection, (ii) a set of topics, and (iii) a set of relevance judgements for each document against each topic. A test collection such generated can be an effective tool

to assess the performance of information retrieval systems. Needless to say, for a large document collection and/or a large number of topics, finding a *complete set*—identifying all relevant documents across topics—of relevance judgements can be a non-trivial, if not an unfeasible, task. Sparck-Jones and Van Rijsbergen (1975) suggested *pooling* as a means of finding relevant documents more efficiently; the pooling method has become the de-facto standard method used at TREC to find relevant documents. At TREC, participating systems submit a *run*—system output that is usually a set of (document, score) tuples, where the score is a system's estimate of the document's likelihood of relevance. Given a set of outputs of retrieval systems, a *depth*-pooling method selects from each system output, a specified number of top ranked documents. The union of these sets of top ranked documents forms the *evaluation pool* that assessors judge for relevancy, resulting in a judged set of documents, i.e. the *test collection*. Systems return results and pooling is performed for each topic in the test collection.

Figure 2.1 describes the pooling process for a single topic. For an ad-hoc search task, given a document collection (or *corpus*), and a *query* string—that represents the search *topic*, various search engines return a ranked list of documents that are likely relevant to the query. For evaluation purposes the output of the system (the run) is representative of the system itself. Each search engine may have different ranking criteria depending on the specific methods used to compute similarity between the query and each document in the corpus. The degree of similarity is usually expressed as a numeric score based on which the documents are ranked. A document's score can thus be considered as a surrogate for the likelihood of relevance of the document for the given query. As Figure 2.1 shows, a document could be ranked differently by various systems. Given a specified depth k, the



Metric	Formulation	System A	System B
Precision	$\|Relevant \cap Returned\| \;/\; \|Returned\|$	0.4	0.3
Recall	$\|Relevant \cap Returned\| \;/\; \|Relevant\|$	1.0	0.75
P@k	$\|Relevant \cap Returned[1..k]\| \;/\; k$	0.6	0.4
Average Precision	$1/\|Relevant\| \cdot \sum_{i}^{\|Returned\|} relevant(i) \cdot P@i$ where, $relevant(i) = 1$ if the document at rank i is relevant	0.761	0.524

Figure 2.1: An example of the pooling process for a topic. Given a document collection and a query: (i) search engines return a ranked list of (document, score) tuples; (ii) the top-k tuples from each ranked list are *pooled* together; (iii) the pooled documents are assessed for relevancy by human assessors; (iv) the resulting test collection is used to compute system performance over appropriate evaluation metrics.

top k ranked documents are selected from each run. These are then *pooled* together and the documents in the pool are then adjudicated by qualified human assessors who determine whether each document in the pool is relevant or not. The documents not in the pool are assumed to be non-relevant. The adjudication process results in a test collection for the given topic. The test collection, once constructed, is utilized to evaluate the quality of the ranked lists returned by each system using appropriate evaluation metrics such as Precision, Recall, P@k—Precision at top-k documents, and Average Precision (Büttcher, Clarke, and Cormack, 2010).

A search engine is usually evaluated on its ability to return relevant documents over non-relevant ones. A judged set of documents that is a representative sample of relevant and non-relevant documents from the document collection may not work because of the number of non-relevant documents far exceeds the number of relevant documents; modern corpora range in the order of hundreds of millions of documents. Furthermore, labelling a presumably large random sample could be expensive in terms of time and effort, especially for multiple topics where each document needs to be judged for relevancy against each topic. The intuition underlying depth-pooling is that top ranked documents are more likely to be relevant, and thus relevant documents can potentially be found earlier/faster. These top ranked documents allow us to better compare between systems; e.g., good search engines would return more numbers of relevant documents at higher ranks. Salient research findings regarding the pooling method are discussed in Section 2.3.1. Alternative pooling methods and techniques to combat incompleteness in relevance judgements are highlighted in Section 2.3.2.

A fixed depth k for pooling is assumed to provide a fair representation in the pool

for each system. The resulting test collection allows for repeatable experiments to be performed and more importantly, it enables comparisons between systems through performance measurement. System performance is measured via a performance metric, and thus performance measures and their properties require careful consideration for any given IR task, along with the underlying test collection.

Our work in Chapter 4 investigates the effects that duplicate sentences have on the evaluation of temporal summarization systems, when they are added to the test collection. In Chapter 5 we present a novel user-oriented evaluation measure for update filtering systems, called modeled stream utility. In Chapter 6, we present an alternative pooling strategy that uses the probability of a sentence being read, as the basis for selection into the evaluation pool, rather than the system assigned score as is done for depth-pooling. In this section we discuss the related work that influences or impacts our research with regards to the evaluation measure, pooling and test collection construction.

2.3.1 Pooling and Test Collection Construction

Test collections mimic real usage scenarios in the form of topics and relevance judgements, and help produce a mean performance evaluation over all topics for a system. Factors affecting test collections and the resulting evaluation include the number and type (having more or less number of relevant documents) of topics, as well as the depth up to which result sets were pooled. Pooling is done with the objective of finding as many relevant documents as would be needed, such that if the unjudged documents are assumed to be non relevant, the judged set can be considered to be a complete set, i.e., a set that contains

all relevant documents for each topic. The construction of a true complete set requires a judgement for every document in the collection with respect to every topic, which is a formidable and likely an unfeasible task. However with ever larger corpora, it is harder to generate a complete set of judgements because of the much smaller size of the judged set relative to the document collection. The end goal is to have a set of relevance judgements that allow for system performance to be measured over prescribed evaluation metrics so as to find out which systems perform better than others.

Zobel (1998) investigated the TREC standard pooling methodology with regards to (i) whether pooling up to limited depths produces reliable performance measurements for systems that did not contribute to the pool, (ii) if depth-pooling finds most relevant documents, and (iii) if the system comparisons can be trusted. They found that limited depth-pooling can indeed produce reliable performance estimates for new systems given large pool depths, and that system comparisons for statistically significant differences can be trusted. However they observed that a large portion of the relevant document set were not found by the depth-pooling method for shallow pool depths.

Voorhees (2000) demonstrated that comparative evaluation of systems remains stable even when the relevance judgements in the test collection are substantially varied. For instance, for the TREC-6 ad-hoc test collection, 4 relevance judgement sets were compared; the judged set generated by NIST, the relevance judgements generated by Cormack, Palmer, and Clarke (1998), the intersection of these sets, and the union of these sets. They found that relative performance measurement using the different relevance judgements does not change significantly, i.e. the Kendall's τ rank correlation between system rankings induced by the different sets of relevance judgements remains greater than 0.8, as long as

at least 15 topics are represented in the test collection. For 25 topics, one can expect a Kendall's τ greater than 0.9.

During test collection construction, typically, a single assessor generates an information need, formulates the topic (a description) for the same, and also assesses the pooled documents for the topic. Trotman and Jenkinson (2007) investigated the effect of assessing a topic using multiple assessors and found no appreciable affect on evaluation of systems. Bernstein and Zobel (2005) found that removing documents containing similar information improves system performance on novelty based measures. Our experiments with including duplicate documents shows that, at least for a temporal filtering task, precision (ELG) and recall (LC) can both show differences in absolute performance for some systems. By adding duplicates, we increase the size of the judged set, i.e., we vary the set of relevance judgements for the test collection; however, there is no appreciable difference in the relative performance of systems (Chapter 4). Quite possibly, pooling based on the probability of updates being read (Chapter 6) may not change the relative performance measurement in comparison to the standard depth-pooling. However, we leave this investigation for future work, pending procurement of relevance judgements for pools created using respective methods.

Retrieval methods perform differently on different topic types. Ideally, relevance judgements for a large number of topics, and a stable effectiveness measure that produces appreciable difference in scores of systems, would help in constructing a reliable test collection. Voorhees and Buckley (2002) point out that more topics require procuring a large number of relevance judgements, making the construction of the test collection costly and they investigate by how much should system scores differ in order to produce a meaningful

comparison between systems.

Test collections can be biased, i.e., the algorithms used for retrieval by various systems, may cause a disproportionate membership in the pool for documents returned using similar methods. As a result, novel retrieval algorithms may not be fairly evaluated as the judged set of documents may not adequately represent the set of relevant documents returned by the new method. The problem is exacerbated when the judged set of documents is small. Buckley et al. (2007) show that smaller sized pools are not immune to bias; comparing systems that did not contribute to the pool with contributing systems is unfair to the former. However the bias only impacts a score negatively (because of unjudged documents). This does not necessarily impact the development of systems via improvements against a baseline. However, some "external" systems (that did not contribute to the pool) may not be able to show improvements because of the high numbers of unjudged documents. Cormack and Lynam (2007) develop a method to estimate the power and bias of pooling strategies. They investigate the power and bias of pools for different numbers of topics and across various depths for pooling. They found that evaluation using MAP is better when pool depth is shallow and topics are many, than otherwise. Analysis along similar lines is necessary for our probability-based pools (Chapter 6). However, such analysis needs relevance judgements for the respective pools, an exercise we leave for future work.

2.3.2 Alternatives to Standard Depth-pooling

Pooling results in test collections that are largely incomplete, i.e., all relevant documents for each topic are not necessarily identified. As a consequence, there is much research

interest in whether incomplete sets of relevant judgements can adequately measure system performance. A related line of research is the development of pooling methods that aid in the construction of test collections more economically or expeditiously.

Alternative Pooling Methods

Zobel (1998) proposed a new pooling method based on the observation that queries for which systems return a higher number of relevant documents in an initial depth pool are more likely to have unjudged relevant documents. Judgements are effected incrementally for such queries depending on the relative cost of procuring the new judgements against an estimated number of new relevant documents likely to be found.

Cormack, Palmer, and Clarke (1998) developed 2 pooling strategies, Interactive Searching and Judging (ISJ), and Move-to-Front (MTF). ISJ constructs a test collection by combining the relevance judgements made by a group of people for given topics. Essentially, a team of researchers uses an interactive search system; each user formulates queries for topic and judges returned documents, until the user decides that the information need is reasonably satisfied. MTF priorities judgements on documents from systems that have most recently contributed relevant documents to the pool.

Aslam, Pavlu, and Savell (2003) develop an approach for result fusion, pooling and evaluation, based on the Hedge algorithm (Freund and Schapire, 1997). Systems are considered as experts advising on relevance of documents. For every relevant document judged, trust increases in all systems that returned that document. Correspondingly, for every non relevant document, trust decreases in all systems that returned that non-relevant document.

Using trusted systems over many iterations of the hedge algorithm results in finding more relevant documents than simply using the depth-pooling strategy.

Carterette, Allan, and Sitaraman (2006) create a strong connection between the evaluation measure and test collection construction. They develop a method to select documents to judge based on average precision in a way that the selected documents can help maximize the performance differences between systems. The resulting set of judgements can be obtained expeditiously and result in a ranking of systems that matches the true rankings with high confidence. Aslam, Pavlu, and Yilmaz (2006) propose a random sampling method for pooling and evaluation of runs using standard effectiveness measures such as MAP. They show that only 4% of a known depth-100 pool is required by their method in order to get an accurate estimate of the effectiveness measure for systems.

Incomplete Relevance Judgements

Buckley and Voorhees (2004) investigate the effect of incomplete relevance judgements on system evaluation and find that most evaluation metrics are not robust to incomplete relevance judgements. They develop *bpref*, an evaluation measure that estimates the MAP of a system when given an incomplete set of relevance judgements. Yilmaz and Aslam (2006) improve upon bpref and develop induced AP, sub-collection AP and inferred AP. Inferred AP in particular computes the expected precision at ranks for randomly selected relevant documents from the collection; the average expected precision gives us the inferred AP. Inferred AP gives a highly accurate estimate of AP even when 30% of the complete relevance assessments are utilized.

Another way to deal with incomplete relevance judgements is to not consider for evaluation all those documents that have not been assessed. Sakai (2008b) and Sakai (2008a) show that condensed-list relevance judgements—where unjudged documents are ignored for evaluation purposes or *elided*—produce over estimated performance scores. In comparison, considering unjudged documents as non-relevant under estimates system performance. They found that eliding unjudged documents is not an effective evaluation strategy in the presence of system bias while pooling, i.e. the discriminative power (or the ability to detect pairwise statistically significant differences) of measures utilizing such condensed-lists may not be any better than simply utilizing a non eliding evaluation. They find that AP can work well with elided judgements, and they advise that sampling methods need to be used carefully in consideration with the evaluation measures.

Aslam and Yilmaz (2007) develop a method to infer a complete set of relevance judgements, given a set of sampled and judged documents, by computing estimates of AP of given systems and estimates of numbers of relevant documents. The inferred set of judgements were found to yield a comparable evaluation as when using a complete set of relevance judgements. Büttcher et al. (2007) utilize document classification techniques to predict relevance of documents, given a set of incomplete and biased judgements (biased towards contributing systems). They recommend that for new retrieval methods (that did not contribute to pooled judgements) to benefit from the test collection, a classifier can be trained on the known relevant documents and the relevance of documents returned by the new methods can be predicted.

2.3.3 Evaluation Measures and Systems Comparisons

Along with test collections, evaluation measures and statistical comparison techniques are an integral component of the IR evaluation toolkit. A good evaluation measure should be robust (or stable)—be consistent in differentiating good systems from bad ones when experimental settings are changed, as well as discriminative—be able to detect statistically significant differences between systems.

Stability of an evaluation measure is the ability of the measure to produce consistent measurements of relative performance across changes in experimental settings (e.g. change in number of topics). Buckley and Voorhees (2000) investigate which evaluation measures are stable, for instance, they find that MAP is more stable than P@k when the number of topics is changed. They recommend a topic set size of at least 25 to obtain estimates of performance having low error rates.

Sakai (2006) utilizes bootstrap hypothesis tests to measure the *discriminativeness* of IR metrics. The bootstrap sampling method tries to estimate the distribution of the data by re-sampling from observed data. Hypothesis tests based on the bootstrap do not need the presumption of the data being normal or symmetric. The authors use the bootstrap hypothesis test to compare retrieval methods. Smucker, Allan, and Carterette (2007) further compare statistical significance tests (sign, Wilcoxon, t-test, bootstrap and randomization) for pairwise system comparison. They recommend that randomization or bootstrap be used for statistical significance tests, as these tests permit the use of any appropriate test statistic (other than mean difference) to compare 2 systems.

IR experiments frequently require comparing ranked lists of systems. Yilmaz, Aslam,

and Robertson (2008) develop τAP, a rank correlation measure that considers differences in top ranks as more important when comparing 2 ranked lists. This improves upon Kendall's τ in the sense that Kendall's τ gives equal weight to a change in rank at any position in the ranked list; however, for IR experiments, we are usually most interested in knowing the changes at the top of the rankings.

2.3.4 Nugget-based Test Collections

Nuggets can be considered to be atomic facts about a topic. They are represented as natural language strings that describe a concept that is relevant to an information need. Nuggets have been utilized for evaluations of IR systems for the TREC Question-Answering tracks (Voorhees, 2005; Voorhees, 2006; Dang, Lin, and Kelly, 2007) as well as the temporal summarization tracks (Aslam et al., 2013; Aslam et al., 2014). Nugget-based test collections (Pavlu et al., 2012; Rajput et al., 2012) have been shown to be scalable—provide for large number of judgements, and reusable—work for novel systems and new documents in a dynamic collection.

Evaluations based on nuggets acknowledge that the same concept can be represented in natural language in multiple ways, for instance, strings representing the same concept might differ in synonyms, paraphrasing, syntax, amongst other features (Table 2.1). The representations of nuggets have evolved over the different IR tasks they have been used for. The sources from which the nuggets are derived or extracted also differ across the tracks. For the QA track, nuggets were extract from the returned system responses. For

Nugget: **the train crashed at the buffer stop**
Representations in text: *...train slammed into the end of the line...* *...slammed into the end of the line...* *...smash into a barrier...* *...hit the barrier at the end of the platform...* *...train plowed into a barrier...*

Table 2.1: A TST 2013 nugget and its various representations in returned sentences as confirmed by NIST assessors.

TST, nuggets were first extracted from Wikipedia[1] (Section 3.1.2). Rajput et al. (2012) developed a semi-automatic method to build a test collection by predicting document relevance and finding nuggets for a topic. They considered sentences of relevant document as candidate nuggets that were iteratively weighted to get a final list of nuggets for a topic. At the NTCIR 1CLICK-task (Kato et al., 2013), nuggets are further broken down into i-units. The 1CLICK task requires systems to return a short summary given the query, rather than a list of results. i-units are manually extracted from nuggets and are relevant (to the topic), atomic (further break down would result in loss of semantics) and/or dependent (entailment of one or more i-units).

The primary advantage to using nuggets for building test collections is that they explicitly codify what information is relevant. This is in contrast to document oriented relevance judgements wherein a very long document could be deemed relevant because it contains one relevant sentence somewhere in its content. This problem is partly alleviated by using graded relevance judgements, e.g., a scale of relevance: highly relevant, relevant, non-relevant. However, with nuggets, the focus is not whether a document is relevant or not,

[1] www.wikipedia.org

but rather the focus is on finding and identifying relevant material. Knowledge of various representations of relevant material (e.g. Table 2.1), can help (semi-)automate relevance judgements. For instance, for the QA track, given a set of nuggets and system responses, POURPRE (Lin and Demner-Fushman, 2005) and Nuggeteer (Marton and Radul, 2006) are evaluation frameworks that automatically evaluate systems' performances. Pavlu et al. (2012) first showed that manually extracted nuggets can be used to infer the relevance of documents given a set of nuggets. This helps in finding a much larger set of relevant documents. They further show that system evaluation using inferred relevance judgements is comparable with TREC-style evaluation. Rajput et al. (2012) extended this approach to build a nugget-based test collection in a semi-automatic iterative method (manual nugget extraction is eliminated), that results in a set of nuggets as well as a set of relevant documents for a topic.

2.3.5 User-models, Test Collections, and Evaluation Measures

Moffat, Webber, and Zobel (2007) develop methods that select documents to judge from across submitted systems, to differentiate best systems from the others, when evaluating over the RBP performance measure. In their methods, the judgement effort is shifted towards documents that can help compare the best systems with each other. Given the RBP user model, and the associated RBP score for a document at a given rank, documents are weighted and appropriately selected for judgement by assessors. This work is similar to our work in Chapter 6, wherein we utilize the MSU user model to select updates into a pool for assessment, based on the probability of them being read.

Radlinski, Kurup, and Joachims (2008) show that implicit feedback can be utilized to measure retrieval quality. Implicit feedback based measures include click through rates on a Search Engine Result Page (SERP), the dwell time—time spent reading a document, the number of page views, the abandonment rate, amongst others (Agichtein, Brill, and Dumais, 2006). However, such methods are arguably more suited to systems having a wide user-base.

Smucker and Clarke (2012c) outline that test-collections created using the Cranfield paradigm need not be at odds with user-oriented evaluation measures. They point out that evaluation measures inherently model the behavior of a user and predict user performance over a given system. Better user models may lead to better evaluation measures that would in turn lead to better estimates of user performance, provided that the test collection is relatively unbiased and complete.

Recall curves over time (Smucker, Allan, and Dachev, 2012), which in turn are inspired by recall curves over characters read (Lin, 2007) could serve as an alternative to the Latency Comprehensiveness (or Recall) measure of TST. In a similar vein, the concept of trailtext and U-measure (Sakai and Dou, 2013) could also serve as an alternative for TST-evaluation. Yang and Lad (2009) demonstrate an evaluation method similar to ours (Chapter 5), however it differs in user model and system interface, and it is has been developed over a much smaller corpus than TST. It will be interesting to compare against their method by calibrating MSU with a ranked interface presented at every session.

Evaluations over Time Intervals

Dietz, Dalton, and Balog (2013) put forth an approach for time-aware evaluation of streaming data. They carry out their experiments in the context of the KBA time-ordered document stream. Specifically, the documents should be "citable" by the Wikipedia article for the topic. Dietz, Dalton, and Balog (2013) find that to keep evaluation of systems fair, the evaluation should be aware of time intervals that contain bursts of intensity for a topic, and propose that the final evaluation score of a run could be the average over the evaluation scores at individual time slices. Kenter, Balog, and Rijke (2015) develop a metric to measure performance of systems that filter documents from streams, in the context of the KBA track, with applicability to similar stream filtering tasks. They demonstrate that traditional metrics like MAP, F_1, nDCG fail to address the performance of a system over time. Their method essentially estimates the trends of F_1 over batches (of time durations). A line fitted through F_1 scores of batches (a *trend* line) allows measurement of change in performance with time. Such research is aligned with MSU with the primary difference being that both the time granularity and bursty access is user-driven for MSU. However, information regarding the effect of bursts on user-behavior can greatly help improve upon MSU in its current form.

Chapter 3

The Temporal Summarization Track at TREC

The TREC Temporal Summarization Track (TST) (Aslam et al., 2013; Aslam et al., 2014) promotes research on the development and evaluation of IR systems designed for retrieving updates about breaking news events. Figure 3.1 illustrates the general Temporal summarization task. For each topic, a *query duration* determines the time interval for which a user is interested in a topic. Given a time ordered document stream, a temporal summarization system processes the documents in temporal order from the stream, and outputs (emits) updates it deems as relevant to the topic. The primary constraint for the task is that at any given time instant, retrieval (or processing) should not involve data from the future, i.e., the retrieval algorithm should respect the temporal ordering of documents for processing.

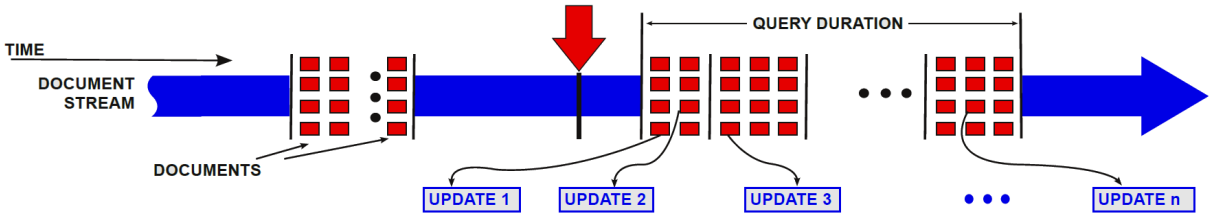


Figure 3.1: The Temporal Summarization task: Following a newsworthy event that occurs at some point in time (red arrow), the system must find and emit sentences concerning the event, from a time ordered stream of documents (blue arrow), for as long as the user is interested in the event (the *query duration*).

The temporal summarization task can also be looked upon as an on-line filtering task wherein sentences relevant to a topic are filtered from a stream of sentences. The number of updates to emit and the time at which to emit them is decided by the system; e.g., a system may emit a potentially relevant update as soon as it is processed, while another system may emit a fixed number of updates at regular or suitable intervals.

3.1 Temporal Summarization Track 2013

TST 2013 was the first iteration of the temporal summarization Track at TREC. Topics in the track were instances of event types: accident, bombing, earthquake, shooting and storm. Given a topic's query string, the track required that systems (runs) return sentences (updates) that are relevant to the topic, from the TREC KBA Stream corpus 2013 (Frank et al., 2014). Every update is associated with the timestamp at which it was emitted by the system.

3.1.1 Corpus, Tasks, Topics

Corpus

TST 2013 prescribed the KBA Stream Corpus 2013[1] (Frank et al., 2014) from which topically relevant updates were to be retrieved. The corpus contains over 1 billion documents with timestamps from 11,948 hours, spanning from October 2011 to January 2013. The documents in the corpus are mainly of 3 types; social (blogs and forums), news (from public news wires), and a subset of the links submitted to `bitly.com` (a URL shortening service). The documents in the corpus were serialized using Apache Thrift[2]—a software framework that allows cross-language development by enabling sharing of data structures via well defined interfaces. Each document in the corpus, apart from its raw text, also included named entity tags, detected language, and its tokenization into sentences and words, all of which were made available using the thrift interface and data structures for the corpus[3]. Although the thrift serialization made the corpus larger, extracting sentences from the corpus was a simple matter of calling the appropriate thrift interface.

Tasks

There were two tasks at TST 2013:

- Sequential Update Summarization (SUS): systems were tasked to return likely relevant updates for a topic.

[1] *KBA Stream Corpus*. `http://trec-kba.org/kba-stream-corpus-2013.shtml`. 2013.
[2] `https://thrift.apache.org/`
[3] `https://github.com/diffeo/streamcorpus`

- Value Tracking (VT): systems were tasked with tracking change in values for one or more event related attributes of the types: locations, deaths, injuries, displaced and financial impact.

Topics

Topics were news events of the type accident, bombing, earthquake, shooting and storm. There were initially 10 topics. Topic 7 was later discarded due to insufficient numbers of relevant updates in the collection. TST 2013 sets the query duration to be 10 days for all topics. Table 3.1 lists the topics for TST 2013.

The TST organizers also provided a training topic "iran earthquake" in order to aid the participants develop systems for the first iteration of the track. Figure 3.2 shows the training topic (as it would appear as a test topic). The topic contains the title, a description (the URL to the event's Wikipedia page), the query duration (`<start>` and `<end>` tags), the query string, the type of event, as well as desired attributes for value tracking.

3.1.2 Evaluation Method and Measures

The underlying theme for the evaluation of TST runs is that updates about events should be early, short and novel. Their evaluation measures compute a score for each system using a combination of functions that test each submitted update for relevancy, latency, verbosity and novelty. TST uses a nugget based evaluation framework. It has two main metrics Expected Latency Gain (ELG) and Latency Comprehensiveness (LC). These are analogous to precision and recall respectively. ELG (Equation 3.1) measures the average

```
<event>
    <id>TRAIN-1</id>
    <title>2012 East Azerbaijan earthquakes</title>
    <description>
        http://en.wikipedia.org/wiki/2012_East_Azerbaijan_earthquakes
    </description>
    <start>1344687797</start>
    <end>1345551797</end>
    <query>iran earthquake</query>
    <type>earthquake</type>
    <locations />
    <deaths />
    <injuries />
    <displaced />
</event>
```

Figure 3.2: Training Topic supplied for TST 2013.

quality of an update while considering the latency and verbosity of each update. LC (Equation 3.3) measures the latency discounted recall of nuggets for a topic. ELG and LC were used to evaluate submissions to the sequential update summarization task. For the value tracking task, scalar attributes were evaluated using Expected Error between reported values and actual values across the query duration, and location attributes were evaluated using the Vincenty distance. For this thesis, we do not focus on the value tracking aspects of the TST.

Phase 1: Gold Nugget Extraction

For evaluating system performance, the TST assessors first identified nuggets about an event from the Wikipedia article for the event. A nugget can be described as a short

id	topic	type	query duration (days)	Attributes for Value Tracking
1	2012 Buenos Aires Rail Disaster	accident	10.0	locations, deaths, injuries.
2	2012 Pakistan garment factory fires	accident	10.0	locations, deaths, injuries.
3	2012 Aurora shooting	shooting	10.0	locations, deaths, injuries.
4	Wisconsin Sikh temple shooting	shooting	10.0	locations, deaths, injuries.
5	Hurricane Isaac (2012)	storm	10.0	locations, deaths, injuries, displaced, financial impact.
6	Hurricane Sandy	storm	10.0	locations, deaths, injuries, displaced, financial impact.
7	June 2012 North American derecho	storm	10.0	locations, deaths, injuries, displaced, financial impact.
8	Typhoon Bopha	storm	10.0	locations, deaths, injuries, displaced, financial impact.
9	2012 Guatemala earthquake	earthquake	10.0	locations, deaths, injuries, displaced, financial impact.
10	2012 Tel Aviv bus bombing	bombing	10.0	locations, deaths, injuries.

Table 3.1: Topics at TST 2013 with their types, query durations and attributes for the value tracking task.

segment of text representing an atomic piece of relevant information. A timestamp was attached to every nugget as determined by the first occurrence of the nugget in the Wikipedia edit history for the event's article. Although the nuggets in the TST qrels (relevance judgements) have an importance grade assigned to them, the track's official metrics compute scores based on binary relevance of the nuggets, i.e., every nugget in an update contributes to a gain increase of 1. The list of nuggets for every topic was prepared before participating systems were evaluated.

Phase 2: Update Nugget Matching

In the second phase of TST evaluation, the top 60 updates as determined by system assigned scores from each run (system) were pooled. This initial set was further expanded using near-duplicate detection by the track organizers. In total, 9113 updates were pooled and assessed by TST assessors across all topics. Assessors not only had to determine which updates were relevant but they also had to match the relevant updates with contained nuggets. Thus an update containing more than one nugget contributes more than one unit of gain.

Evaluation Measures

The key evaluation criteria are that the updates should be early (low latency), short (low verbosity) and novel (contain new information than was previously emitted). The latency of reporting the nugget is the difference between the update's timestamp and the nugget's timestamp. The latency of reporting a nugget is penalized by the TST metrics using a discounting function that smoothly reduces unit gain by 80% for delays greater than one day. It is possible for an update to be earlier than the nugget's timestamp, in which case the latency discount function awards the update a bonus for reporting the nugget early. Once an update containing a nugget is emitted (i.e., a nugget is reported), subsequent occurrence of the same nugget in later updates does not contribute to increase in gain.

Additionally, sentences are penalized for being overly verbose. For every sentence a verbosity normalization (Equation 3.2) is computed that determines the extra number of nuggets the sentence could have contained. The lower the number of terms not matching

a nugget, the lower is the verbosity normalization value for a sentence, with a minimum verbosity normalization value of 1. Essentially, the verbosity penalization serves to inflate the number of submitted updates and thus provides a better estimate of precision for the total content returned by a system.

The TST evaluation has 2 primary metrics, Expected Latency Gain (ELG) which measures the gain per update while discounting latency and penalizing verbosity, and Latency Comprehensiveness (LC) which measures the latency discounted coverage of nuggets relating to the topic. ELG and LC are analogous to Precision and Recall respectively. In effect, temporal summarization (TS) systems that produce early and shorter updates, are ranked higher than others on ELG.

The TST track's ELG metric, combines the latency, novelty and verbosity to generate a score that represents the effectiveness of a temporal summarization system. Given a set $\mathcal{D}$ of updates submitted by a system, ELG is defined as,

$$\mathbf{ELG_V}(\mathcal{D}) = \frac{1}{\sum_{d \in \mathcal{D}} \mathbf{V}(d)} \sum_{d \in \mathcal{D}} \mathbf{G}(d, \mathcal{D}) \tag{3.1}$$

where,

$\mathbf{G}(d, \mathcal{D})$ is the latency discounted gain for update d. $G(d, \mathcal{D})$ is non zero when d is earliest update from the set $\mathcal{D}$ to report one or more nuggets for the topic and

$$\mathbf{V}(d) = 1 + \frac{|words\_in\_update\_d| - |words\_matching\_nuggets\_in\_d|}{average\_length\_of\_nuggets} \tag{3.2}$$

is the verbosity normalization of update d.

Thus gain is only experienced when reading the updates that report nuggets earliest. Once a nugget is reported, it does not contribute to gain if it appears again in later updates. If all the updates emitted by a system have a verbosity of 1, then ELG would essentially give us the *gain per update* for the system.

The LC metric replaces the denominator in Equation (3.1), with $\sum_{n \in \mathcal{N}} \mathbf{R}(n)$, where $\mathcal{N}$ is the set of nuggets identified by the assessors and $\mathbf{R}(n)$ is the relevance for n based on its importance ($\mathbf{R}(n) = 1$ for binary relevance).

$$\mathbf{LC}(\mathcal{D}) = \frac{1}{\sum_{n \in \mathcal{N}} \mathbf{R}(n)} \sum_{d \in \mathcal{D}} \mathbf{G}(d, \mathcal{D}) \tag{3.3}$$

Essentially LC computes the recall of relevant information nuggets by the system. The track used binary relevance of nuggets when reporting scores for participating systems.

As proposed by Sakai and Kando (2008), for evaluation scenarios where relevance judgments are incomplete, the non-judged sentences are elided for TST evaluation. Eliding seems appropriate to keep the evaluation fair because the number of updates emitted by participating systems ranges from 107 to 2,815,770; however, the pool formed by taking the top 60 updates (Aslam et al., 2013) from each run for each topic, consists of only 9,113 updates. The eliding however, causes the verbosity normalization in equation (3.1) to be based on the number of judged updates rather than the number of updates submitted. This conflation of eliding and verbosity normalization makes ELG difficult to interpret.

3.1.3 Participating Systems Overview

In all, 7 teams submitted 28 systems (runs) for evaluation at TST 2013 for the SUS task. Two of the runs were in an inconsistent format and thus they were excluded from the evaluation (these runs are excluded from our experiments as well). Runs scoring well on ELG (analogous to Precision), score poorly on LC (analogous to Recall) and vice-versa. Runs with fewer updates overall, scored higher on ELG, and runs with larger number of updates scored higher on LC. Table 3.7 shows the submitted runs ordered by ELG.

The top run, `cluster5`, has an ELG of 0.136 with a total of 197 updates across all topics, submitted over the course of 10 days (i.e. over the query duration). The run at rank 2, `run2`, scored 0.127 on ELG with 844 updates submitted across all topics over 10 days. The top LC scoring run was `rg1` with an LC score of 0.571.

Yang et al. (2013) [Group: BJUT] shortlisted documents using BM25 (Robertson and Zaragoza, 2009). They then performed K-means clustering (K = 50) for the sentences of the shortlisted documents and chose cluster centroids to be part of the summary. Unfortunately their runs were formatted incorrectly.

Xu, Oard, and McNamee (2013) [Group: hltcoe] employed a sentence selection model that was a linear combination of similarity (of query, sentence and documents) and novelty. They also performed query expansion using Wikipedia.

Liu et al. (2013) [Group: ICTNET] shortlisted only those documents whose titles contained query terms for the topics. They trained a "trigger" word (e.g. kill, die, injure) model and selected sentences that contained trigger words from shortlisted documents. Duplicate sentences were removed using the simHash algorithm (Charikar, 2002) and sen-

tences greater than length of 50 words were not included in the summary.

Baruah et al. (2013) [Group: UWMDS] first shortlisted documents from the corpus using the Language Modeling with Dirichlet Smoothing (LMD) retrieval model. Then, 3 different approaches, the `grep` command, BM25 with NLP based query expansion, and passage scoring (Büttcher, Clarke, and Cormack, 2010), were used to score sentences.

McCreadie et al. (2013) [Group: uog] shortlist 10 likely relevant documents per hour in corpus time. They then experimented with various strategies for selecting sentences that included MMR (Carbonell and Goldstein, 1998), query expansion using Wikipedia, and adaptive techniques to decide the number of updates to emit every hour.

Xi et al. (2013) [Group: wim] shortlisted documents that contained all query terms and had lesser than 40 sentences. Sentences were scored for importance and novelty and these scores were further combined to create a confidence score. Sentences with lengths smaller than 10 words or greater than 40 words were discarded.

Zhang et al. (2013) [Group: PRIS] trained a hierarchical-LDA (hLDA) model (Blei et al., 2003) over training documents for query expansion. The authors manually selected keywords from the resultant topics generated by the hLDA model. They scored sentences using the overlap of the keywords with a sentence while discarding sentences greater than 20 words in length.

3.2 Temporal Summarization Track 2014

3.2.1 Corpus, Tasks, Topics

Corpus, Task

TST 2014 prescribed the KBA Stream Corpus 2014[4] (Frank et al., 2014) as the document collection. The 2014 corpus improved upon the KBA Stream Corpus 2013 by, fixing timestamp inconsistencies, improved character encoding conversions as well as performing NLP tagging for all English-like documents. The revised corpus also added more documents spanning a total of 13,663 hours from October 2011 to April 2013. The track organizers also provided a TST specific subset of the corpus where only the documents within topic query durations were supplied.

TST 2014 required participating systems to return temporal summaries for specified topics. The value tracking subtask was not continued in TST 2014 and the participants only submitted runs for the sequential update summarization task.

Topics

TST 2014 had 15 topics (events) in total (Table 3.2). The track expanded the set of event types to {accident, bombing, earthquake, hostage, impact event, protest, riot, shooting, storm}. A major change in the topics was the difference in the query durations across topics. For TST 2013, a systems ELG or LC score could be averaged across topics in order

[4] *TREC KBA Stream Corpus*. http://trec-kba.org/kba-stream-corpus-2014.shtml. 2014.

id	topic	type	query duration
11	Costa Concordia disaster and recovery	accident	18.1
12	Early 2012 European cold wave	storm	27.0
13	2013 Eastern Australia floods	storm	13.0
14	Boston Marathon bombings	bombing	5.2
15	Port Said Stadium riot	riot	10.0
16	2012 Afghanistan Quran burning protests	protest	7.3
17	In Amenas hostage crisis	hostage	4.0
18	2011-13 Russian protests	protest	21.0
19	2012 Romanian protests	protest	14.0
20	2012-13 Egyptian protests	protest	13.0
21	Chelyabinsk meteor	impact event	10.0
22	2013 Bulgarian protests against the Borisov cabinet	protest	11.0
23	2013 Shahbag protests	protest	18.0
24	February 2013 nor'easter	storm	12.0
25	Christopher Dorner shootings and manhunt	shooting	10.3

Table 3.2: Topics at TST 2014 with their types and query durations.

to measure the performance of the system. However, for TST 2014, non-uniform query durations necessitate normalization of performance scores in order to be comparable across topics.

3.2.2 Evaluation Method and Measures

Phase 1: Gold Nugget Extraction, Phase 2: Update Nugget Matching

The identification of topic nuggets and pooling procedures were the same as those for TST 2013.

Evaluation Measures

The evaluation metrics of TST 2014 differs from the evaluation metrics of TST 2013. TST 2014 defines a *normalized* Expected Latency Gain (nELG) (Equation 3.4) and Comprehensiveness (C) (Equation 3.5). The official track metric was the Harmonic Mean $\mathcal{H}$ of normalized Expected Gain and Comprehensiveness. Additionally, TST 2014 utilizes the importance grade of nuggets for evaluation rather than binary relevance as was done for TST 2013. The normalized Expected Latency Gain is formulated as:

$$\mathbf{nELG_V}(\mathcal{D}) = \frac{1}{\sum_{d \in \mathcal{D}} \mathbf{V}(d)} \frac{1}{Z} \sum_{d \in \mathcal{D}} \mathbf{G}(d, \mathcal{D}) \tag{3.4}$$

where, Z is the maximum obtainable expected gain. The Comprehensiveness is formulated as:

$$\mathbf{C}(\mathcal{D}) = \frac{1}{\sum_{n \in \mathcal{N}} \mathbf{R}(n)} \sum_{d \in \mathcal{D}} \mathbf{G}(d, \mathcal{D}) \tag{3.5}$$

where, $\mathcal{N}$ is the set of nuggets identified by the assessors and $\mathbf{R}(n)$ is the relevance for n based on its importance.

A major change in the evaluation process is that unjudged updates are not elided for TST 2014. All submitted updates that are not judged by assessors are considered to be non-relevant. However, a key aspect of not eliding is its effect on the estimation of verbosity of unpooled updates, which requires knowing the length of each submitted update. In total 1,353,971 updates were submitted to TST 2014 across all runs (6,224,775 updates were submitted to TST 2013). Extracting lengths for over a million sentences from a corpus containing over a billion documents can be a cumbersome exercise. As such, the verbosity

for unjudged updates was estimated to be $1 + 1/(avg.nugget.length)$ for each unjudged update in the track's evaluation script[5].

For TST 2014 evaluation, the organizers identified exact duplicates of pooled updates from the all submitted runs and release qrels extended with duplicates of judged updates. Baruah, Roegiest, and Smucker (2014) first identified that a there are a large number of exact duplicates in the corpus and suggested that they be included for evaluation. The evaluation at TST 2013 did not consider a duplicate of a pooled update return by a system as having the same relevance judgement. However, it was found that 13 out of 26 submitted systems experienced a statistically significant change in performance scores when duplicates were included into the judged pool. We describe the effect of expanding the qrels with duplicates in Chapter 4.

3.2.3 Participating Systems Overview

In all, 6 teams submitted 24 systems for evaluation at TST 2014. The top run, `2APSal`, has an $\mathcal{H}$ of 0.1162 with a total of 381.4 updates submitted per topic on average. The track also tried out an Expected Latency metric that measured the average lateness of returned updates. It was found that Expected Latency does not correlate well with the official measure $\mathcal{H}$, and systems, even if they deliver updates late, they may perform well on the Harmonic mean of nELG and C.

Zhao et al. (2014) [Group: BJUT] indexed documents using the Lemur Toolkit[6] and

[5] *Evaluation script for Temporal Summarization 2014*. http://trec.nist.gov/data/tempsumm/2014/tseval.py. 2014.

[6] *Lemur Project*. http://www.lemurproject.org/.

retrieved sentences using BM25 with query expansion. They refined the retrieved set of sentences using a combination of text similarity and clustering, finally choosing the cluster centroids for output. Their runs performed well in the track, especially for nELG.

Chen et al. (2014) [Group: ICTNET] shortlisted only those documents whose titles contained all query terms from the topic's query string. They utilized unsupervised (Latent Dirichlet Allocation (Blei, Ng, and Jordan, 2003)), as well as supervised (Support Vector Machines (Cortes and Vapnik, 1995)) learning methods to select terms for query expansion. The expanded queries, along with their respective method derived weights, were used to score sentences for output.

McCreadie et al. (2014) [Group: uog] develop a real-time filtering method, in contrast to their TST 2013 method, where they emitted ranked updates every hour. The documents are first classified as being on/off topic using a classifier trained on TST 2013 topics. Then sentences from on-topic documents are classified as being useful or not using a supervised classifier that checks for length of sentences, boilerplate sentences, emergency related content, presence of named entities, and other related features. Finally, sentences are checked for novelty using cosine similarity with previously emitted sentences. The group also tried to alleviate the vocabulary mismatch problem between topic query strings and the content of the nuggets by selecting sentences in close proximity to the identified likely relevant sentence.

Qi et al. (2014) [Group: BUPT_PRIS] shortlisted documents within the query duration that matched the query strings. They tried 3 different query expansion methods based on Wordnet (Oram, 2001), Word2Vec (Mikolov et al., 2013) and neural networks, after which

sentences were scored using a combination of keyword overlap and latency discounting.

Abbes et al. (2014) [Group: IRIT] first construct an "event model" for terms that might occur in likely relevant documents given the type of an event. The event model is constructed using the topic nuggets of TST 2013. They then filter documents from the KBA Stream corpus and note the top scoring documents for each hour. Finally, they select sentences that contain less that 25 words, are novel, and match terms from the event model and the query string.

3.3 Participation at TST 2013

A team of 3 students participated at the TREC 2013 temporal summarization track from the University of Waterloo (Baruah et al., 2013). The team decided on a staged approach for the retrieval of sentences. They first shortlisted likely relevant documents using query-likelihood with Dirichlet smoothing (Equation 3.6). Thereafter, each team member utilized a different approach to retrieve sentences. Our specific approach used a passage retrieval technique (Equation 3.7) and we also experimented with query expansion for constructing our runs. Our runs did not perform well on ELG, although they performed well on the LC metric.

The track organizers provided a single training topic, "iran earthquake" (Figure 3.2), which participants used to develop their systems, as did we.

3.3.1 Corpus Preprocessing

The KBA stream corpus contains over a billion documents. Each document has its own timestamp. As per the constraints of the temporal summarization task simulation, retrieval algorithms were not allowed to use term statistics from the "future", i.e., any retrieval methods could only use the set of documents containing a timestamp smaller than or equal to the timestamp of the current document under processing. This constraint ruled out creating single monolithic index for the whole corpus for temporal summarization.

Running Index

Instead, for each of the 11,948 hours spanning the corpus, we computed term statistics for all the documents that had timestamps within the hour. This resulted in 11,948 files (`hour.counts`), each containing the term counts for the vocabulary in that hour. Then, given query terms, to get the term statistics to score a particular document d at its time stamp, one simply needs to cumulatively sum the term counts in the `hour.counts` files, starting from the first hour h_0 to the hour h_{d-1}, where h_d is the hour containing document d (Algorithm 1). Thus our term counts could be off by the term counts accumulated in at most 59 minutes 59 seconds for our h_d. Although, given the large number of hours between the corpus start time and each event's start time, the effect on document scores is minimal.

Given a set of terms T, Algorithm 1 returns the total number of occurrences of each term in T, along with the number of occurrences of all terms in the vocabulary (l_M), within a specified time interval. Algorithm 1 cumulatively adds term counts by going over the `hour.counts` files that lie within the specified time interval. With the Cumulative Counts

Algorithm 1 Compute term counts cumulatively between the time interval $[h_{start}, h_{end}]$. T is the set of terms for which term counts are needed, $\mathcal{V}$ is the vocabulary, l_M is the total number of term occurrences in the interval $[h_{start}, h_{end}]$

CumulativeCounts(T, h_{start}, h_{end})

if $h_{start} > h_{end}$ **then**
  for all $t \in T$ **do**
    $t.count = 0$
  end for
  return T, 1
end if
$l_M = 0$
$h = h_{start}$
while $h < h_{end}$ **do**
  $hour.counts \leftarrow$ read in `hour.counts` file for h
  for all $t \in T$ **do**
    $t.count = t.count + hour.counts(t)$
  end for
  for all $w \in \mathcal{V}$ **do**
    $l_M = l_M + hour.counts(w)$
  end for
end while
return T, l_M

algorithm in place, we can compute the unigram probabilities of term occurrences within a specified time period $[h_{start}, h_{end}]$, by dividing term counts for each term in T by l_M (Algorithm 2).

Algorithm 2 Compute the probabilities of term occurrences in the interval $[h_{start}, h_{end}]$. T is the set of terms for which probabilities are needed

IntervalUnigramProbabilities(T, h_{start}, h_{end})

Require: $h_{start} \leq h_{end}$
  $P(T) \leftarrow 0$
  $T.counts$, $l_M \leftarrow$ CumulativeCounts(T, h_{start}, h_{end})
  for all $t \in T.counts$ **do**
    $P(t) \leftarrow t.count/l_M$
  end for
  return $P(T)$

3.3.2 Shortlisting Documents

We shortlisted documents using language modeling with Dirichlet smoothing ranking function as outlined by Büttcher, Clarke, and Cormack (2010),

$$score_{LMD} = \sum_{t \in q} q_t \cdot log\left(1 + \frac{f_{t,d}}{\mu} \cdot \frac{l_C}{l_t}\right) - n \cdot \left(1 + \frac{l_d}{\mu}\right) \tag{3.6}$$

where, given a query q of length n, q_t is the frequency of term t in q, n is the total number of terms in the query, $f_{t,d}$ is the frequency of term t in the document d, l_t is the number of times term t appears in the corpus , l_C is the total number of times *any* term appears in the corpus (sum of lengths of all postings lists), l_d is length of the document and μ is the Dirichlet smoothing parameter. For our experiments we set μ to 2500.

For a given query we scored all documents within the query duration using Equation 3.6. We shortlisted all documents with $score_{LMD} > 0$; these documents would contain reasonably important query terms while being of reasonable length. The threshold is rather arbitrarily set to balance the likely relevance with length of the document. We chose a fixed threshold to shortlist documents for each hour (instead of, for instance, the top-K documents per hour) because $score_{LMD}$ may vary hour-on-hour depending on the documents within the hour, thereby causing the threshold to fluctuate. Using a threshold based on a fixed document similarity score is useful instead of a fixed selection of top-K documents because it allows us to process more (than K) likely relevant documents (if they do exist) within each hour.

Additionally, shortlisting a fixed number of documents per hour (e.g. top-K as per $score_{LMD}$) would require waiting till the end of the hour to emit documents causing each of the top-K documents to have the same timestamp (the end of the hour). Choosing a fixed score threshold allows us to emit all shortlisted documents as soon as they are scored, without any delays. Estimating the adequate threshold value would need training but since TST 2013 was the pilot run of the track, we did not have past data to work with. Furthermore, different thresholds may be required for different topics. Thus we chose to set a threshold of $score_{LMD} = 0$, as a generic filter for likely relevant documents of reasonable length.

3.3.3 Selecting Sentences

While documents were shortlisted using Equation 3.6, sentences were shortlisted using 3 different strategies by the 3 team members. We used techniques based on regular expressions, BM25 with query expansion and Passage Scoring (Baruah et al., 2013).

Specifically, our method scored sentences using a passage retrieval algorithm (Clarke, Cormack, and Lynam, 2001) as described in Büttcher, Clarke, and Cormack (2010),

$$score_{cover} = \sum_{t \in q'} (log(l_C/l_t)) - m \cdot log(l) \tag{3.7}$$

where, q' is a subset of the query term set, l is the length of the passage, and m is $|q'|$. We considered each sentence as a passage for Equation 3.7. The subset q' establishes a "cover" of query terms in a given sentence. The $score_{cover}$ is higher for covers that contain important terms as per the *term weight* $log(l_C/l_t)$, especially when the length of the sentence is small. We threshold $score_{cover}$ in a similar manner to $score_{LMD}$ in that we do not emit sentences that have a $score_{cover} \leq 0$, i.e. we want a likely relevant sentence that is not overly long.

3.3.4 Query Expansion

Since the period of interest spans 10 days for each event, the topical discourse in the document/sentence content may shift with the passage of time. The track provides the type of the event and also highlights specific attributes of interest for each topic ({ injuries, locations, deaths, displaced, financial impact}). Each of these attributes may have similar

or related terms appearing at various times in the discourse. These attributes provide leeway for directed diversification of retrieved sentences, going beyond simply using the query string for retrieval.

We first generated combinations of "seed" queries (Table 3.3). We selected expansion terms from the top 20 documents for each seed query using the formula

$$n_{t,k} \cdot w_t \tag{3.8}$$

where, $n_{t,k}$ is the number of times term t appears in the top-k documents, and w_t is the inverse collection frequency ($log(l_C/l_t)$) of term t. Note that l_t is the number of occurrences of term t from the start hour of the corpus till the current hour that contains the top-20 documents. The expansion terms thus change with every passing hour since the length of postings list for term t may not have a uniform rate of growth for every hour.

We observed the expansion terms generated by the different types of queries for successive hours. We found that seed queries of type SSA and GSA did not produce good expansion terms; most expansions were dominated by the type of the event (for GSA) or the query string (SSA). On the other hand, the GAA seed query provided changing expansion terms for successive hours (Tables 3.4, 3.5). The SAA query type adds the query string to the GAA query and thus produces expansion terms more related to the topic. Though the expansion terms generated by SAA query type seemed related to the topic in general, they did not seem diverse enough to capture more details about the event. For instance, the SAA expansion terms do not show much variety within 3 hours (Table 3.4 and Table 3.5), whereas the GAA expansion terms relate to technical information about the

query type	seed query
Generic All Attributes (GAA)	earthquake injuries locations deaths displaced financial impact
Generic Single Attributes (GSA)	earthquake injuries earthquake locations earthquake deaths earthquake displaced earthquake financial impact
Specific All Attributes (SAA)	iran earthquake injuries locations deaths displaced financial impact
Specific Single Attributes (SSA)	iran earthquake injuries iran earthquake locations iran earthquake deaths iran earthquake displaced iran earthquake financial impact

Table 3.3: Choice of seed queries to generate expansion terms. Specific queries include the topic query string.

earthquake e.g., ANSS[7], RMSS[8], shakemap[9]. However the SAA seed query does generate names of affected towns (Haris, Ahar, Varzqan) early on (Table 3.4)

Fusing lists of expansion terms

In order to include expansion terms from both the SAA and GAA seed queries, we first ranked the respective expansion term lists, as per Equation 3.8, and then generated a combined list using Reciprocal Rank Fusion (RRF) (Cormack, Clarke, and Büettcher, 2009),

$$rrf(t) = \sum_{i}^{L} \frac{1}{k + r_i(t)} \tag{3.9}$$

[7]ANSS: Advanced National Seismic System http://earthquake.usgs.gov/monitoring/anss/

[8]"RMSS: root-mean-square travel time residual in seconds" - source http://earthquake.usgs.gov/earthquakes/glossary.php

[9]Shakemap: A map that presents information on the shaking of ground rather than epicenter and magnitude. http://earthquake.usgs.gov/research/shakemap/

Top 10 expansion terms for training query type		
GAA	**SAA**	**RRF-fused**
earthquake	injured	earthquake
quake	earthquake	injured
magnitude	magnitude	magnitude
injured	haris	quake
hundreds	varzaqan	killed
killed	ahar	hundreds
seismic	quake	haris
iran	killed	ahar
earthquakes	hundreds	varzaqan
northwestern	iran	iran

Table 3.4: **Hour 2012-08-11-18**: Expansion terms for the training topic "iran earthquake", generated using seed queries of type Generic All Attributes (GAA) and Specific All Attributes (SAA).

Top 10 expansion terms for training query type		
GAA	**SAA**	**RRF-fused**
earthquake	earthquake	earthquake
magnitude	iran	seismogram
seismogram	villages	anss
anss	magnitude	nsmp
nsmp	least	rmss
rmss	northwestern	magnitude
recenteqsww	earthquakes	recenteqsww
crustal	tv	crustal
shakemap	injured	shakemap
seismologist	news	seismologist

Table 3.5: **Hour 2012-08-11-21**: Expansion terms for the training topic "iran earthquake", generated using seed queries of type Generic All Attributes (GAA) and Specific All Attributes (SAA).

where, L is the number of lists to fuse, $r_i(t)$ is the rank of the term t in list i, and $rrf(t)$ is the fusion score for the term t. Büttcher, Clarke, and Cormack (2010) prescribe value of k to be 60, which we use for our experiments.

A higher fusion score indicates that the term is highly ranked in both, SAA and GAA, term expansion lists. We tried to incorporate this information by slightly modifying Equation 3.7 to

$$score_{cover} = \sum_{t \in q'} \left(log \left(\frac{l_C}{l_t} \cdot rrf(t) \right) \right) - m \cdot log(l) \tag{3.10}$$

where, q' now represents a subset of the *expanded* query. Note that multiplying the l_C/l_t by $rrf(t)$ drives down the importance of the term t while scoring. Thus, Equation 3.10 ensures that not only the general importance of the term is high but also the fusion score of the term is high. The only exceptions were the original query terms, for which, the $rrf(t)$ was fixed at $1/(k+1)$. Note that the length requirements for the selection of the sentence became more stringent due to this modification.

Thus, given $score_{cover}$ and the fused query term expansion list, our method had 2 parameters; r the number of query + expansion terms in total, and c the smallest allowable cover of query term subsets in a sentence. For instance, if $r = 25$ and $c = 4$, then a sentence would need to contain at least 4 query terms out of 25, to be even considered for retrieval.

3.3.5 Constructing Runs

There were 3 main hurdles for run construction. The first problem was that expansion terms resulted in non-relevant sentences scoring high with $score_{cover}$. We reasoned that

since relevant terms could be spread across likely relevant documents, we could weight the $score_{cover}$ for each sentence with the $score_{LMD}$ of its source document. The weighted $score_{cover} \times score_{LMD}$ would help raise the sentences from relevant documents higher up in the returned list.

The second problem was that term statistics for expansion terms were hard to compute efficiently because of (i) our running hour-on-hour "index" and (ii) the fact that there were different expansion terms generated for each hour (of the 240 hours) in the query duration. Getting accurate term statistics for each new term would mean running *CumulativeCounts* (Algorithm 1) for single terms rather than as a set. To alleviate this problem, we generated a minimal background model, i.e. rather than computing query (and expansion) term statistics from the start of the collection, we computed computed term statistics from a much smaller *Background* Language model starting 2 days (20% of query duration) prior to the query start time. We updated the *Background* model with the query and expansion term counts from each hour that was processed for the query. Thus the background model "grew" with time.

In hindsight this was a bad idea. Post-hoc analysis revealed that our chief estimator of term importance, $log(l_C/l_t)$, where both l_C and l_t are limited to the Background model, does not stabilize until documents in 1000 hours were processed. Figure 3.3 shows the growth of $log(l_C/l_t)$ with the passage of time in the corpus. This growth pattern is typical of all topic query strings. The figure shows that initially the term weights fluctuate as they are computed over the documents within the first few hundred hours of the corpus. The term weights stabilize after about 1000 hours. Then, as the as the sum total of the occurrences of all vocabulary terms (l_C) increases, if a term occurs with a higher

frequency in general (e.g. "hurricane"), its weight decreases after a time. Interestingly, during the query duration, it can clearly be seen that the term weights for both "sandy" and "hurricane", drop considerably. This is because they occur at a much higher rate within this interval.

Needless to say, a Background language model spanning 48 hours was highly inadequate in order to compute term weights for query and expansion terms alike. This is a major factor that may have hurt the performance of our runs.

The third hurdle was that we had to deal with the large number of duplicate sentences in the corpus. We returned the earliest unique sentence identified by our system for each run.

Our runs were constructed using the following procedure; Given an event with query Q with a period of interest $[q_{start}, q_{end}]$, first generate a background term statistics model (B) using IntervalUnigramProbabilities(Q, $q_{start} - 48$, q_{start}). Then for each hour from q_{start} to q_{end},

1. Retrieve top-20 documents (D_h) for the current hour h.
2. Generate expansion terms (E_h) for hour h using D_h (as described in Section 3.3.4).
3. Select expansion terms to identify likely relevant sentences from hour $h + 1$ of the query duration.
4. Remove duplicates.
5. Update background model B with term counts of Q and E_h, executing algorithm

c (Cover)	**r** (#query terms)	**unique sentences**
2	**25**	**7425**
2	50	7699
2	100	5495
4	25	6111
4	**50**	**7073**
4	100	5446
10	25	3488
10	50	5047
10	100	5700

Table 3.6: Number of unique updates obtained for values of parameters c and k for the training topic. Rows in bold represent submitted runs.

IntervalUnigramProbabilities(T, $q_{start} - 48$, h) if necessary for new terms T in $\{Q \cup E_h\}$.

For the first hour, since there are no expansion terms, we returned likely relevant sentences using the unexpanded query Q only.

We varied the cover of query terms c and the number of expansion terms r as shown in Table 3.6 and observed the number of unique sentences returned for the training topic (Figure 3.2). Our submissions were essentially recall oriented approaches as we were trying to get as many information nuggets about the given attributes of the topics as possible. We therefore selected the runs having parameters ($c = 2$, $r = 25$) and ($c = 4$, $r = 50$). These parameter values gave us a large number of unique sentences as well as an opportunity to check for differences in performance for a reasonable number of expansion terms.

3.3.6 Results

Admittedly, the design of our experiments for TREC 2013 temporal summarization needed a more refined approach. Rather than change one factor at a time and evaluate iteratively, many factors were changed, and we did not submit an adequate number of runs. Table 3.7 lists the results of the track. Our runs did achieve a high recall (LC) but performed poorly on ELG (precision).

Of note, one of our runs, `UWMDSqlec2t25`, found the highest number of nuggets (38) per topic, on average. The runs `rg1` and `rg2` also find close to 38 nuggets per topic. These runs utilized distributional similarity for term expansion (Vechtomova, 2012), however, the seed words for expansion were chosen *manually* (Baruah et al., 2013). In contrast, our query term expansion method is completely automatic. Even though our runs return a higher number of nuggets, the Latency Comprehensiveness is lower because not all nuggets were delivered early enough.

For the ELG evaluation measure as formulated in Equation 3.1, returning a large number of updates would significantly impact the score on ELG. It is not a surprise that runs with fewer overall updates score higher on ELG because the verbosity penalization is not as heavy if the number of updates is small.

3.3.7 Conclusion

Our participation at the TREC 2013 temporal summarization track gave us an opportunity to work on the interesting problem of information retrieval over a document stream. The

RunID	ELG	LC	# updates returned	average # updates per topic	# Judged Updates per topic	# found nuggets per topic
cluster5	0.136	0.126	197	21.9	21.8	6.89
run2	0.127	0.251	844	93.8	55.3	16.89
run1	0.125	0.253	880	97.8	56.9	17.11
TuneExternal2	0.118	0.203	7,195	799.4	49.7	12.89
TuneBasePred2	0.114	0.244	24,265	2,696.1	74.8	15.11
cluster3	0.103	0.176	381	42.3	38.2	9.67
cluster2	0.074	0.260	1,099	122.1	89.0	16.00
uogTrNMTm1MM3	0.069	0.216	3,229	358.8	89.3	15.67
cluster1	0.067	0.288	1,483	164.8	100.9	17.00
cluster4	0.067	0.292	1,467	163.0	100.7	17.00
BasePred	0.067	0.368	79,116	8,790.7	167.7	22.00
Baseline	0.063	0.381	114,687	12,743.0	183.9	24.44
uogTrNSQ1	0.060	0.184	1,251	139.0	80.7	13.11
EXTERNAL	0.055	0.413	202,285	22,476.1	184.4	26.00
uogTrNMTm3FMM4	0.049	0.170	1,515	168.3	96.4	14.22
uogTrNMM	0.045	0.254	8,592	954.7	140.0	19.11
uogTrEMMQ2	0.040	0.259	18,700	2,077.8	147.6	19.00
SUS1	0.036	0.128	21,048	2,338.7	73.1	8.78
rg4	0.028	0.516	376,770	41,863.3	281.6	31.56
rg3	0.026	0.506	382,807	42,534.1	275.0	31.89
rg2	0.022	0.562	2,696,036	299,559.6	402.8	37.78
rg1	0.021	0.571	2,815,770	312,863.3	414.1	37.89
UWMDSqlec4t50	0.018	0.530	1,923,621	213,735.7	357.2	36.22
UWMDSqlec2t25	0.017	0.537	2,070,504	230,056.0	370.4	*38.00*
CosineEgrep	0.010	0.018	107	11.9	11.9	0.89
NormEgrep	0.001	0.061	1,362	151.3	65.3	3.56

Table 3.7: TST 13 Participation results for our runs. `UWMDSqlec2t25` found the most number of nuggets per topic on average. Both runs scored high on Latency Comprehensiveness. Note that the average number of nuggets per topic is 119.67 (Table 5.3).

algorithms not only needed to return likely relevant sentences but were also required to do so in a timely manner. Poor results over ELG notwithstanding, analysis post track participation led to interesting research projects:

1. The evaluation pool for TST 2013 was constructed taking the top sentences as per system assigned confidence scores for each returned sentence. We observed a large number of duplicate sentences in the corpus. What might happen if a system had returned the duplicate of a pooled (and then judged) relevant sentence? We answer this question in Chapter 4.

2. Some submitted runs returned a very large number of updates whereas some returned very few. The specified period of interest for each topic was 10 days. The top ELG run on average delivered only 22 updates per topic over a 10 day period returning less than 7 nuggets on average (Table 3.7), whereas the average number of nuggets per topic is 119. A typical user who is interested in the event for 10 days, may find 2 updates per day insufficient information, especially if the event impacts the user in a personal capacity. On the other hand returning thousands of updates may inundate the user. How can we effectively evaluate systems that filter event related updates from a web-scale document stream, given various types of users that consume the filtered updates? This question forms the core of this thesis and Chapter 5 presents a detailed exposition of our research in developing a user-oriented evaluation measure for update filtering systems.

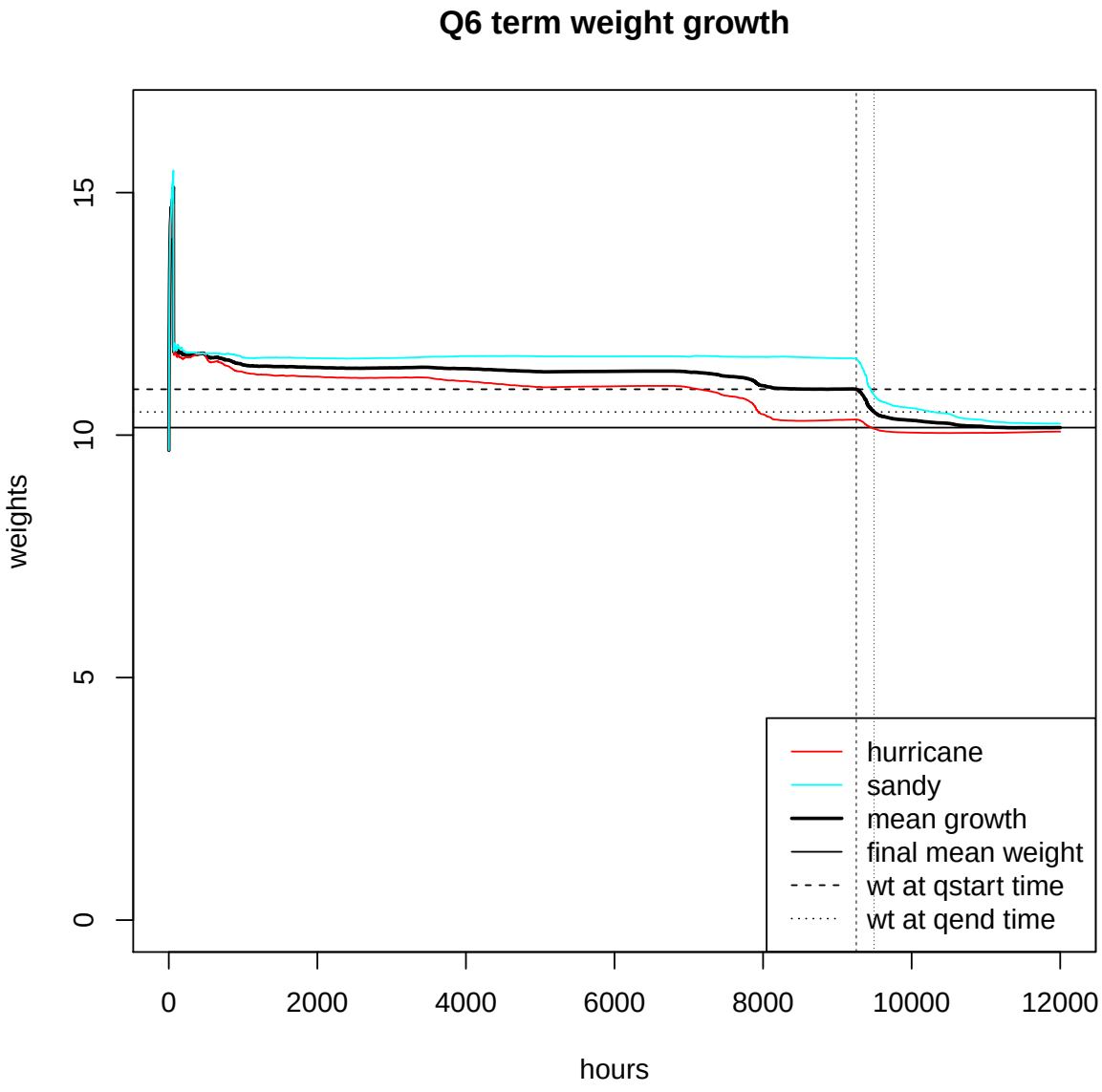


Figure 3.3: Growth in term weight ($log(l_C/l_t)$) of query terms for topic "Hurricane Sandy". The X-axis indicates hours in corpus time. The Y-axis shows the weight values. The vertical dashed and dotted lines indicate the start and end of the topic query duration respectively. The horizontal lines show the mean term weight for the query string at the start and end of the query duration, and the final mean term weight for the collection.

Chapter 4

Evaluation in the Presence of Duplicates

We observed a large number of duplicate sentences to be present in the KBA Stream corpus. We found that the number of duplicates in the corpus could be as large as 1000x their presence in the pool for TST 2013. In this chapter we investigate the effect of expanding the evaluation pool by adding *exact* duplicates of the judged sentences from the corpus.

The assessors pooled sentences from the participating systems to create an evaluation pool. The sentences that were judged for relevancy form the judged set of sentences for the track. The runs, however, were only required to submit the sentence *identifier*. Thus, if a sentence d' that is not in the pool is an exact duplicate of the sentence d which is in the pool, then, a system returning d' may not be fairly evaluated (if $d == d'$ is relevant or

otherwise), especially when d' has an earlier timestamp than d.

Indeed, the track organizers found exact duplicates within the pool itself (Table 4.1; #known duplicates). To account for the duplicates in the pool, the identifier of each duplicate sentence was mapped to the identifier of a designated original sentence. This mapping is preserved while computing scores for system evaluation. However this within-pool mapping does not map the original sentence with all its duplicates from the corpus. The aim of our experiment was to (i) expand the within-pool mapping of duplicates to original sentence to include all duplicates from the corpus, and (ii) observe the change in evaluation scores when using the duplicates-expanded qrels (judged set of sentences).

Our hypothesis was that, given the large number of duplicates, the evaluation scores for systems would change considerably. Our experiments (Baruah, Roegiest, and Smucker, 2014) on expanding the judged set of sentences of TST 2013 show that the relative ranking of systems (runs) is largely unaffected with the addition of duplicates. However, the absolute system score changes significantly for 13 out of 26 systems on ELG and for 12 out of 26 systems for LC respectively.

We conducted a similar experiment on the judged set of sentences of TST 2014 and we found no significant differences in relative performance of systems. Also, the effect of expanding the judged set of sentences with duplicates, is minimal for the absolute scores of the systems, with only 3 runs showing statistically significant changes in score for the Harmonic Mean measure of the TST 2014 (Section 3.2). The fewer changes in absolute scores of runs can be attributed to the differences in the evaluation process between TST 2013 and TST 2014 (Section 4.3.2).

4.1 Expanding the Judged Set of Sentences

We first identified duplicates of all pooled sentences from the corpus and included them into the pool by expanding the within-pool duplicates to original mapping. Table 4.1 shows the change in the pool on its expansion using duplicates. The known-duplicates column lists the total number of duplicates found within the pool for a given topic. The with-duplicates-in-corpus column lists the number of sentences for which duplicates were found in the corpus. For instance, for topic 1, 100 of the 779 pooled sentences are duplicates which were found during the pooling process, and duplicates were found for 309 of 779 sentences from within the corpus. Similarly for the relevant-with-duplicates-in-corpus column; it lists the number of relevant sentences for which duplicates were found in the corpus.

The expanded-judged-set columns shows the total number of duplicates found in the corpus for the pooled sentences of each topic. It can be seen that the judged set of sentences expands by almost 1000 times (to 9,034,179 from the original 9,113) sentences.

Topic	Original Judged Set of Sentences					Expanded Judged Set	
	#sentences	#known duplicates	#with duplicates found in corpus	#relevant sentences	#relevant with dup.s in corpus	#sentences	#relevant sentences
1	779	100	309	431	146	833794	1445
2	912	180	474	381	202	2241589	6301
3	762	112	494	211	154	552145	25199
4	1463	276	946	410	260	264474	22587
5	1069	0	689	82	63	821897	17043
6	1517	187	905	493	270	730296	18661
8	1128	205	609	172	102	1057643	1741
9	873	172	423	168	97	2430455	2384
10	610	89	338	287	143	101886	1895
Total	9113	1321	5187	2635	1437	9034179	97256

Table 4.1: The number of sentences in the Original judged set vs. the Expanded set.

Table 4.2 lists the most frequently occurring duplicates for the topics of TST 2013. The 3 most frequently occurring duplicates constitute 67% of the total number of duplicates found. These sentences appear to be headers or footers (boiler plate sentences) of web pages or news articles. The most frequently occurring relevant sentence was found to occur 5,403 times. This relevant sentence, "National Hurricane Center in Miami said Isaac became a Category 1 hurricane Tuesday with winds of 75 mph.", describes "Hurricane Isaac" the event of topic 6.

It is likely that the large number of duplicates found are due to news/web syndication services. The KBA stream corpus contains documents sourced from various news wires, and these news articles are the second largest subset of documents in the corpus, with documents from social media/blogs/forums forming the largest constituent set. After a news event, many articles might be expected to be written about it and these would be expected to be distributed via syndication all over the web.

4.2 Evaluating TST 2013 Runs using qrels Expanded with Duplicates

After the duplicates were identified from within the corpus, we included them into the judged set of sentences while maintaining the mapping procedure put in place in the track qrels, whereby, every duplicate sentence d' is mapped to its corresponding prototype d. With the expanded qrels, we re-evaluated ELG and LC for each submitted run to TST 2013.

Frequency	Topics	Duplicate Sentence
3376809	2,9,1	All rights reserved./All rights reserved
2013684	2,9,8	Yahoo!
673876	3	New User ?
529085	5	3.
294662	8	This material may not be published, broadcast, rewritten or redistributed.
. . .		
166557	6,9	U.S.
111503	8,9	Register Sign In Help New Firefox ®16 Optimised for Yahoo! Notifications Help Mail My Y!
. . .		
81985	6	Join Here .

Table 4.2: Examples of Duplicate Sentences with high number of occurrences across all topics for TST 2013.

Figure 4.1 plots the ELG scores of runs, for the expanded vs. the original set of judged sentences. The rank correlation between the 2 system rankings is quite high with a Kendall's $\tau = 0.899$. However, the ELG scores for 13 runs are significantly different (on a paired t-test with p-value ≤ 0.05).

Figure 4.2 plots the LC scores of runs, for the expanded vs. the original set of judged sentences. The rank correlation between the 2 system rankings is also high with a Kendall's $\tau = 0.942$. LC scores for 12 runs are significantly different (on a paired t-test with p-value ≤ 0.05).

Almost half the runs do not show any change in absolute score. This is primarily because very few or no duplicate sentences were found for their contributions to the pool. Additionally, these runs had fewer submitted sentences and were well represented in the pool, whereas the larger runs (with more numbers of sentences) showed changes in scores

due to the addition of duplicates in the qrels. We observe that, in general, there is a decrease in the ELG score. This is because the inclusion of duplicates increases the verbosity normalization component of the ELG evaluation measure. Recall that the sentences not in the pool were elided for evaluation (Section 3.1.2) and with the addition of duplicates, the verbosity of any system that returns duplicates would increase. Only one run benefited on ELG with the addition of duplicates, i.e., at least for one run, relevant and unjudged duplicates were found from the corpus, thereby causing an increase in the ELG score, although the increase in ELG is not statistically significant (Figure 4.1).

In contrast, a majority of the runs show an increase in LC. This is because the duplicates in the run may have an earlier timestamp than the pooled and judged sentence. An earlier timestamp for a duplicate would cause the LC to increase since the latency penalization is lesser for early delivery of relevant material and in some cases the latency penalization also awards a bonus (Section 3.1.2), if a sentence reports a nugget earlier than the nugget's timestamp. A nugget's timestamp is extracted from the respective topic's Wikipedia article (Section 3.1.2).

It is interesting to draw attention to the two outliers at ranks 4 and 5 on the original judged set ELG scale in Figure 4.1. These runs seem to show a relatively larger decrease in ELG when the expanded set of judgements is used. The points correspond to runs `TuneExternal2` and `TuneBasePred2` respectively. Table 3.7 shows that these runs submitted 7,195 and 24,265 updates respectively. Aslam et al. (2013) report that these two runs are revised (or fixed) runs. The `hltcoe` group submitted an initial version of these runs; this version contributed updates to the pool; the group later submitted a revised version of these runs after the evaluation pool was created. The updates from the revised runs

were not included into the evaluation pool. Although these revised runs did well on ELG using the original pool, it appears that the runs contain many duplicates of the judged set, causing a larger verbosity normalization to drive down the ELG score when using the expanded set of judgements. In other words, eliding of unjudged updates helped the revised runs since there were fewer common updates between the revised runs and the original evaluation pool; however, when duplicates were considered, the impact of verbosity normalization was harsher. Nevertheless, as the figure shows, the ranking of the runs changes only by one position when the expanded set of judgements is used.

4.2.1 Effect of Variations in Duplicate Detection

Our experiments so far identified exact duplicates. We also tried variations based on simple transformations like *lowercase*-ing, *whitespace*-collapsing (reducing consecutive whitespaces to single whitespace) and *white-lower*, a combination of lowercase and whitespace collapsing. The lowercase transformation is a common normalization technique used while indexing documents. The aim was to detect duplicates while ensuring minimal information loss.

The lowercase-ing identified 10,872,223 duplicates, however only 44 new relevant duplicates were found. The whitespace transformations did not produce new duplicates. The Kendall's τ between the lowercase-transformed-duplicates expanded set induced system rankings and the track's systems rankings is listed in Table 4.3. It is the same as the Kendall's τ between the exact-duplicates expanded set induced rankings and the track's rankings.

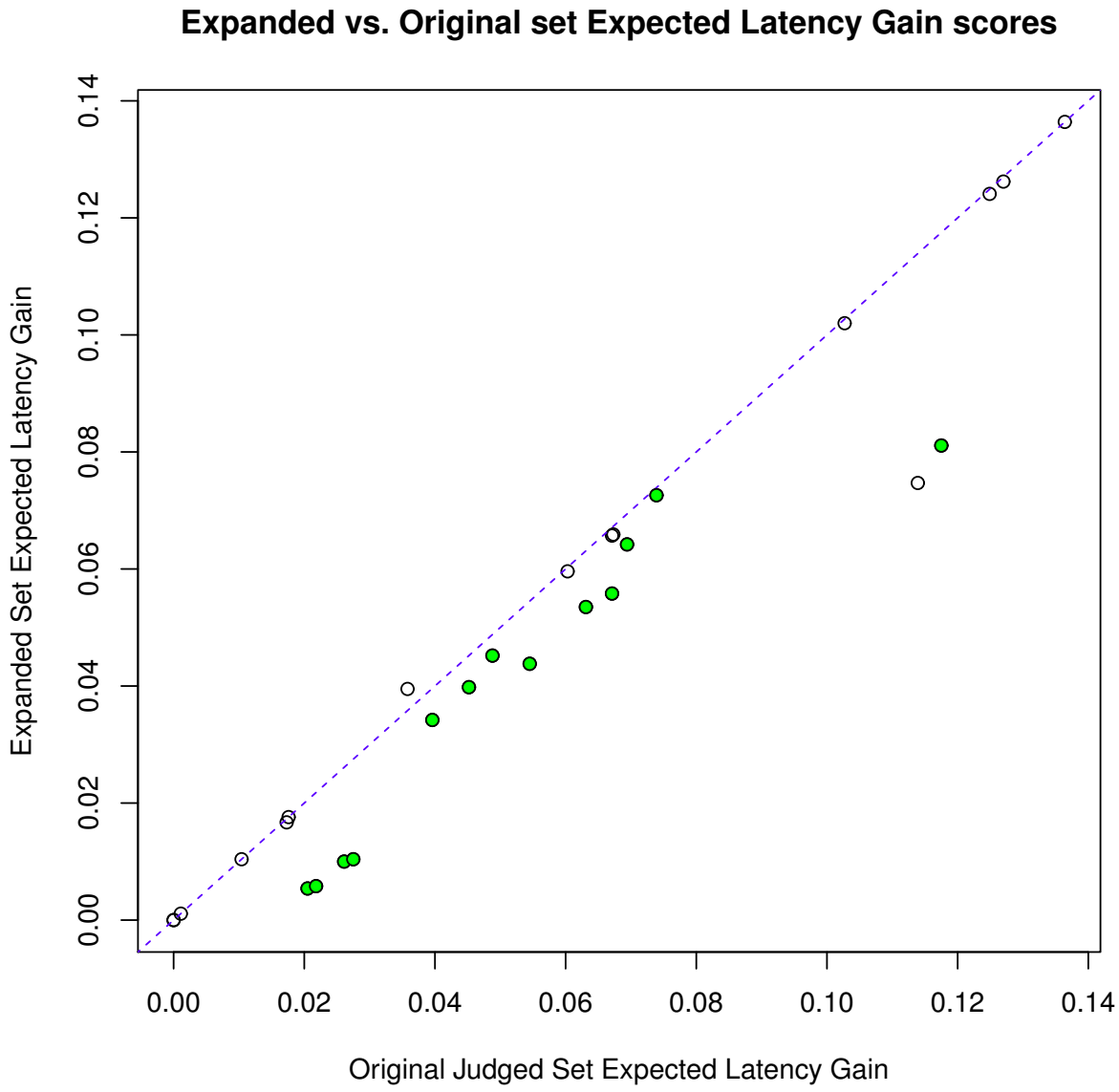


Figure 4.1: ELG scores for the systems using the duplicates-expanded set of judged sentences vs. the ELG scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute ELG score for the respective run.

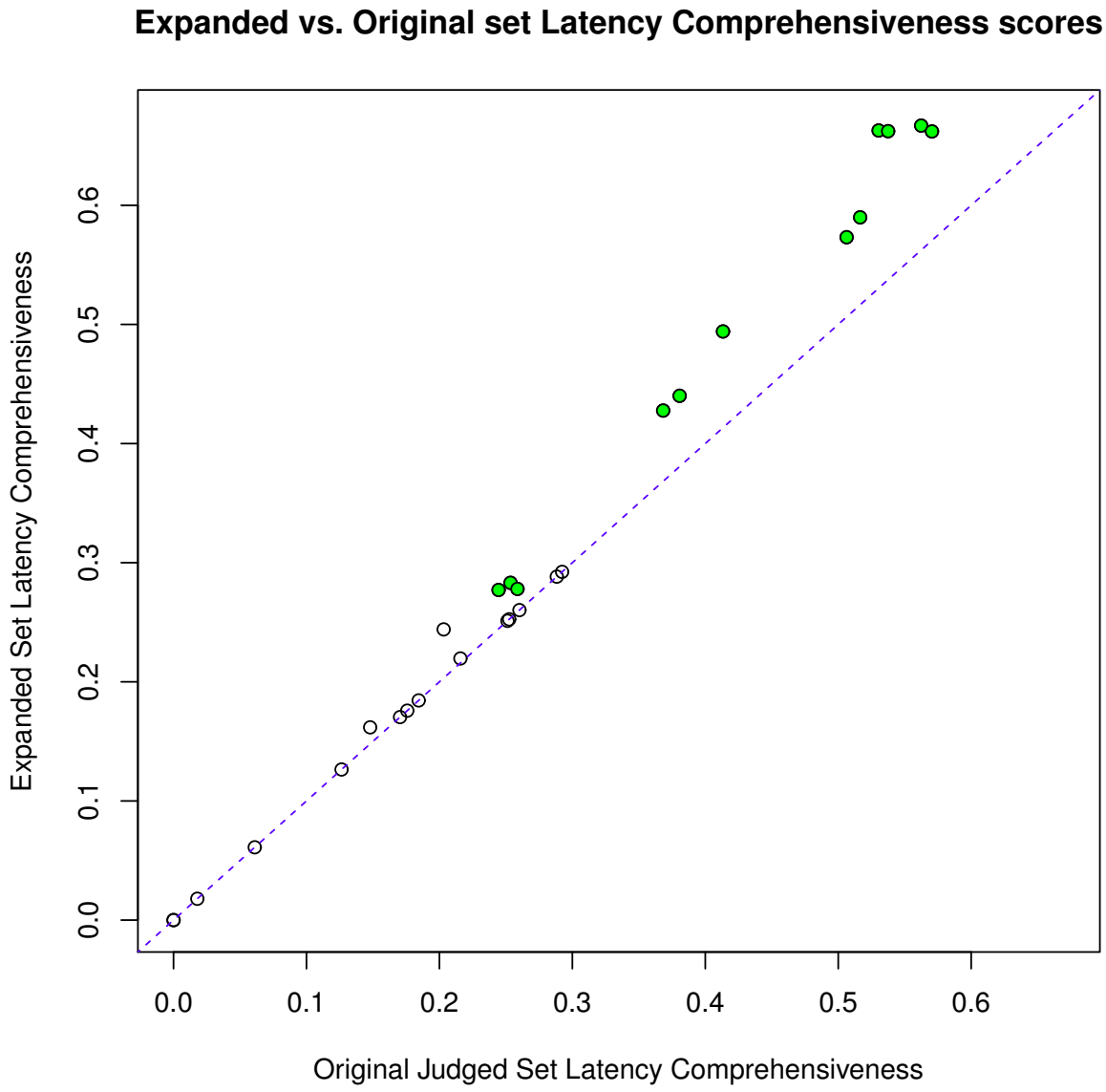


Figure 4.2: LC scores for the systems using the duplicates-expanded set of judged sentences vs. the LC scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute LC score for the respective run.

Judged Sentences expanded with	Kendall's τ for Ranking Metric			
	E[LG]	E[G]	LC	C
exact duplicates	0.899	0.894	0.942	0.937
lowercase duplicates	0.899	0.894	0.942	0.937

Table 4.3: Rank correlation between Original Judged sentences vs Expanded set, for TST 2013 measures.

4.2.2 Discussion

Although, the relative performance of the runs did not change, we can see that the addition of duplicates to the qrels helps in estimating absolute scores more accurately. The evaluation of TST 2013 follows the guidelines laid out by Sakai and Kando (2008), that suggest eliding to be appropriate for test collections having highly incomplete sets of judgements, which is indeed the case for the TST. It is indeed the case that the by adding duplicates, we have expanded the judged set and therefore, increased the verbosity normalization component. This would cause the ELG scores to go down, which is clearly the case as seen in Figure 4.1.

However, ignoring the unjudged sentences for evaluation may not truly reflect user experience. There should be a difference in the scores between systems that return 1000 updates vs. systems that return 100,000. Eliding causes the verbosity penalization to be limited to the number of sentences in the pool which is a bonus for systems returning a large number of sentences because very few of their sentences are present in the pool.

4.3 Expansion of TST 2014 qrels with Duplicates

For the TST 2014, the track organizers released an *extended* set of judged sentences along with the judged set of sentences created for the track. The extended set of judged sentences was created by identifying exact duplicates of pooled sentences from all the submitted runs. These exact duplicates were then added to the qrels to create the extended set.

For evaluation of submitted runs with duplicates extended qrels, expanding the pool using only the duplicates from the within the submitted runs is sufficient. This is in contrast to the method described in Section 4.1, where duplicates were identified from within the corpus (rather than from only the submitted runs). Table 4.4 lists the duplicates of pooled (and judged) sentences identified from within the submitted runs for TST 2014. Table 4.5 similarly list the duplicates of pooled (and judged) sentences from within the submitted runs for TST 2013. As can be seen, the number of submitted duplicates has a much lesser dramatic volume than the duplicates found from the corpus.

4.3.1 Evaluating TST 2014 Runs using qrels Expanded with Duplicates

As was the case for TST 2013, when duplicates are added to the qrels, there is no significant change in relative performance of the submitted systems for TST 2014. The relative performance remains unchanged for the HM, nELG, ELG and LC measures of the TST 2014 track. Table 4.6 lists measures used for evaluation at TST 2014, and the respective correlations, between the scores computed using the standard (original) qrels and the scores

topic	#pooled unique updates	#duplicates in pool	#submitted duplicates
TS14.11	1,008	141	441
TS14.12	654	159	969
TS14.13	570	98	7,871
TS14.14	984	398	33,524
TS14.15	813	95	1,161
TS14.16	759	146	2,617
TS14.17	768	234	22,087
TS14.18	916	160	3,097
TS14.19	758	168	1,745
TS14.20	612	148	4,326
TS14.21	919	306	14,473
TS14.22	608	158	1,892
TS14.23	723	219	2,057
TS14.24	951	247	3,330
TS14.25	701	230	4,451
Total	11,744	2,907	104,041

Table 4.4: Number of unique sentences in the pool, known duplicates (in the pool), duplicates found within the submitted runs, for TST 2014.

topic	#pooled unique updates	#duplicates in pool	#submitted duplicates	#duplicates in corpus
1	679	100	950	833,115
2	732	180	5,371	2,240,857
3	650	112	9,215	551,495
4	1,187	276	5,419	263,287
5	1,069	0	3,929	820,828
6	1,330	187	12,935	728,966
8	923	205	5,739	1,056,720
9	701	172	9,118	2,429,754
10	521	89	1,095	101,365
Total	7,792	1,321	53,771	9,026,387

Table 4.5: Number of unique sentences in the pool, known duplicates (in the pool), duplicates found within the submitted runs, for TST 2013.

Evaluation Measure	Kendall's τ	#runs with stat. Sig. difference in score
HM(nELG, LC)	0.976	3
nELG	0.985	3
ELG	0.980	2
LC	0.949	7

Table 4.6: TST 2014 measures, rank correlations between the standard and duplicates-expanded qrels, and the number of runs that showed statistically significant (p-value $\leq$ 0.05 over a paired t-test) changes in scores.

computed used the duplicates expanded qrels.

Figures 4.3, 4.4, 4.5 and 4.6 show the correlations between the evaluation scores computed using the original judged set (standard qrels), and, the judged set expanded with duplicates, for the HM, nELG, ELG and LC metrics of the track respectively. As can be seen, for HM, ELG and nELG the scores of the runs are highly correlated. Of particular interest here is the fact that very few of the 24 submitted runs experienced a statistically significant change in absolute score, for TST 2014. In comparison, half of the 26 submitted runs experienced a statistically significant change in absolute scores, for TST 2013. For TST 2014, only the LC scores (Figure 4.6) are affected when using the duplicates expanded qrels, with only marginal differences in the scores found for the HM, ELG and nELG measures.

4.3.2 Discussion

The effect of including duplicates for evaluation of temporal summarization runs remains same with regards to the relative performance measurement across both TST 2013 and

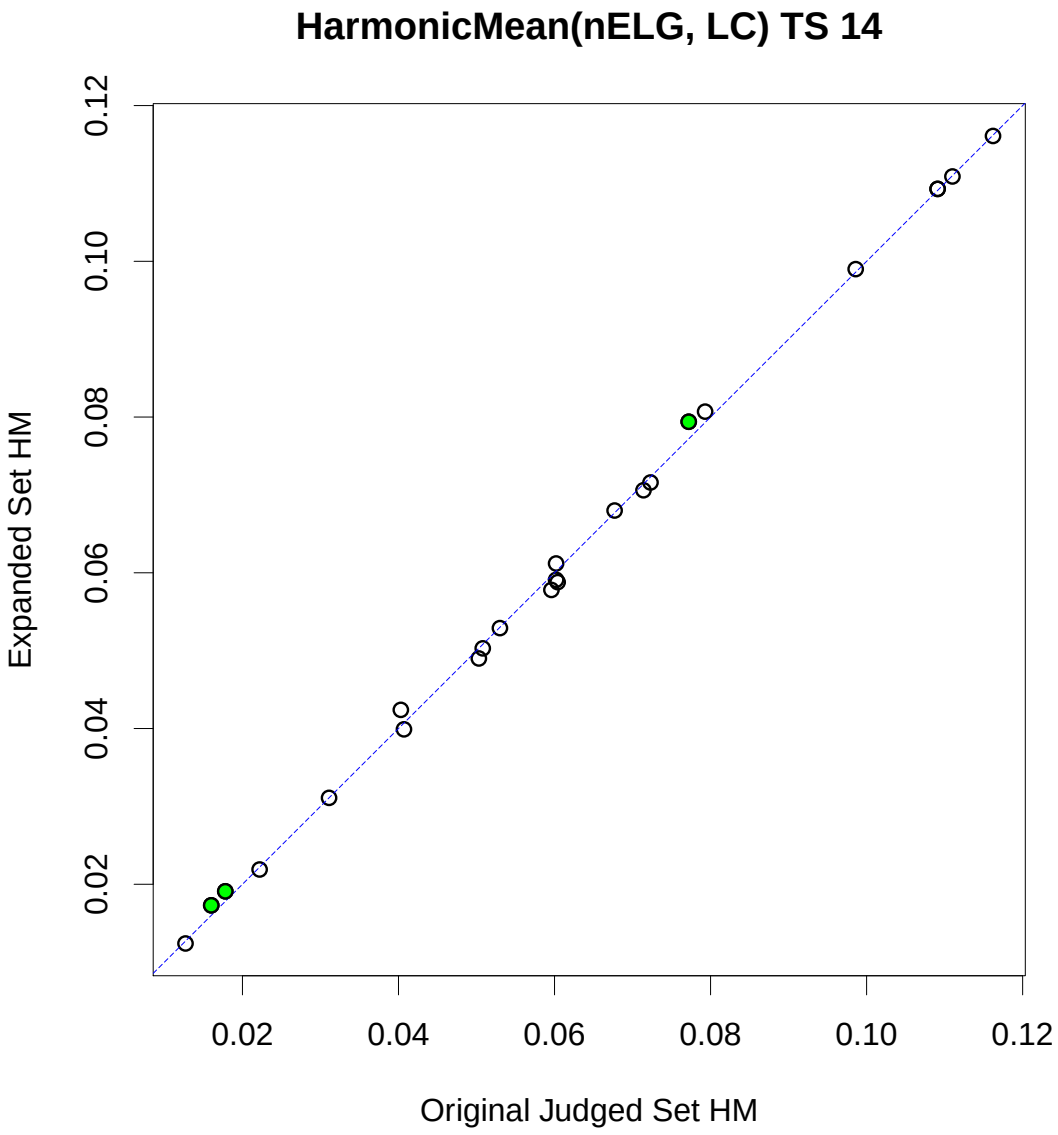


Figure 4.3: HM(nELG, LC) scores for the systems using the expanded set of judged sentences vs. the HM scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute HM score for the respective run.

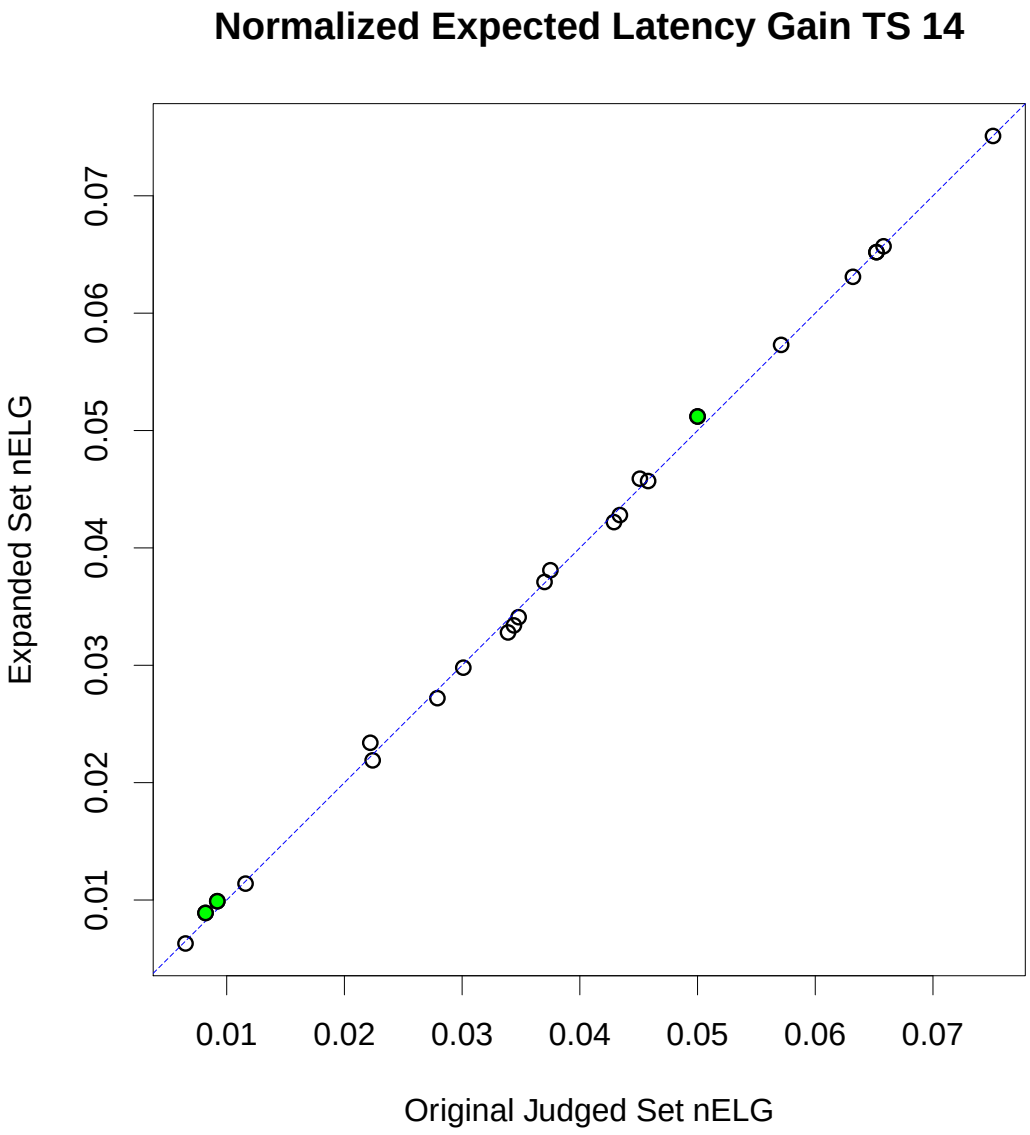


Figure 4.4: nELG scores for the systems using the expanded set of judged sentences vs. the nELG scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute nELG score for the respective run.

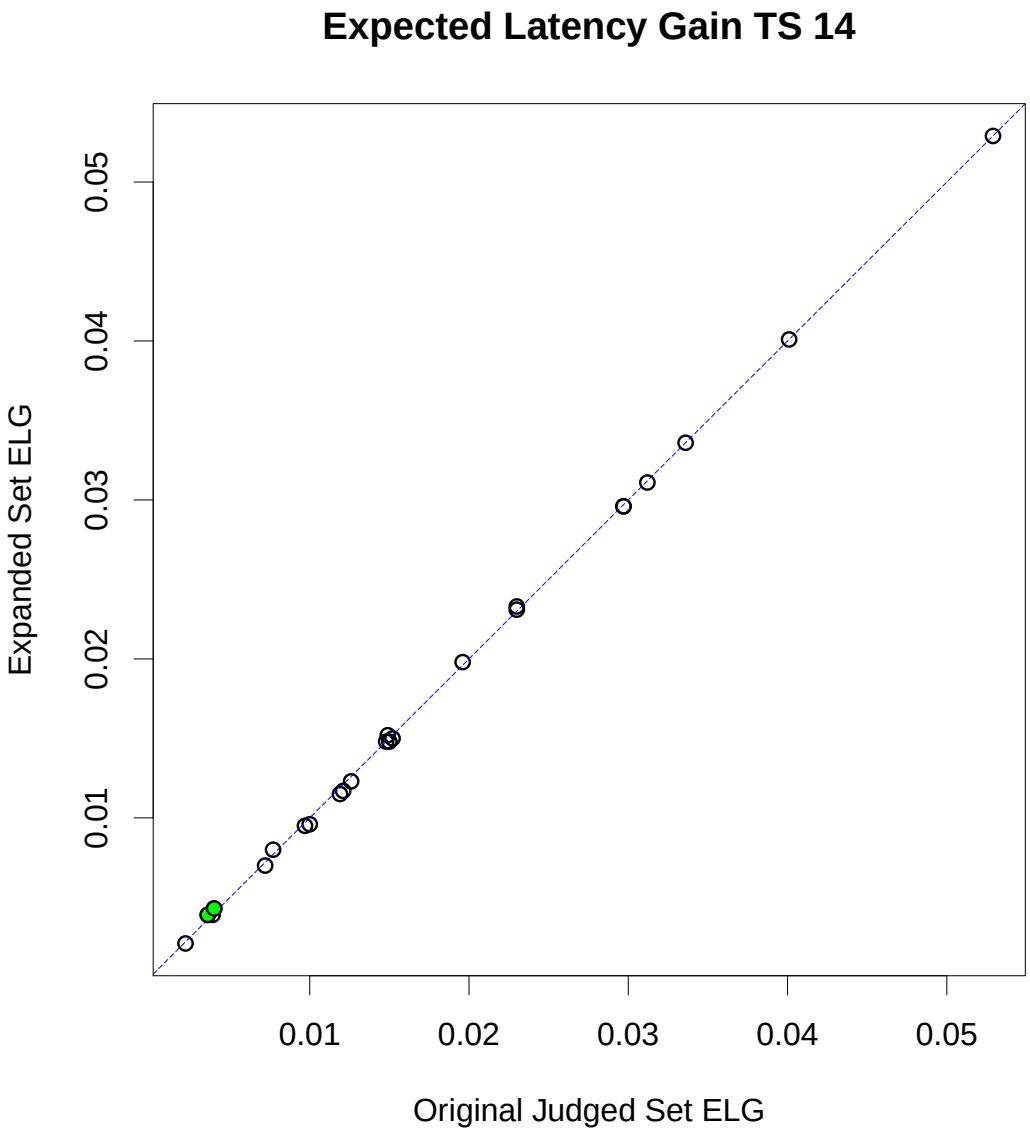


Figure 4.5: ELG scores for the systems using the expanded set of judged sentences vs. the ELG scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute ELG score for the respective run.

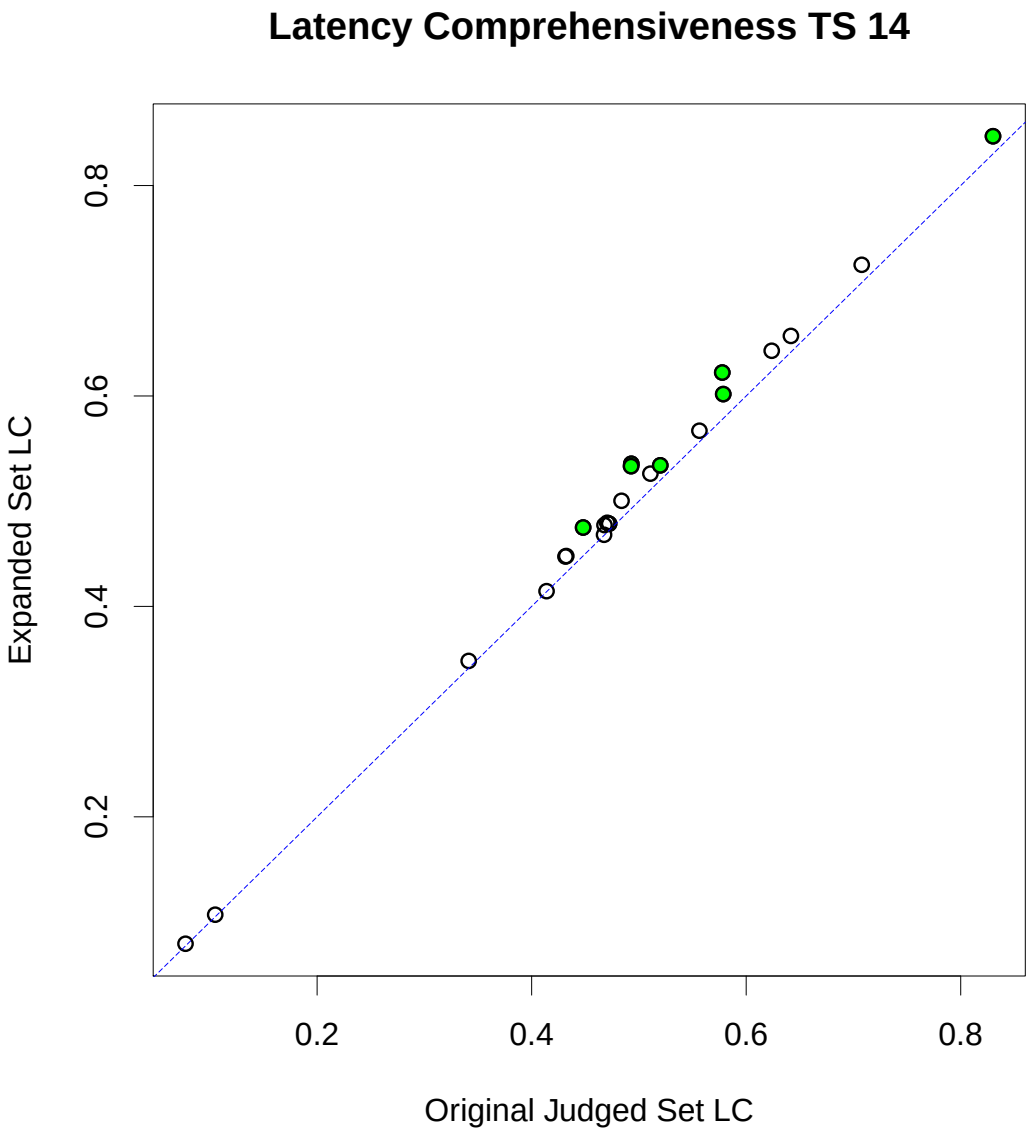


Figure 4.6: LC scores for the systems using the expanded set of judged sentences vs. the LC scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute LC score for the respective run.

TST 2014. However, there is considerable difference in the number of runs that were affected in terms of absolute scores. 13 (of 26) runs (Figure 4.1) showed a statistically significant change in absolute scores for ELG in TST 2013, whereas only 2 (of 24) runs (Figure 4.5) showed a significant change in absolute scores for ELG in TST 2014.

A major change in the evaluation process between TST 2013 and TST 2014 is that TST 2014 does not elide unjudged sentences. Instead, sentences not in the evaluation pool are considered non-relevant for TST 2014. For TST 2013, eliding limits the number of sentences being considered for evaluation for each run, to the judged set of sentences. Adding duplicates to the qrels increases the size of the judged set and hence it also increases the verbosity penalization for ELG. For TST 2014, the non-eliding of unjudged sentences helps to avoid any additional verbosity penalization because all the duplicates have already been penalized for verbosity, thus causing minimal changes in scores even when duplicates are considered in the qrels.

For LC, for TST 2013, 12 (of 26) runs (Figure 4.2) showed a statistically significant change in absolute LC score. For TST 2014, all but 4 runs show a change in the LC score, however, only 7 runs show a statistically significant difference. The reason for the change in score, as was the case for TST 2013, is that runs encounter duplicate sentences having earlier timestamps, thereby reducing the potential latency discount and thus increasing the LC score. Regardless of the change in score for LC, the primary measure for TST 2014, the harmonic mean of nELG and LC, does not show significant change in the relative performance, and shows a significant change in absolute performance for only 3 runs when duplicates are added to the qrels.

4.4 Conclusion

In tasks like temporal summarization where the same information may be available from multiple sources (documents), identification of duplicates is essential in order to determine which systems returned identified relevant information earlier than others. We investigated the effect of expanding a judged set of sentences with their exact duplicates found from the corpus (or submitted runs), on the evaluation of the temporal summarization tracks at TREC 2013 and 2014.

We found that the addition of duplicates to qrels does not affect relative system performance measurement across both tracks, and across various evaluation measures such as ELG, LC, and the Harmonic Mean of nELG and LC. However, the addition of duplicates to qrels does change the absolute scores for half the systems on ELG and LC, for TST 2013. The absolute scores on ELG change minimally for TST 2014, mainly because the TST 2014 evaluation does not elide unjudged sentences.

Not eliding helps to alleviate the effect of including duplicates in qrels for measuring relative performance. The absolute scores of runs are affected depending on the evaluation measure. For instance, absolute scores of runs on ELG for TST 2014 remain largely unaffected; however, LC scores are affected. It is advantageous to know that duplicates have no appreciable effect on ranking a given set of systems; however, including duplicates does help to measure absolute scores more accurately, over a variety of evaluation measures, for temporal summarization.

Chapter 5

Modeled Stream Utility

Modeled Stream Utility (MSU) is a user oriented evaluation measure, designed to measure the performance of systems that produce a stream of likely relevant updates about a breaking news event. MSU employs a model of user behavior for streaming information access and utilizes it for measuring system performance. We describe the MSU evaluation model in Section 5.1. We simulate users reading updates that were returned by the participating systems at TST 2013 and we utilized the TST 2013 test collection to demonstrate system performance measurement using MSU (Section 5.3). We also compare MSU with the TST measures ELG and LC whilst varying the MSU user model parameters.

Our evaluation framework takes as input the updates emitted by a temporal summarization system (Figure 5.1b). With the help of the track's *qrels* (relevance judgements), we map every relevant update to the nuggets it contains. We term this filtered stream of time ordered updates (and their contained nuggets) as the *update-trace* (Figure 5.1c). We

simulate a user reading updates in sessions (Section 5.1.3) via an interface that presents updates in a reverse chronological order (Section 5.1.4). The user model, the user interface and the update-trace combined, enable us to evaluate systems for users having different behaviours. We describe the formulation of MSU in Section 5.2 and demonstrate the evaluation of systems using MSU in Section 5.3.

5.1 User Model for Streaming Information Access

A particular news event may hold different importance for different users. Users may also be constrained by the amount of time they have available to keep up-to-date on an evolving event. Thus, users may have different behaviours regarding when and how frequently they decide to get updated about an event. We endeavour to capture these aspects of user behaviour in our MSU user model for streaming information access.

5.1.1 Model Parameters for a Single User

We model a single user having 3 parameters

- average session duration D: The amount time a user spends with the system on average.
- average time away A: The amount of time the user spends not using the system.
- reading speed V: The speed at which a user reads text.

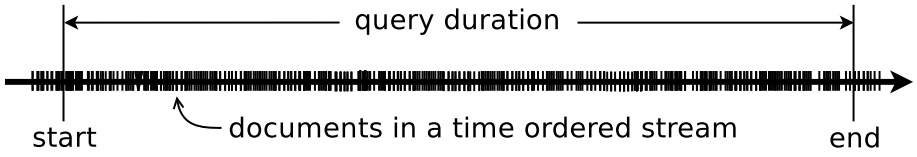
A user is imagined to alternate between spending time with the system (D) and away from the system (A). The number of updates a user reads in a given session is then determined by the reading speed (V) of the user.

One can imagine a user checking back for updates from time to time. A typical user may spend some time reading updates (a session), and then perform some other daily/routine/specific task for a time (away time). The user may return for more sessions at a later times, for as long as the topic holds the user's interest. Thus, the average session time and average away time, capture to some degree, the interest in the topic (event) as well as the time availability of users for reading updates about an event.

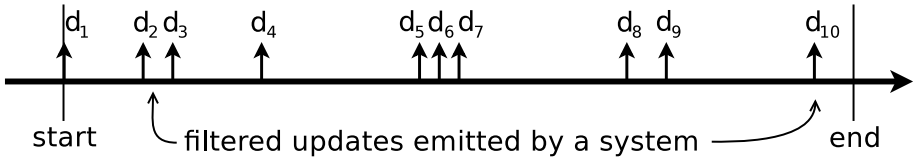
5.1.2 Modeling a User Population

The single user's model has 3 parameters, D—mean session duration, A—mean away time, and V—the reading speed. To simulate a population of users, for each of whom the parameters D, A and V would differ, we require appropriate parent distributions for the respective parameters. We model the parent distributions as lognormal distributions over the user population.

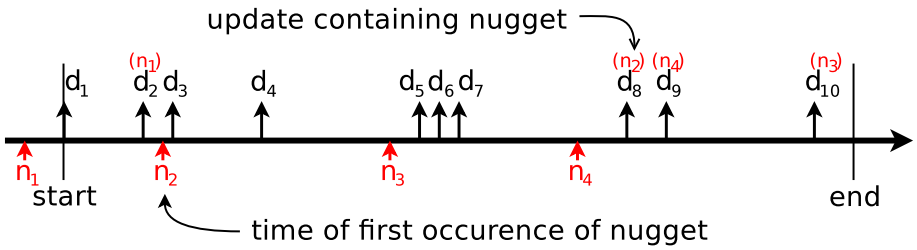
Log-normal distributions can reflect human performance adequately across various tasks (Clarke and Smucker, 2014; Doherty, Massink, and Faconti, 2001). A log-normal distribution is a distribution of a random variable, the log of which, is normally distributed. A log-normal distribution is parameterized by the mean (μ) and standard deviation (σ) of the natural logarithm of the underlying data. If the underlying data has mean M and standard deviation S, then, for the lognormal distribution, the variance is $\sigma^2 = log(1 + S^2/M^2)$, and



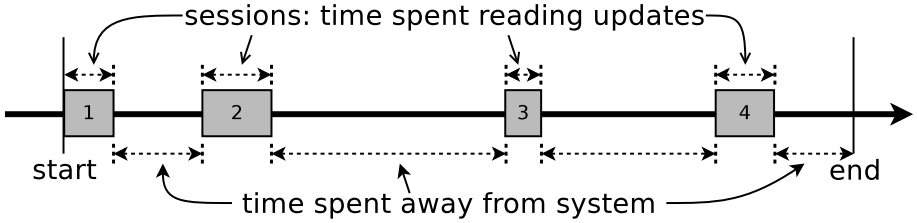
(a) Input: A time ordered document stream.



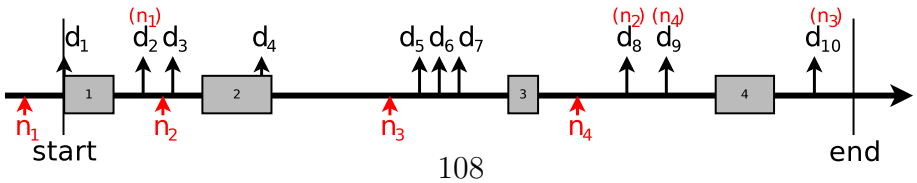
(b) Output: Stream of updates $d_1..d_{10}$ emitted at various times by a system.



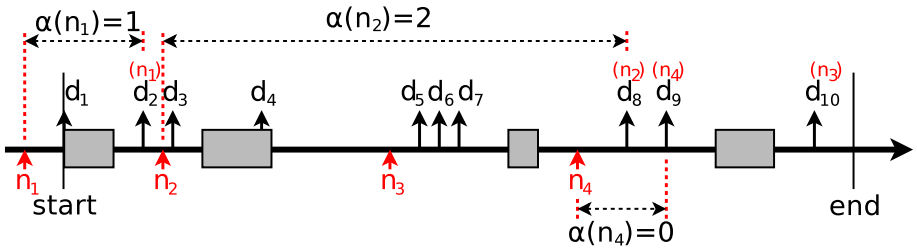
(c) Update-trace: Times of first occurrence of nuggets are identified. Updates containing nuggets are noted.



(d) User-trace: Simulated behavior of a user who reads updates from the stream from time to time.



(e) Reading-trace: Determines which updates are available to read for every session. e.g. d_3, d_2 are available to read at the start of session 2. The user's reading speed V determines which nuggets are actually read. Reading updates that contain nuggets adds to gain.



(f) $\alpha(n)$: Gain is discounted by $L^{\alpha(n)}$. $\alpha(n)$ is the number of sessions between the first occurrence of nugget n and the current session within which an update reporting n is read by the user. α is computed only if the update containing n is read by the user.

Figure 5.1: The MSU Evaluation Process.

the mean is $\mu = log(M) - 0.5\sigma^2$.

For session durations, we construct a *parent* log-normal distribution $Pln_D(M_D, S_D)$, that represents the mean session durations for a user population. M_D and S_D are the mean session duration and the standard deviation for the session duration across a user population respectively. Similarly, we construct $Pln_A(M_A, S_A)$ to represent the mean away times for the user population. Previous studies (Clarke and Smucker, 2014) have recorded that users on average have a reading speed of 4.3 words per second. The distribution of reading speed across users has been described as the log-normal distribution Pln_V with μ=1.29 and σ=0.558.

Thus to simulate a single user u, we sample from Pln_D to get the mean session duration D, we sample from Pln_A to get the mean away time A and we sample from Pln_V to get the reading speed V respectively. Multiple samples from respective parent distributions give us a simulated user population with each user having potentially different stream browsing behavior.

5.1.3 Modeling User Behavior

Once we have the M and A time for user u, we can then simulate the user alternating between spending time with the system and away from the system generating a *user-trace* of reading and away times bounded by the query duration (Figure 5.1d).

A user can be expected not to have fixed session duration and away times; i.e., the duration of each session or away time may vary about their respective means. We model individual session and away durations using respective exponential distributions. For instance, durations of individual sessions can be sampled from an exponential distribution with the rate parameter $1/D$ (where D is the mean session duration for the user). A random deviate sampled from this distribution is the length of a session and the mean of all sampled deviates is D. Similarly, random deviates from an exponential distribution parameterized by the rate parameter $1/A$ represent the lengths of time spent away from the system. Using an exponential distribution has the advantage that it requires only one parameter.

It may be the case that session durations follow a different distribution. Indeed, the parent distributions could also be different than lognormals. For our experiments we have assumed respective distributions based on prior research. The correct distributions, however, may possibly be estimated after observing actual user behavior via a user study, or through search log analysis (Section 7.3.1). To keep the model simple, we do not currently model effects of day/night periods on browsing behavior that could affect session and away durations.

5.1.4 User Interface and User Interaction Behavior

The user interface and interaction mechanism for the system may also affect user behaviour. We therefore also simulate a simplistic user interface for reading updates wherein, for every session, the user reads the latest update first, i.e., in reverse chronological order. Such user interfaces are a common feature in social networking services like Twitter[1], Facebook[2], Tumblr[3], where most recent updates are presented first (at the top of the page or rank 1) to the user with progressively older updates appearing lower in the presentation order.

MSU assumes the following user behavior and system interactions for evaluating a system. A user first initializes the system with a query representing the breaking news event, e.g., "hurricane sandy". The system then starts filtering the input stream and emits the latest updates that are likely relevant to the query. Users check the system as per their user-trace to read the emitted updates, for as long as the user is interested in the event (the length of the query duration). At the start of every session in the user-trace, the updates emitted between the end of the last session and the start of the current session are presented to the user in a reverse chronological order, so that most recent information is available to read first. The user starts reading the latest update and then reads the next older update, and so on, until he encounters an already read update and stops reading further. In case the session ends, the last update that is partially read in the session is considered as unread. The first session always starts immediately after the initialization step. The number of updates a user reads in a session depends on the reading speed V for

[1] www.twitter.com
[2] www.facebook.com
[3] www.tumblr.com

the user.

5.2 MSU Evaluation Model

TST 2013 uses a nugget based evaluation framework where nuggets represent key facts of information that are pertinent to a given topic. If a returned update contains one or more nuggets, then it is considered to be a relevant update. The ELG evaluation metric measures the average quality of updates returned by a system (Section 3.1.2)

For MSU, we employ the notion of unit gain per nugget read by a modeled user. When a user reads an update that contains a nugget we consider that nugget to be *reported* to the user. We also maintain that an update containing a previously read nugget does not contribute to an increase in gain.

We observe that for TST 2013, the nuggets can be extremely short (1 word) or long (180 words) with the mean nugget length of 11.67 across topics. In comparison, average length of an update submitted to TST 2013 is 62.96 words. For shorter nuggets, the containing sentence provides a context that helps determine relevancy. We therefore enforce that gain for an update is only awarded if it is read completely. No gain whatsoever is awarded for a partially read update.

By overlaying the user-trace over the update-trace, and by considering the reading speed of users, we get a *reading-trace*. The reading-trace helps us to determine exactly which updates have been read at every user session (as shown in Figure 5.1). A user is considered to have experienced gain if the user reads a relevant update.

5.2.1 Measuring Lateness

At every session, a user would expect to see updates about what transpired between the previous session and the current session (i.e., what new information came to light during the just concluded away time). Therefore, we are now in a position to determine for every reported nugget n, the function $\alpha(n)$, that measures how many sessions ago the nugget should have been reported (Figure 5.1f). $\alpha(n)$ tries to capture the user's perception of late reporting of nuggets. As long as the latest nuggets are available to read at every session, the user may find the system to be suitable; otherwise the user experiences a drop in the gain from late reported nuggets.

We employ a notion of discounting gain by latency in a manner that aims to reflect a user's perception of lateness. A nugget, if reported late, would be considered less relevant, the later it is reported. Ideally, this relation between lateness and relevance of a nugget should be captured by a probability distribution. For now, we assume that this relation follows an exponentially dropping probability distribution.

We compute the gain for every reported and read nugget n as,

$$g(n) = 1 \times L^{\alpha(n)} \tag{5.1}$$

where, L is the penalty for late reporting of nuggets. L is a pre-determined value representing the loss in gain of information if it is reported late. For instance, $L = 0.5$ indicates that a nugget is half as likely to be considered relevant for every session in which it is not reported. That is, L reflects a user's preference for receiving nuggets as early as possible.

For our experiments, we vary L from 0 to 1, where 0 indicates that a nugget would be considered non-relevant if it is not reported in the immediately following user session, and 1 indicates that the nugget remains relevant regardless of how late the reporting. $L = 1$ might be preferred by users who want to know all details about the news topic irrespective of its time of occurrence; such users could be report writers, news summarizers, or analysts. $L = 0$ might be preferred by users who are already in the know and would rather have the very latest information; such users could be health responders, law enforcement officials, or relief providers.

5.2.2 Expressing Modeled Stream Utility

With the knowledge of which updates were read by a user, MSU for the user is simply the number of nuggets read by the user. The gain from each reported nugget is added up to get the cumulative binary latency discounted gain for a user over the query duration. Thus, the binary discounted gain for a simulated user for a given topic is

$$MSU = \sum_{n} g(n) \tag{5.2}$$

where, each n is a nugget that was read by the user. The MSU for a topic is the average MSU for users and the MSU for a system is the average MSU over topics.

The simple formulation of MSU is afforded by the MSU user model that has 6 parameters in all. These are

1. Mean session duration across a user population (M_D).

2. Standard deviation of session duration across a user population (S_D).

3. Mean away time across a user population (M_A).

4. Standard deviation of away time across a user population (S_A).

5. Lateness Decay parameter (L) reflecting the loss in likelihood of a nugget being considered relevant.

6. The reading speed (V) of a user.

Although the reading speed V has a lognormal distribution (Clarke and Smucker, 2014), for our experiments, we assume that a user's reading speed does not vary depending on times of crisis or by level of interest. To the best of our knowledge, there is no prior evidence to indicate that a user's reading speed varies in times of crisis, or according to the level of interest of the user. As such, we utilize the reading speed distribution described by Clarke and Smucker (2014), without any modifications.

Thus given parameters M_D and S_D we can construct a distribution Pln_D of session durations, from which we sample D for users. Similarly, given parameters M_A and S_A, we can construct a distribution Pln_A of away times, from which we sample A for users. We sample the reading speed from the reading speed distribution Pln_V. L is a value determined by the user.

Rank (by ELG)	GroupID	RunID	ELG	LC	average # updates per topic	MSU	Rank (by MSU)
1	PRIS	cluster5	0.136	0.126	21.9	4.35	17
2	ICTNET	run2	0.127	0.251	93.8	9.45	4
3	ICTNET	run1	0.125	0.253	97.8	9.46	3
4	hltcoe	TuneExternal2	0.118	0.203	799.4	5.34	16
5	hltcoe	TuneBasePred2	0.114	0.244	2696.1	5.49	15
6	PRIS	cluster3	0.103	0.176	42.3	5.99	11
7	PRIS	cluster2	0.074	0.26	122.1	9.31	5
8	uogTr	uogTrNMTm1MM3	0.069	0.216	358.8	7.28	7
9	PRIS	cluster1	0.067	0.288	164.8	9.57	1
10	hltcoe	cluster4	0.067	0.292	163	9.55	2
11	PRIS	BasePred	0.067	0.368	8790.7	5.84	13
12	hltcoe	Baseline	0.063	0.381	12743	5.87	12
13	uogTr	uogTrNSQ1	0.06	0.184	139	6.85	9
14	hltcoe	EXTERNAL	0.055	0.413	22476.1	5.6	14
15	uogTr	uogTrNMTm3FMM4	0.049	0.17	168.3	6.33	10
16	uogTr	uogTrNMM	0.045	0.254	954.7	7.63	6
17	uogTr	uogTrEMMQ2	0.04	0.259	2077.8	6.88	8
18	wim_GY_2013	SUS1	0.036	0.128	2338.7	3.62	18
19	UWaterlooMDS	rg4	0.028	0.516	41863.3	1.44	20
20	UWaterlooMDS	rg3	0.026	0.506	42534.1	1.45	19
21	UWaterlooMDS	rg2	0.022	0.562	299559.6	0.32	25
22	UWaterlooMDS	rg1	0.021	0.571	312863.3	0.3	26
23	UWaterlooMDS	UWMDSqlec4t50	0.018	0.53	213735.7	1.02	21
24	UWaterlooMDS	UWMDSqlec2t25	0.017	0.537	230056	0.61	23
25	UWaterlooMDS	CosineEgrep	0.01	0.018	11.9	0.55	24
26	UWaterlooMDS	NormEgrep	0.001	0.061	151.3	0.73	22

Table 5.1: ELG, MSU and respective ranks for each run.

5.3 MSU Parameter Sweep

To better understand the parameters of MSU's user model, we simulate user populations with various characteristic behaviours. We simulate these users reading updates from the runs submitted to TST 2013. Finally, we compare the various MSU rankings (induced by various MSU model parameters) in terms of the rankings produced by ELG (Section 5.3.2).

We simulate users using our user model that has 6 parameters. 4 of these parameters define the away time (mean M_A, stdev S_A) and session duration (mean M_D, stdev S_D) of a user population. The decay parameter L, is representative of the severity for late information as perceived by the user. The reading speed parameter V is sampled once for every user from Pln_V. For each user population, we measure system effectiveness using MSU and compare the relative performance of systems over MSU with the relative performance of systems over ELG.

For an initial realization of our user model we simulate 1000 users having a M_D of 2 minutes (and S_D of 1 minute), and a M_A of 3 hours (and S_A of 1.5 hours), with L set as 0.5. We presume that these choices of parameters represent users who are reasonably interested in the event.

We then proceed to explore our parameter space in order to better understand the interplay between the parameters, i.e., how do users having different user-traces perform when using various systems. We vary away time from 5 minutes to a day, session duration from 30 seconds to 30 minutes, and L from 0 to 1, to generate 2646 parameter sets simulating a total of 2,646,000 users. Our choices of parameter ranges are influenced in a large

part by the nature of the TST task, where the updates are essentially sentences. For applications where documents / passages / media form updates, the parameters would need to be set/calibrated accordingly based upon the characteristics of the returned document. A search through parameter space, allows us to see which types of users might have benefited most (or performed the worst), based on TST's ELG metric's rank order.

5.3.1 MSU for Reasonable Users

We simulate 1000 users with parameters (M_A= 3 hours, S_A= 1.5 hours, M_D= 2 minutes, S_D= 1 minute, L= 0.5) representing "reasonable" users. We construct the corresponding log normal distributions for away time (Pln_A) and session duration Pln_D. We sample Pln_A and Pln_D 1000 times for each user's mean away time A, and mean session duration D, respectively. Further we generate a user-trace for each of the 1000 modeled users using their A and D parameters. Every run has 9 update-traces (one for each topic). For a run, we merge one user-trace with the update-trace of every topic to create the reading-trace. The reading-trace aids in the computation of user performance (MSU) for every topic. We measure MSU for 1000 simulated users for each topic and compute the average MSU per user for a topic. A system's effectiveness is then simply the average MSU across topics.

Figure 5.2 shows the correlation of MSU with ELG. The runs are ordered as per their rank on ELG, e.g., the right most point represents the top run at TST 2013. Thus going from right to left along the ELG axis, we can see the ranking by ELG. Similarly, going from top to bottom along the MSU axis, shows the relative ordering as per MSU. The two metrics have a correlation of Kendall's $\tau = 0.4708$ for the reasonably interested users.

AP correlation (t_{AP}), developed by Yilmaz, Aslam, and Robertson (2008) and extended to handle ties by Smucker, Kazai, and Lease (2013), computes a correlation that treats differences in high ranks as more important than low ranks. Our reasonable users have a $t_{AP} = 0.4052$. As we can see from the Figure 5.2, the runs in the middle-lower positions of the ELG ranking show a spectacular jump to the top positions. For such a scenario Kendall's τ could be suitable as it equally weights all changes in rank positions. We use Kendall's τ for comparing MSU with ELG, however, we also report τ_{AP}.

Table 5.1 can help us understand this graph to some extent. The top ELG run `cluster5` produces a gain of 6.889 with 21.889 updates emitted per topic over a 10 day query duration. The simulated user spends about 2 minutes every 3 hours, i.e., about 160 minutes reading, on average, over a 10 day query duration. With an average length of each update being 62.959 words, and the average reading speed being 4.3 words per second, it takes on average just 318.63 seconds to read all the 21.889 updates of `cluster5`. The user, who is willing to read for 160 minutes in total, thus derives very low gain from `cluster5` on average. On the other hand, with an adequate supply of updates, the user should be able to derive more gain on using the system.

Runs `cluster1` and `cluster4`, both emit on average, 164.7 and 163 updates per topic respectively. They both have the same amount of cumulative gain over the query duration (17 nuggets each, as per Table 3.7). They have also been submitted by the same team. Over a thousand users, the 2 runs produced a MSU of 9.566 and 9.549 respectively. Note that the users are potentially not reading all the updates. They are only reading updates as per the session durations in their reading-traces (Figure 5.1).

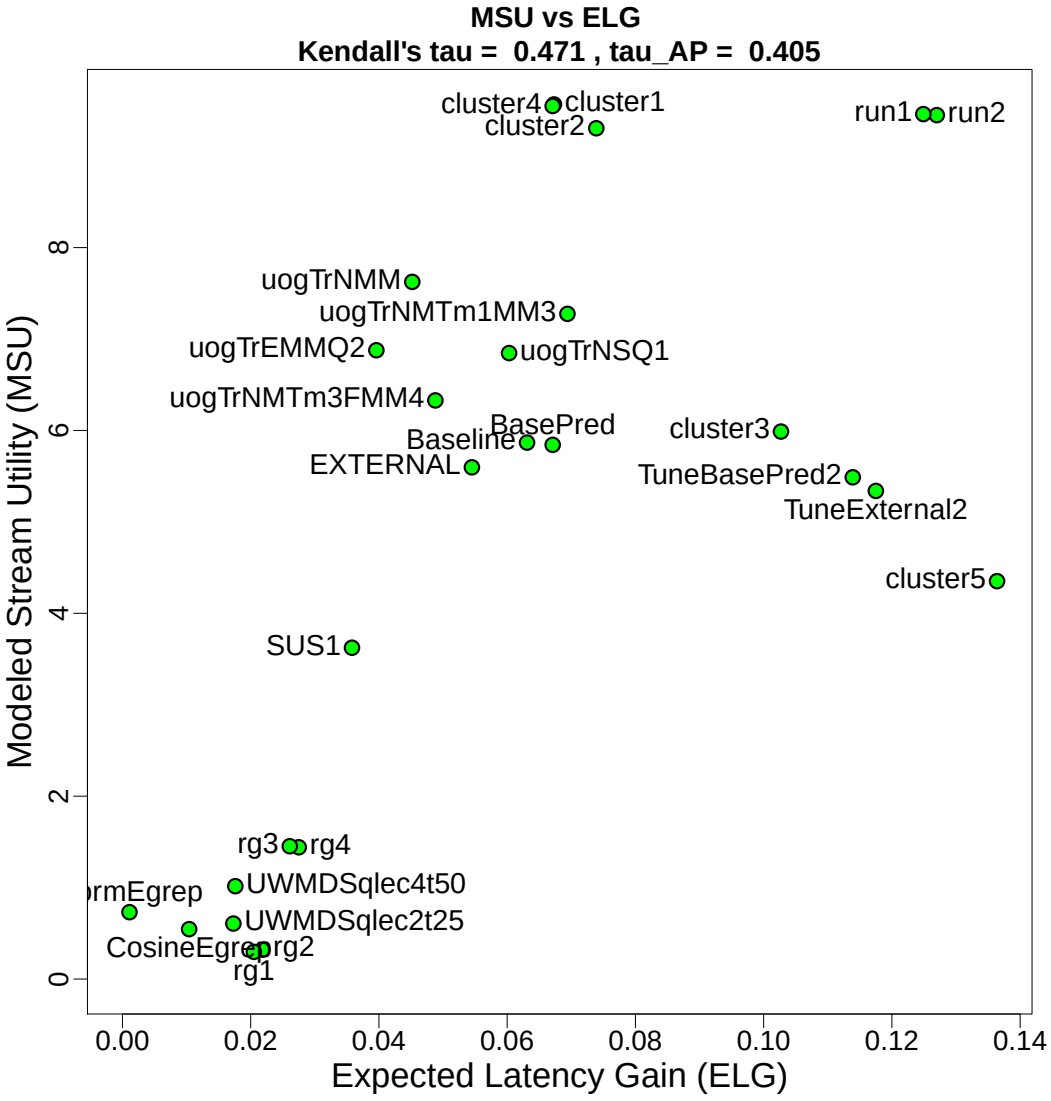


Figure 5.2: MSU vs ELG correlation for users reading updates for about 2 minutes every 3 hours on average.

Figure 5.3 shows the correlation of MSU with LC. The two measures have a correlation of Kendall's τ = -0.108. That is, MSU is slightly negatively correlated with LC. Note that ELG and LC are themselves negatively correlated (Kendall's τ = -0.28). The negative correlation between ELG and LC is unsurprising as they are analogous to precision and recall respectively. MSU is more positively correlated with ELG rather than LC which is reasonable since both ELG and MSU explicitly count the number of nuggets in the content read by the user and thus for remaining experiments, we compare MSU with ELG only. The main difference between ELG and MSU is that the ELG assumes that the user reads all the returned content which is also penalized for verbosity, while MSU considers the subset of updates read by a user across sessions; in MSU the verbosity normalization component is subsumed in the time spent reading an update.

This experiment demonstrates the necessity for accurately estimating the number of updates read by a user in order for the user performance to be judged in correspondence with the actual gain achieved. This leads to the question about number of updates that a system should emit in order to improve upon user performance. Table 3.7 shows that the maximum average number of nuggets returned per topic by a system is 38 (run `UWMDSqlec2t25`). However it is extremely unreasonable to suppose that a user would read 230,056 updates per topic that were returned by the respective run. Obviously, the quality of updates should be high, and, the number of updates should be reasonable. For the reasonable users, the MSU top 5 runs have 90–163 likely relevant updates emitted per topic. If the reasonable users relax parameter L from 0.5 to 1, then the maximum MSU across all systems is 15.368 for run `cluster4`, which is close to its absolute cumulative gain of 17 nuggets over 10 days (Table 3.7).

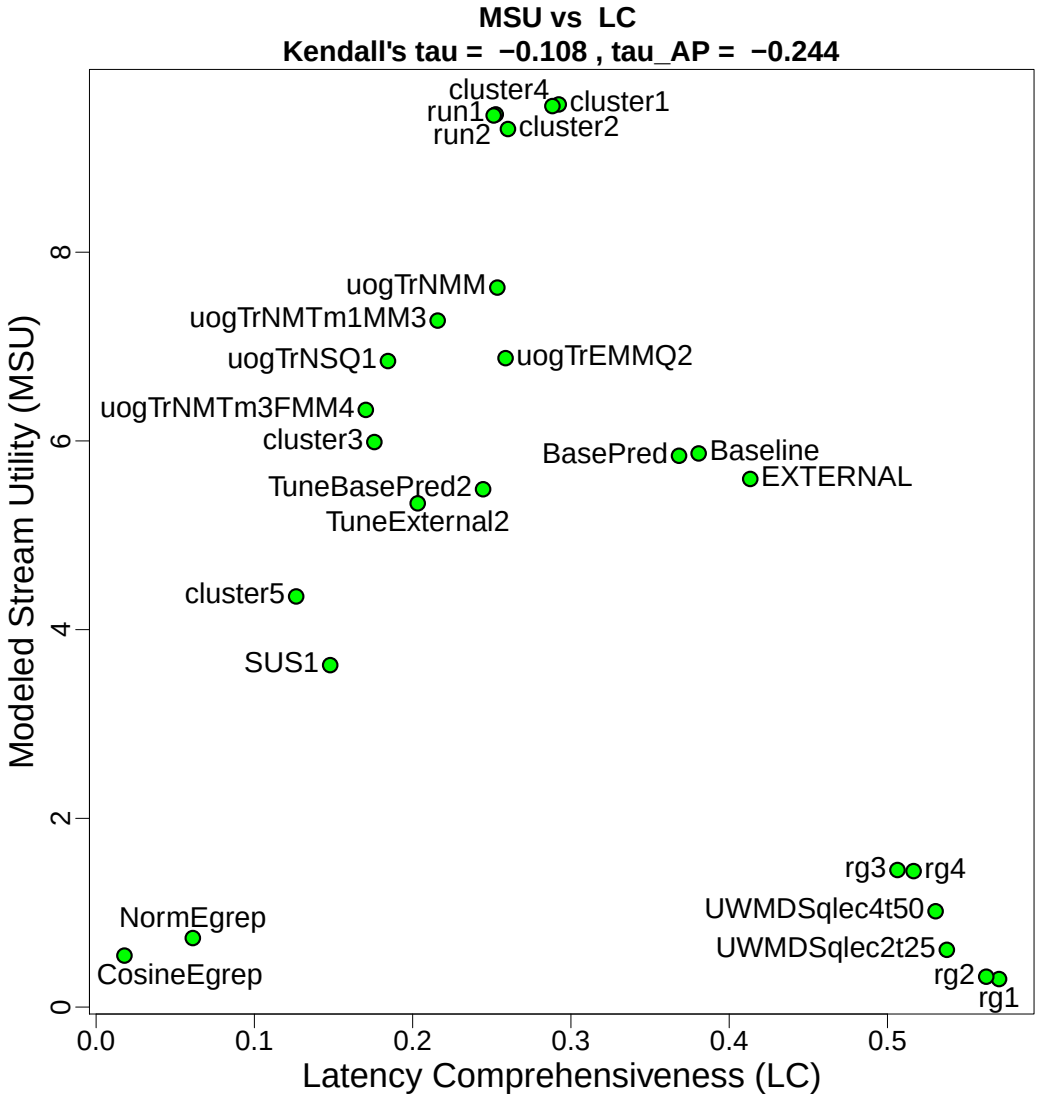


Figure 5.3: MSU vs LC correlation for users reading updates for about 2 minutes every 3 hours on average.

5.3.2 Exploring the User Model Parameters

Our initial experiment suggests that users with different behaviours might perform differently with the TST systems. Therefore an exploration of various possible user behaviours is warranted. We vary the MSU user model parameters M_A, S_A, M_D, S_D and L to construct user populations having different characteristic behavior. We conduct the parameter sweep by choosing the parameter values as follows;

- mean session durations M_D from the set {0.5, 1, 2, 5, 15, 30} minutes.
- mean away times M_A from the set {5, 10, 30 minutes, 1, 3, 6, 24 hours}.
- lateness penalty L from the set {0, 0.1, 0.25, 0.5, 0.75, 0.9, 1}.

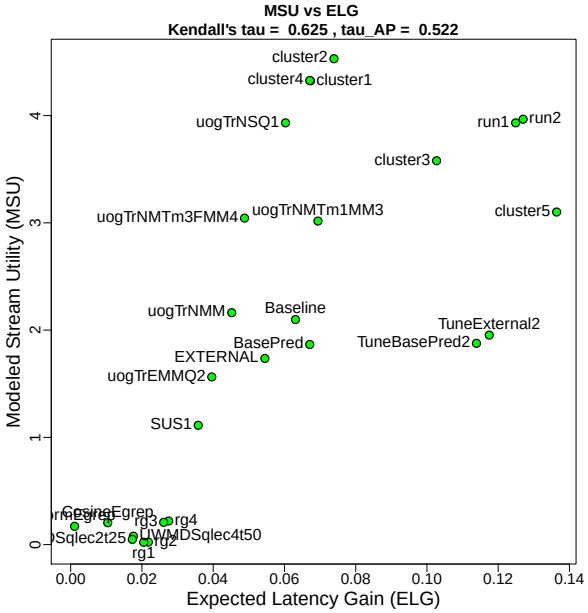
We keep the sampled reading speeds the same across all parameter sets for respective users. For the standard deviations (S_D and S_A) we choose a multiplier for the respective means from the set {.5, 1, 2}. For instance, a mean session time of 30 seconds, will be associated with standard deviations of 15, 30 and 60 seconds. Thus we generate a total of 7 (mean away times) $\times$ 3 (away time stddev's) $\times$ 6 (mean session durations) $\times$ 3 (session duration stddev's) $\times$ 7 (lateness penalties) = 2646 parameter-tuples (points in the parameter space). For instance, the parameter tuple (M_A= 6 hours, S_A= 3 hours,M_D= 5 minutes,S_D= 5 minutes, $L = 1$), is a point in the parameter space that represents users who spend an average of 5 minutes every 6 hours on average reading updates and these users do not discount relevance by lateness at all.

For each tuple in the parameter-set (selected point from the parameter space), we simulate 1000 users, generate their user-traces, determine reading-traces, and compute the

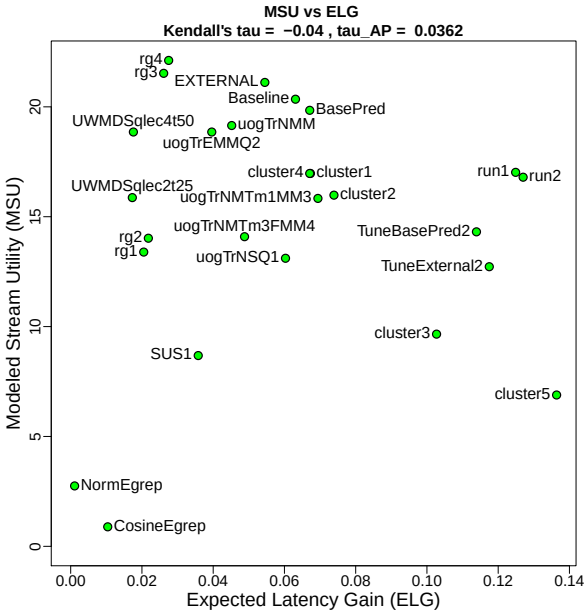
MSU across users for that point. We then compute the correlation between the relative user performance at each point with ELG.

Figure 5.4a compares the relative system performance at point (M_A= 24 hours, S_A= 12 hours, M_D= 1 minute, S_D= 2 minutes, L = 0.1), that has the maximum correlation of MSU with ELG (Kendall's τ = 0.6246), in our parameter set. It is thus likely that ELG seemingly caters to highly time constrained and selective users with low tolerance for late reporting. It is interesting to note that even when spending 60 seconds reading updates the top ELG run (`cluster5`) performs below par. This is primarily because even at the very low M_D of 1 minute the users are able to read more material than delivered by `cluster5`. With the average update length being 62.959 words, with an average reading speed of 4.3 words per second, in one minute an average user can read about 4.1 updates. Therefore, reading 21.89 updates takes at most 5.33 minutes, assuming every update is read. Over a period of 10 days a user spends 10 minutes reading on average. Thus `cluster5` runs out of updates for the user to read, making way for other runs to take the top position.

Figure 5.4b compares the relative system performance at point (M_A= 5 minutes, S_A= 10 minutes, M_D= 30 minutes, S_D= 15 minutes, L = 1), that has the minimum correlation of MSU with ELG (Kendall's tau = -0.04), in our parameter set. This set of users seem inclined to spend almost all their time with the system reading updates taking a 5 minute break every 30 minutes on average, without any dissatisfaction for late reporting of information. Unsurprisingly, these users achieve the highest amount of MSU (22.1131) with run `rg4`. However, this data point seems unreasonable as no user would realistically behave like this over a 10 day period.



(a) Maximum correlation of MSU with ELG.



(b) Minimum correlation of MSU with ELG.

Figure 5.4: Maximum and minimum correlations of MSU with ELG.

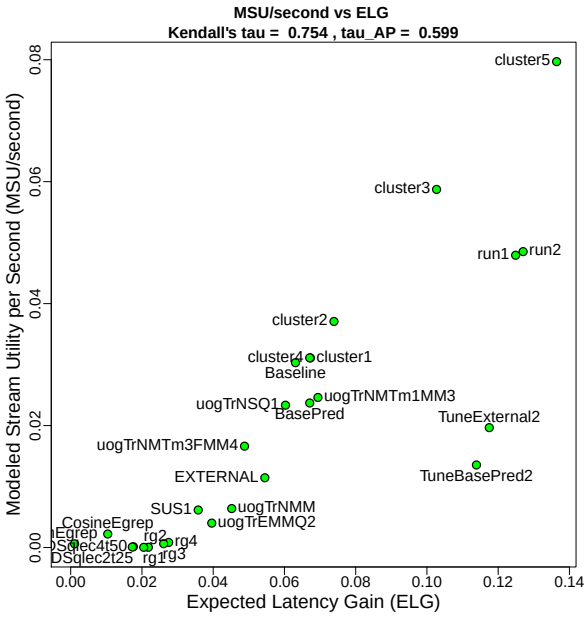
This experiment outlines the necessity of including user behaviour models in the design and evaluation of update systems. As Figure 5.4a shows, measuring relative performance of systems by ELG seems is more suited to users who read updates for 1 minute, once per day on average, at least for the participating systems at TST 2013. This short session duration is evocative of simply reading a daily newspaper's headlines (or lesser). Indeed, the highest gain achieved in Figure 5.4a is by `cluster2` (MSU of just 4.5306), over a 10 day query duration, or 1 relevant fact every 2 days.

5.3.3 MSU and Set-Oriented Metrics

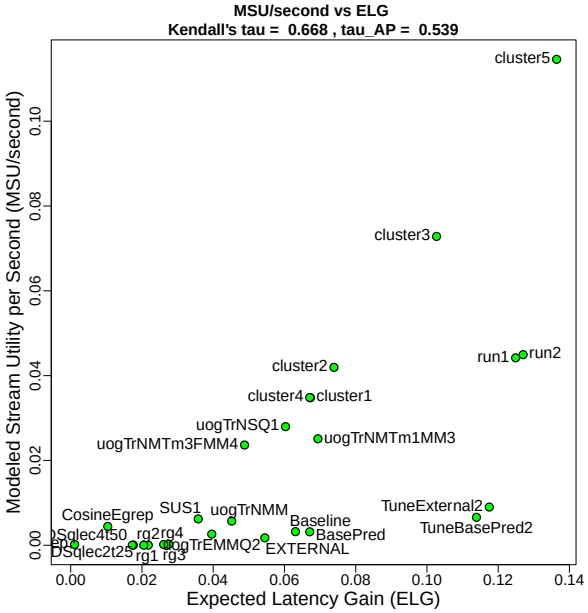
We can divide MSU by the number of updates read, or the number of updates in the run, or the time spent reading. The MSU/sec and $MSU/updateRead$ in particular, are similar to TST's ELG measure. Since MSU depends on the time a user spends reading, we report here our analysis for the MSU/sec measure.

In our experiments, we found that MSU/sec has the highest correlation (Kendall's τ = 0.7538) with ELG (Figure 5.5a), when users simulated with parameters (M_A= 24 hours, S_A= 12 hours, M_D= 1 minute,S_D= 1 minutes, L = 0.1). However, plain MSU with the same parameter settings ranks `cluster5` at position 10. The run `cluster5` scores high on MSU/sec primarily because it has a very low number of updates submitted (198 updates in total). For our simulated users, the sessions in the reading-traces over this run finish earlier than usual because there is less (or no) material to read.

Figures 5.4b, 5.5b exemplify the problem with using set-oriented metrics for evaluation of streams. Performance of users simulated with parameters (M_A= 5 minutes, S_A= 10



(a) Maximum correlation of MSU/sec with ELG.



(b) High correlation of MSU/sec with ELG, even when the model parameters result in minimum correlation of MSU with ELG (Figure 5.4b).

Figure 5.5: Correlations MSU/sec is high regardless of total gain.

minutes, M_D= 30 minutes, S_D= 15 minutes, $L = 1$) has the lowest correlation (Kendall's τ = -0.04) for MSU vs ELG (Figure 5.4b), however MSU/sec remains correlated (Kendall's $\tau = 0.6677$) with ELG. Consider a highly contrived case: a system, returning only 1 update which is relevant, would score very high on MSU/sec, however, the absolute gain would be extremely low (i.e. 1 nugget).

In reality, TST topics have upwards of 50 nuggets and while metrics like ELG (approximating gain per update) and MSU/sec can identify systems having high precision, these systems do not correspond to all user behaviours. The disparity in time available to read and the actual amount of material available to read is more pronounced with users who are desirous of spending more time with the system reading updates. Set oriented metrics fail to measure user performance adequately for such scenarios.

5.3.4 Everyone's a Winner

The disparity between the maximum and minimum Kendall's τ of MSU with ELG (Figure 5.4) indicates that there could be specific kinds of user behaviour for which a particular system (run) might be most suited. We therefore find instances in our parameter sweep for which a particular system was ranked the highest across all parameter sets. We also list the performance of the system at that parameter setting. Some systems achieved their best rank for multiple parameter sets. In such cases, we chose the parameter set for which the system had the highest MSU. Table 5.2 lists the highest rank achieved by each system. The systems are ordered by their ELG rank order as in Table 5.1.

The relationship between time spent and user performance can be seen as we look

run (ELG rank order)	Best Rank	MSU	M_a	S_a	M_s	S_s	L
cluster5	8	4.06	1 day	12 hrs	30 sec	30 sec	0.9
run2	1	9.27	1 day	12 hrs	5 min	2.5 min	0.1
run1	1	14.65	1 day	12 hrs	30 min	15 min	0.9
TuneExternal2	13	9.77	3 hrs	1.5 hrs	30 min	15 min	0.9
TuneBasePred2	11	9.89	1 hrs	30 min	15 min	7.5 min	0.9
cluster3	5	5.83	1 day	12 hrs	30 sec	15 sec	1.0
cluster2	1	12.02	3 hrs	1.5 hrs	30 sec	30 sec	1.0
uogTrNMTm1MM3	6	12.79	1 hrs	30 min	30 sec	15 sec	1.0
cluster1	1	16.45	5 min	10 min	30 sec	30 sec	1.0
cluster4	1	16.15	10 min	20 min	30 sec	30 sec	1.0
BasePred	1	14.17	6 hrs	6 hrs	30 min	15 min	0.9
Baseline	1	15.02	30 min	15 min	30 min	15 min	0.9
uogTrNSQ1	4	11.35	3 hrs	3 hrs	30 sec	15 sec	1.0
EXTERNAL	1	20.38	30 min	1 hrs	30 min	15 min	1.0
uogTrNMTm3FMM4	5	10.52	3 hrs	1.5 hrs	30 sec	15 sec	1.0
uogTrNMM	1	19.07	30 min	15 min	15 min	7.5 min	1.0
uogTrEMMQ2	2	18.55	30 min	15 min	15 min	15 min	1.0
SUS1	17	7.44	1 hrs	1 hrs	2 min	60 sec	1.0
rg4	1	22.11	5 min	10 min	30 min	15 min	1.0
rg3	2	21.54	5 min	5 min	30 min	15 min	1.0
rg2	18	14.02	5 min	10 min	30 min	15 min	1.0
rg1	19	13.39	5 min	10 min	30 min	15 min	1.0
UWMDSqlec4t50	6	12.20	5 min	5 min	30 min	15 min	0.9
UWMDSqlec2t25	14	15.87	5 min	10 min	30 min	15 min	1.0
CosineEgrep	19	0.30	1 day	12 hrs	60 sec	30 sec	0.3
NormEgrep	19	0.52	3 hrs	1.5 hrs	30 sec	15 sec	0.5

Table 5.2: Parameter sets that resulted in Best Ranks for respective systems.

at the parameter settings from top to bottom of Table 5.2. Systems that had very few updates submitted, performed well for users who might visit a system about once a day for 30 seconds to 30 minutes on average. `cluster2` seems to be the best performing system for users who return to the system about every 3 hours for 30 seconds on average. As we go lower, we see that spending more time reading (larger M_D) and taking shorter breaks (smaller M_A) improves performance of systems that ranked lower on ELG. These systems are typically those that submitted a large number of updates.

Reading updates once each day for 30 seconds (`cluster5`) is comparable to reading headlines of a daily newspaper. Runs `run1` and `run2` fare much better in MSU than `cluster5`, as their users are willing to spend slightly more time with the system. However checking for updates once a day may only be suitable for users with low to moderate interest in the event, or for if the event evolves with low dynamism (slowly). For rapidly evolving events or more interested users checking back every 3 hours or so for 30 seconds, `cluster2` seems to work best.

For users who check back with very high frequency `cluster1` and `cluster4` perform best. However, the average number of updates submitted by both these runs are 164.8 and 163 respectively, over a period of 10 days (or about 16 updates per day). This means that for some sessions, when returning every 5 minutes for 30 seconds on average, there may be no updates to read. However, it seems that the updates are of high quality, and since the users are forgiving of lateness (L=1), they derive the highest gain from these systems.

Finally, for the users who want to gain as much information as possible, systems like `BasePred` (reading for 30 minutes every 6 hours), `EXTERNAL` (reading for 30 minutes with

30 minute breaks), `uogTrNMM` (reading for 15 minutes with 30 minute breaks) and `rg4` (reading for 30 minutes taking 5 minute breaks), result in the best user performance. It is unreasonable to assume that a single user may spend 30 minutes reading with 5 minute breaks over a period of 10 days. However the MSU is maximum for `rg4` and it may be possible for it to be deployed where teams of people are monitoring streams. A team of users (as in government, aid providers or monitoring organizations) may monitor the stream on a continuous basis and derive the maximum possible gain.

5.4 Discussion

The TST evaluation measures seem to be geared for evaluating systems that "push" the most relevant updates to the user, i.e., the system decides the amount of information to be presented via the updates. The intrinsic assumption is that the users would read every update emitted by the system, necessitating the system to emit fewer but high quality updates. This leads to measures that are analogous to Precision (ELG) and Recall (LC) that measure the quality of a set of returned updates. However, for a long running evolving event, it is hard to know how much relevant information exists or how much new information would be generated in the future. In case of TST 2013, the number of nuggets varies across topics from 37 to 418 (Table 5.3). Emitting updates containing multiple nuggets would be optimal however, the (anxious) user may find the wait frustrating, whereas the time constrained user would find it extremely convenient. That is to say that, a user actively seeking updates, may gain more in reading more number of updates, rather than passively waiting for updates to be pushed by the system.

topic	#nuggets identified by assessors	#nuggets retrieved by systems	nugget recall
1	56	45	0.80
2	89	47	0.53
3	139	75	0.54
4	97	55	0.57
5	108	41	0.38
6	418	106	0.25
8	88	58	0.66
9	45	29	0.64
10	37	28	0.76
Total	1077	484	–
Average	119.67	53.78	0.57

Table 5.3: Nuggets identified, nuggets retrieved by all systems, and the nugget recall per topic. Note that no single system retrieves all nuggets for a topic.

We therefore propose a model of evaluation in which the user behaviour drives the amount of gain that is possible to achieve. A user more interested in the event could/should possibly derive more gain when using a system that generates a stream of updates. The more the content presented to the user, the more time is required to read it. Thus, the notion of verbosity penalization (of ELG) is essentially subsumed by the time spent reading sentences. Moreover, if users read longer updates, they might experience lesser gain over all as they are likely to miss reading other potentially relevant updates.

A key advantage of the MSU evaluation model is that it is possible to calibrate (and re-calibrate) it by observing real user behavior. ELG, on the other hand, does not provide an easy means to be calibrated to known user behavior. Our experiments and analysis show that what matters is the amount of material read. By specifying the amount of material in user terms, we have a way of then calibrating a measure once we know actual

user behavior. Observing actual user behaviour while evolving events are actually taking place would involve observing users in a live setting. The sudden nature of news events and the variable length of evolving events, makes a user study difficult to organize. Search log-analysis may provide some indirect insight into user behaviour when such events are running.

Our analysis also demonstrates that there is a case for personalization of stream filtering systems for different user behaviours. Alternatively application-specific (or latency specific) system development would require an evaluation calibrated for the specific task.

5.4.1 Evaluating Runs with MSU using qrels Expanded with Duplicates

In this section we look at how the duplicates expanded qrels affect evaluation of runs using MSU. We simulate the "reasonable users" (Section 5.3.1) and evaluate the runs submitted to TST 2013 and TST 2014 with MSU. We find that the relative performance of the systems does not change significantly as measured by MSU (Table 5.4). However 13 (of 26) and 17 (of 24) runs show statistically significant changes in MSU scores for TST 2013 and TST 2014 respectively.

Figure 5.6 shows the correlation plot for the MSU scores for TST 2013 runs (duplicates expanded qrels vs standard qrels). 15 runs experience an increase in MSU with 13 showing a statistically significant change in MSU. In comparison, for TST 2014 (Figure 5.7), all 24 submitted runs experience an increase in MSU with 17 runs showing a statistically significant change. The MSU score increases with the inclusion of duplicates in the qrels because

MSU evaluation for track	Kendall's τ	#runs with stat. Sig. difference in score
TST 2013	0.914	13
TST 2014	0.927	17

Table 5.4: MSU evaluation of runs for TST 2013, TST 2014; respective rank correlations between standard and duplicate expanded qrels; number of runs with statistically significant (p-value ≤ 0.05 over a paired t-test) changes in MSU scores.

some duplicates have an earlier timestamp than the sentences present in the original pool. This reduces the effect of the late reporting penalty for each nugget (Section 5.2.1), as relevant sentences are available to users in earlier user sessions rather than later ones.

Unlike ELG or LC, all the sentences are not assumed to be read by MSU, i.e., MSU evaluates runs based on a subset of the submitted sentences that are read by simulated users. Adding duplicates to the qrels helps MSU in producing a fairer evaluation because the expanded qrels increase the likelihood of a relevant (judged or duplicate) sentence to be present in the subset of submitted sentences that are read by the simulated users.

5.5 Conclusion

We introduce an evaluation measure that integrates a model of user behaviour for evaluating streams of filtered information. Our user model simulates a user checking back with the system to read latest information from time to time. Users can check back with different frequencies and for different amounts of time depending on various factors. We also find, that for streams of updates, the gain is sensitive to the amount of content consumed by the user. Our evaluation model is flexible, in that, it allows for evaluating stream generating

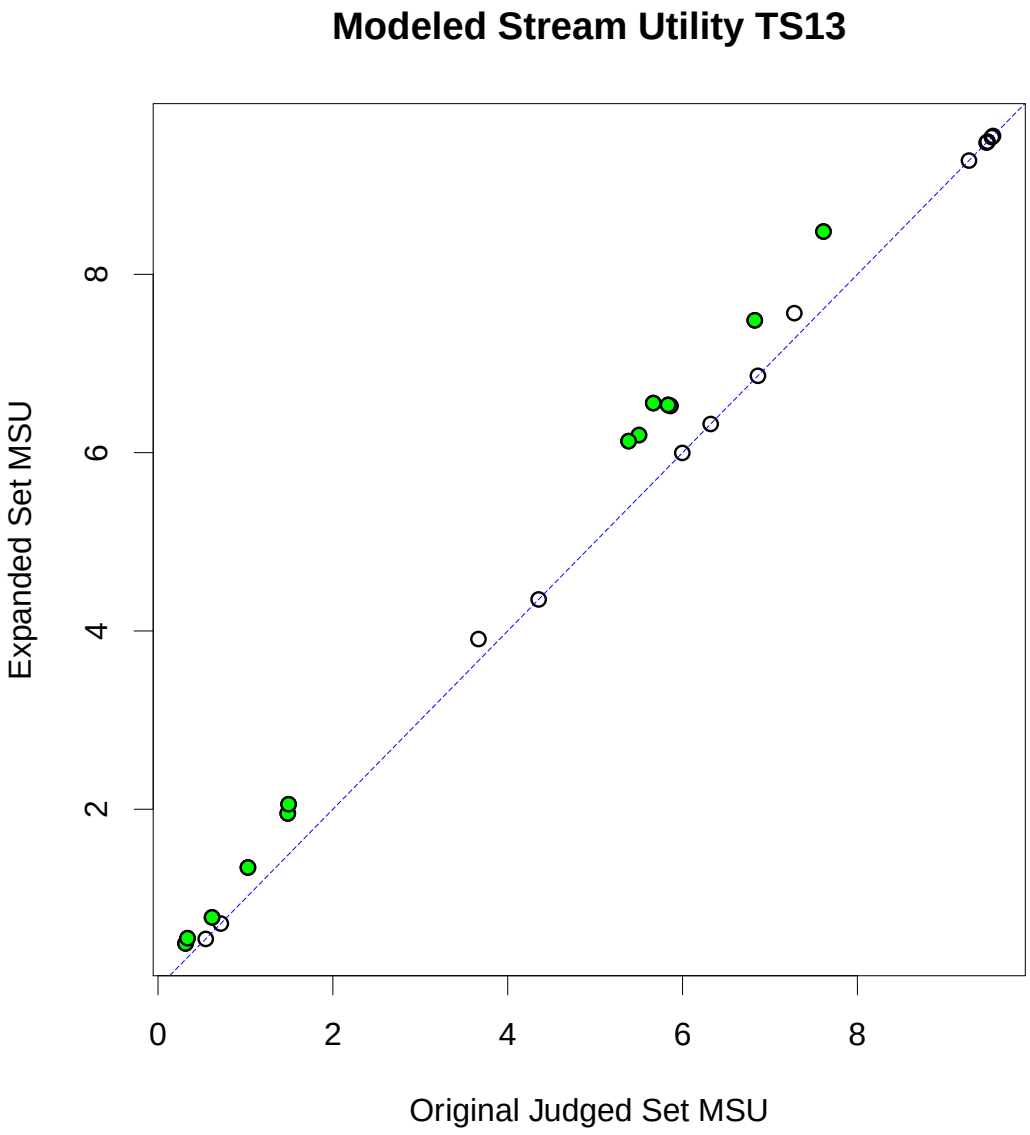


Figure 5.6: TST 2013 MSU scores for the systems using the duplicates-expanded set of judged sentences vs. the MSU scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute MSU score for the respective run.

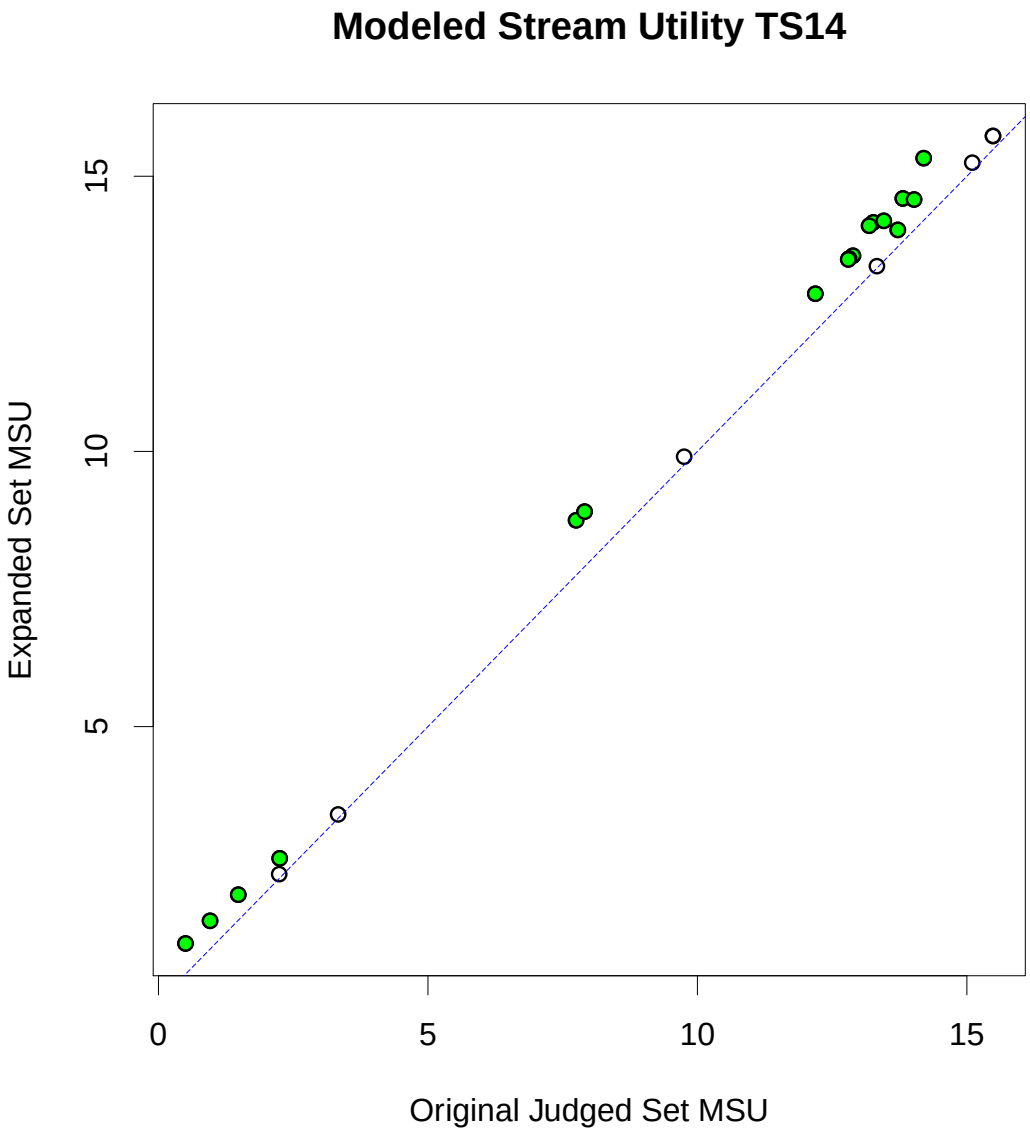


Figure 5.7: TST 2014 MSU scores for the systems using duplicates-expanded set of judged sentences vs. the MSU scores for the systems using the original set. The original (track's) system rank order is from right-to-left on the X-axis. The green colored points indicate a statistically significant (p-value ≤ 0.05 over a paired t-test) difference in absolute MSU score for the respective run.

systems for various user behaviours. A user model oriented evaluation provides a way to calibrate metrics to effectively measure performance of stream filtering systems.

Chapter 6

Simulation-based Pooling

Given a model of user behavior, on simulation, it is possible to note which updates (sentences) are read by each simulated user. Given a user population, one can compute the likelihood of an update being read, for a sampled user from the population. This chapter presents an alternative pooling methodology that utilizes the probability that an update is read by a user, to select updates into the evaluation pool. The resulting *probability-based* pools (Baruah, Roegiest, and Smucker, 2015) differ considerably from the pools constructed using the TREC standard depth-pooling method (Voorhees, Harman, et al., 2005). We present the intuition behind the probability based pooling idea in Section 6.1. We discuss methods of estimating the probability that an update is read by a user ($P(read)$) in Section 6.2, depth-pooling based on $P(read)$ in Section 6.3, and pooling based on probability mass cover in Section 6.4.

6.1 Motivation for Simulation-based Pooling

Under the MSU user model, for any given system, some updates may be read by more users than others in the period of interest (query duration). Figure 6.1 illustrates this effect for the run `cluster1`; it shows the proportion of simulated users that read an update emitted by the system at a given time. We simulated 100,000 users with the reasonable users parameterization of the MSU user model (Section 5.3.1). The X-axis in Figure 6.1 spans the 10 day query duration; the specified period of interest for each topic in the temporal summarization track (TST) 2013 spanned 10 days (Section 3.1). The Y-axis shows the proportion of the 100,000 simulated users that read each update.

As can be seen from Figure 6.1, most updates emitted by `cluster1` are read by most of the simulated users. Very few updates are read by a lower proportion of users; these are updates that are read by between 20% to 40% of the users near the start of the query duration. However, for other systems, depending on the number of updates and the times at which the updates are emitted, the proportion of users reading each update differs. For instance, Figure 6.2 shows that most updates are not read by a large proportion of simulated users for the run `TuneExternal2`; very few relevant updates are read by a higher proportion of users in comparison to `cluster1`.

The numbers of relevant updates read by proportions of users varies considerably across submitted runs; in general we observed that very few of the most frequently read updates are relevant. Since identified (judged) relevant updates are read by fewer users, estimation of system performance may be inaccurate when evaluating with MSU. Ideally, runs should have relevant updates read by a larger proportion of the user population; conversely, the

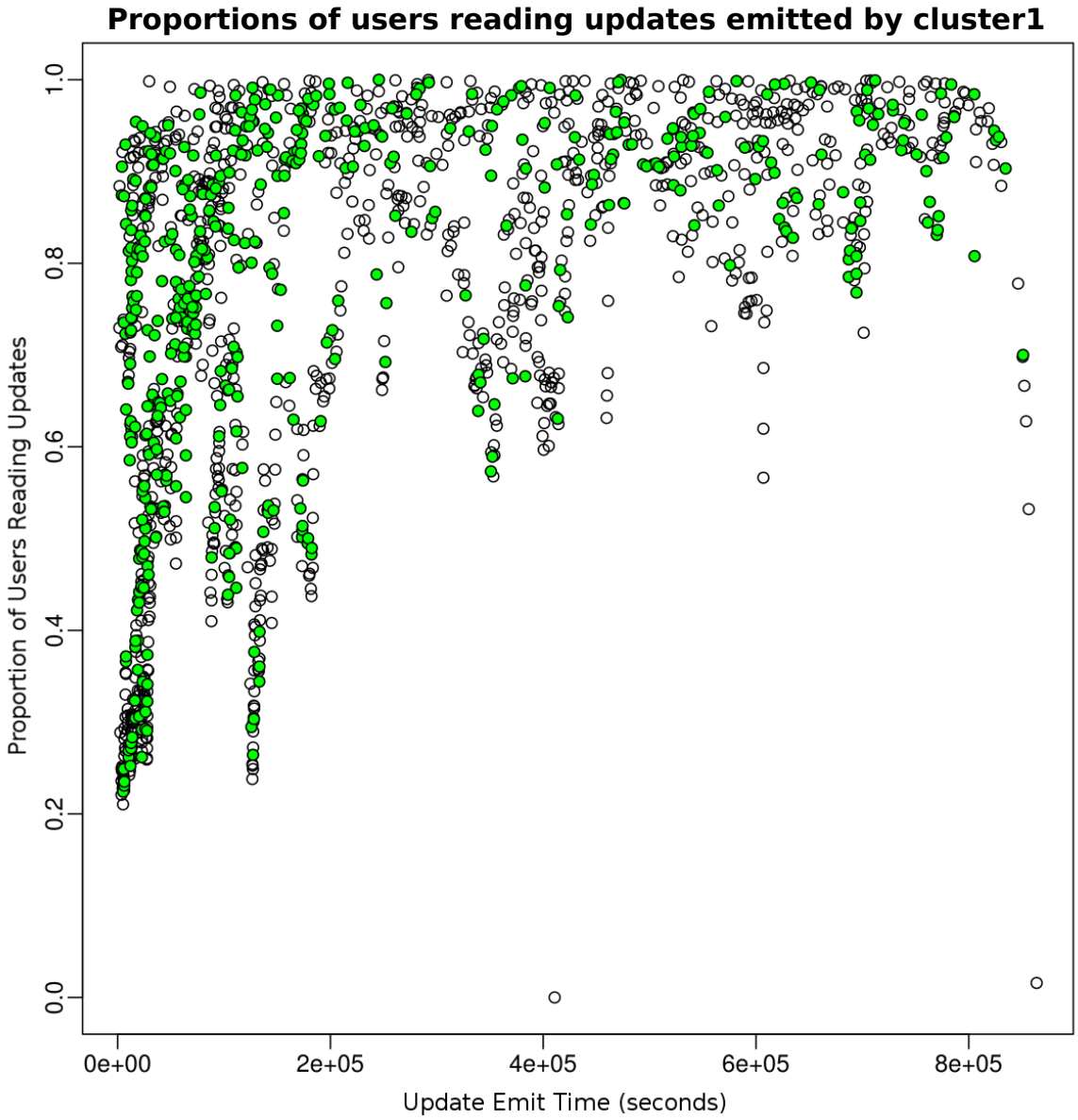


Figure 6.1: Proportions of 100,000 simulated users reading updates emitted by the run `cluster1`. The run returned 1,483 updates across all topics of which 497 were relevant; relevant updates are indicated as green dots in the figure.

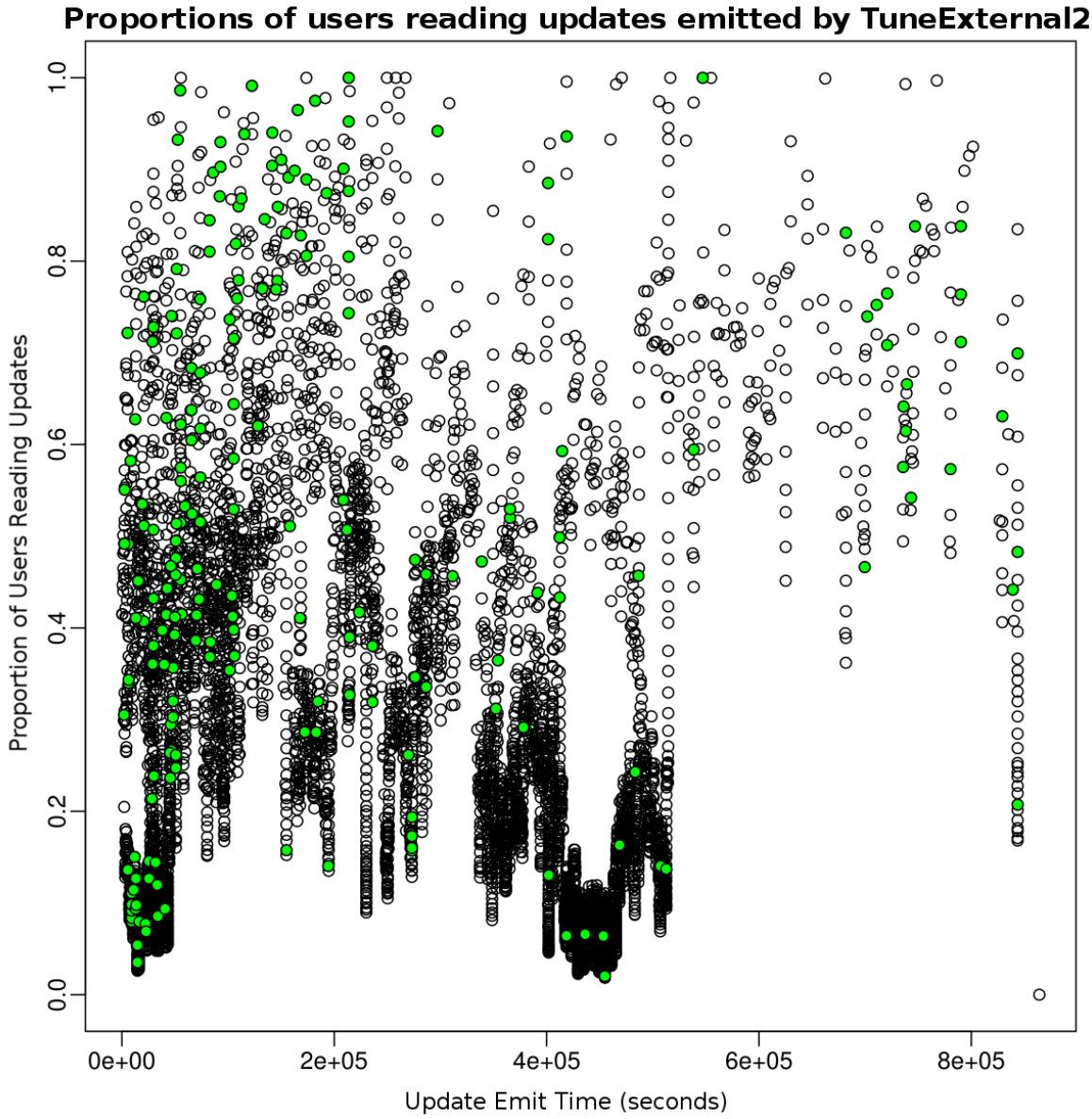


Figure 6.2: Proportions of 100,000 simulated users reading updates emitted by the run `TuneExternal2`. `TuneExternal2` returned 7,195 updates across all topics of which 214 were relevant.

updates that are read more often could be assessed for relevancy in order to better estimate system performance. Accordingly, updates that are read more frequently by users could better serve to evaluate systems using MSU.

6.1.1 Proportions of Users not Reading Relevant Updates

If we look at all the updates submitted by participating runs for a particular topic (Figure 6.3), we see that under the MSU user model for reasonable users, fewer relevant updates are read by higher proportions of users, and many relevant updates are read by a lower proportion of users . Other topics in the track show a similar spread of (relevant) updates read by various proportions of users and we find that in general, many identified relevant updates were read by fewer users. It could be the case that the subset of updates that are read by a greater proportion of the simulated users could contain un-pooled and thereby unjudged relevant updates.

TST 2013 utilized a variant of standard pooling (Aslam et al., 2013) wherein the top 60 updates, as determined by system assigned scores, were pooled together to create the evaluation pool. However, most relevant updates, as identified by the NIST assessors, are not read by modeled users when systems are evaluated using MSU. Figure 6.4 presents the histogram of relevant updates read by proportions of simulated users; it is an alternative time agnostic visualization for the data presented in Figure 6.3. Of the 1616 relevant updates returned across all runs for topic 10: only 5 were read by all modeled users; 25 were read by more than 99% of the users; and 546 relevant updates were read by less than 1% of the users. However, as Figure 6.3 shows, there are many updates that are read by

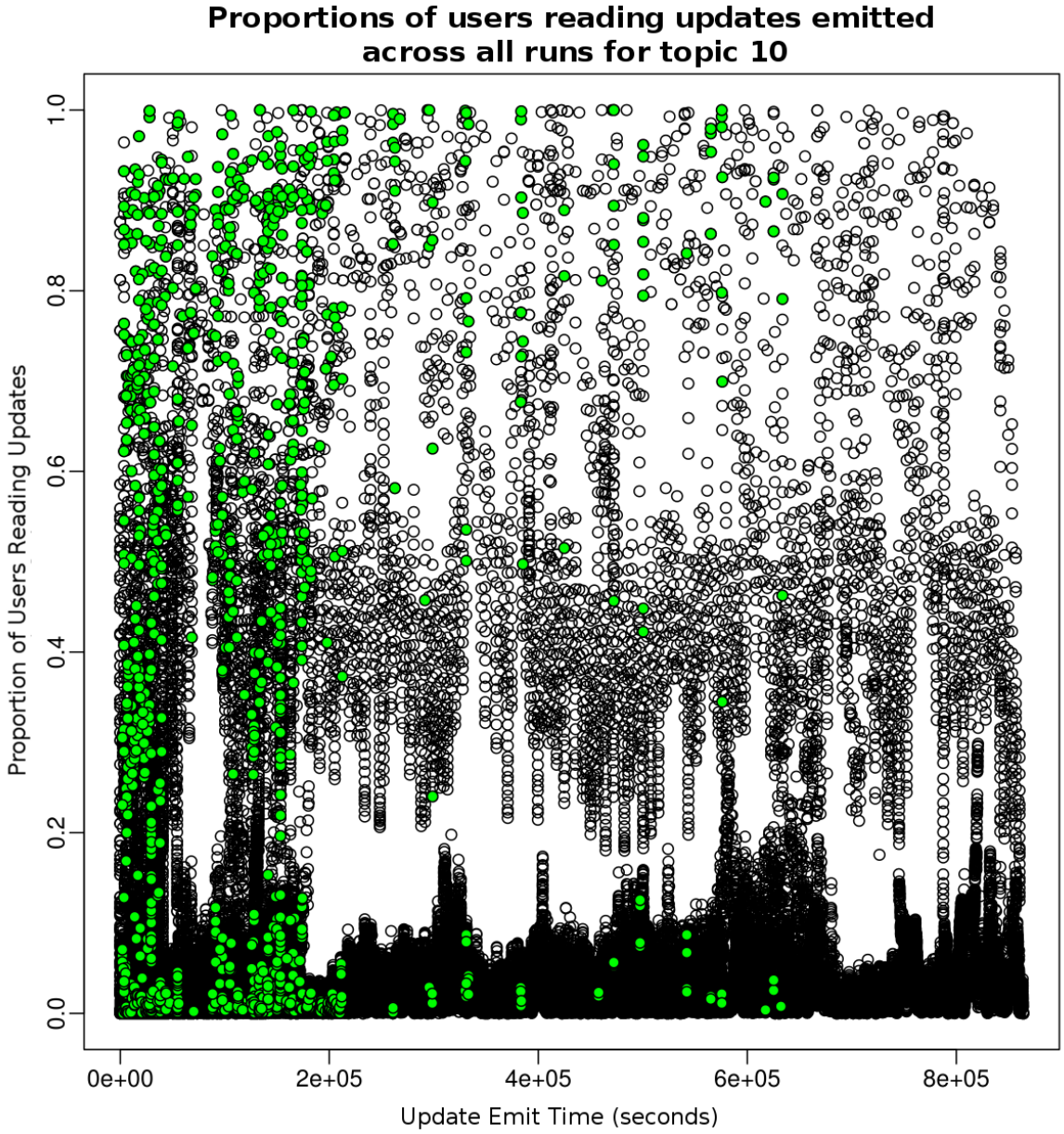


Figure 6.3: Proportions of 100,000 simulated users reading all updates submitted across all runs for topic 10 of TST 2013. Across all runs 418,332 updates were submitted for topic 10 of TST 2013; 1,616 relevant updates were returned.

greater than 90% of the users. Some of these updates are unjudged and they could contain some number of relevant updates.

All other topics for the track have a similar distribution of relevant updates being read by proportions of users. It may thus be beneficial to identify relevant updates within the set of updates read more frequently by users when evaluating systems with MSU. This leads us to the intuition that each submitted update has a probability of being read; in other words, there could be a probability distribution over the updates indicating the likelihood of a given update being read, given the modeled user population. Such a distribution may allow us to pool updates that have higher probabilities of being read for constructing a judged set of updates. A judged set thus constructed may be more suited for evaluation using MSU.

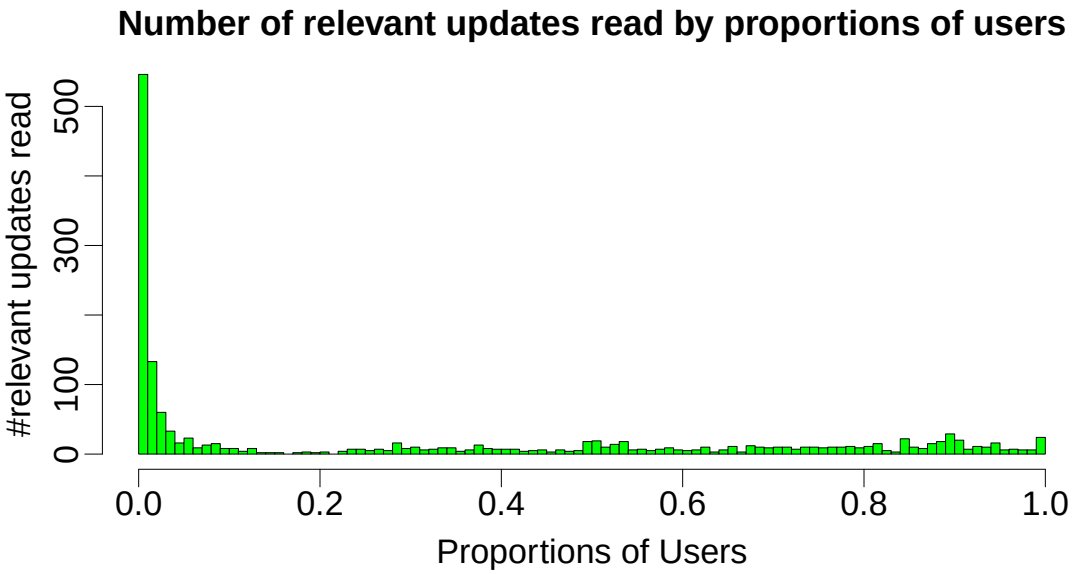


Figure 6.4: The number of relevant updates read by proportions of simulated users, for topic 10 from the temporal summarization track, at TREC 2013.

6.2 Estimating the Probability of an Update being Read

Computing the probability of an update being read involves simulating users reading updates from a system. Assuming that the user population follows MSU's "reasonable" user population model (Section 5.3.1), 10,000 users were simulated from a population of users that had a mean time spent reading of 2 ± 1 minutes and a mean time spent away of 3 ± 1.5 hours. Given the set $U = \{u_i | 1 \leq i \leq m\}$ of m simulated users, and the set $D = \{d_j | 1 \leq j \leq n\}$ of n updates returned by a system, then

$$read(i,j) = \begin{cases} 1, & \text{if user } u_i \text{ reads update } d_j. \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

indicates if a given user read a particular update. We can further imagine a matrix for the $read(i,j)$ indicator variable with rows corresponding to users and columns corresponding to updates returned by a system (Figure 6.5). The $read(i,j)$ matrix allows us to visualize the computations of different kinds of probabilities of updates being read.

For instance, one may estimate the probability of an update d_j being read ($P(d_j)$) as the probability that a randomly selected user would read update d_j. This probability corresponds to the likelihood of ones in each column of Figure 6.5;

$$P(d_j) = \frac{1}{|U|} \sum_i read(i,j), (1 \leq i \leq m) \tag{6.2}$$

	d_j							
	0	0	1	1	0	0	...	0
	0	1	0	0	1	0	...	0
	1	1	$\ddots$					
u_i	0	1						
	1	0						
	$\vdots$	$\vdots$						
	0	1						

Figure 6.5: An example of a $read(i, j)$ matrix given users u_i and updates d_j.

This formulation is essentially the *proportion of users* that read a given update d_j. In fact, the Y-axis in Figures 6.1, 6.2 and 6.3 represents this proportion when $|U|$ is 100,000, under the MSU reasonable user model.

Equation 6.2 considers each update in isolation and thus the probabilities of being read are specific to individual updates. Under the MSU user model, each simulated user, either reads an update or does not read it. The simulation allows us to record exactly which updates are read and which are not. This read or not read knowledge about updates allows us to build a probability distribution over the set of updates returned by a system, where the distribution indicates the likelihood of an update being read. Updates not read by the user have a probability mass of zero; whereas updates read by the user have a non zero probability of being read. This non zero probability is the same for every update read by the user and is the inverse of the number of updates read by the user because under the MSU user model each update is read at most once. For instance, consider the first row of Figure 6.5. The user represented by the first row reads only 2 updates; i.e., the probability mass for each of these read updates is 1/2 and the remaining updates have a probability

mass of zero. The probability of an update d_j being read can now be estimated as[1]

$$P(d_j) = \frac{1}{|U|} \sum_i \frac{read(i,j)}{\sum_q read(i,q)}, (1 \leq q \leq n, 1 \leq i \leq m) \tag{6.3}$$

Here, $P(d_j)$ estimates the probability that given a randomly selected user u_i, what is the likelihood that d_j is an update read by u_i. In terms of Figure 6.5, Equation 6.3 corresponds to the random experiment: first pick a row at random, and then pick a 1 at random from the row; what is the likelihood that the picked 1 was from column j. We refer to this formulation of $P(d_j)$ as being *balanced*; $P(d_j)$ is computed as an average over all users' probability distributions of reading d_j, i.e., the contribution for probability estimation is balanced across each simulated user in Equation 6.3.

An alternative (and perhaps more intuitive) formulation of $P(d_j)$ is

$$P(d_j) = \frac{\sum_i read(i,j)}{\sum_i \sum_q read(i,q)}, (1 \leq q \leq n, 1 \leq i \leq m) \tag{6.4}$$

where $P(d_j)$ is the probability that an update is read given all the updates that were read by all simulated users. In terms of Figure 6.5, Equation 6.4 corresponds to the random experiment: pick a 1 at random from the matrix; what is the likelihood that the picked 1 is from column j. This formulation also results in a probability distribution over updates; a distribution that is reflective of the number of ones in each column of Figure 6.5. In Equation 6.4, $P(d_j)$ is computed as the ratio of number of times d_j is read to the number

[1]NOTE: Equation 6.3 is the corrected formulation for Equation (1) as defined in Baruah, Roegiest, and Smucker (2015).

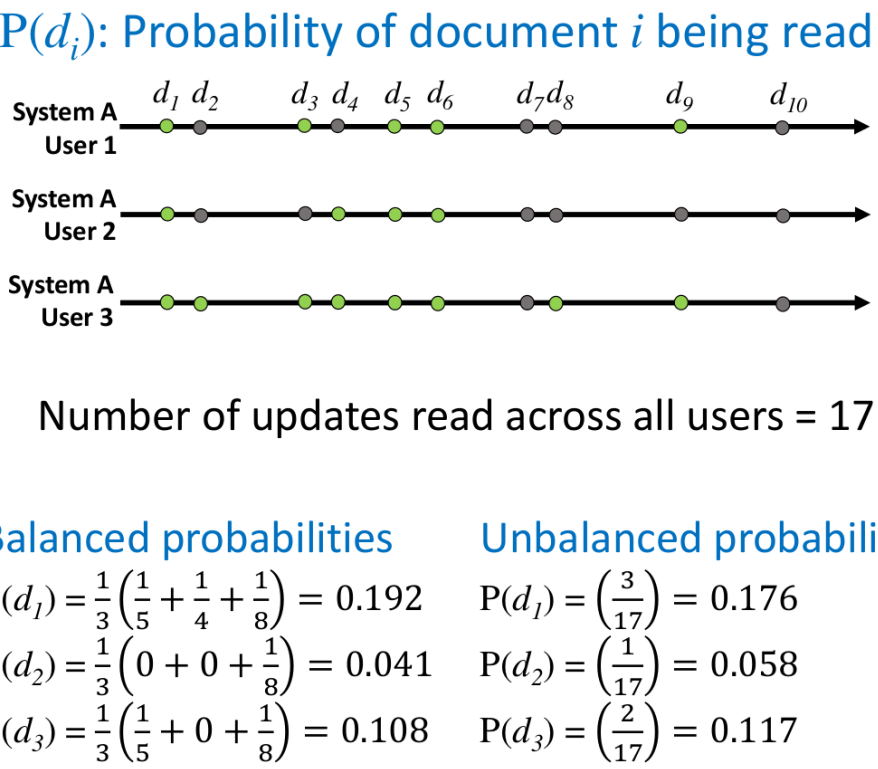


Figure 6.6: Example illustrating the difference between *balanced* (Equation 6.3) and *unbalanced* (Equation 6.4) formulations of $P(read)$. Green colored dots represent updates read by various users over the given System A. Updates d_2 and d_3 are awarded higher probabilities by the unbalanced formulation than the balanced formulation.

of times *all* updates are read, across all users. This formulation favors updates read by users that read more, and we term it as *unbalanced* (in contrast to Equation 6.3 which is balanced across users).

Figure 6.6 shows the difference in the two methods of computing $P(d_j)$. As can be seen, the *unbalanced* (Equation 6.4) formulation distributes a higher amount of probability mass to updates that are read less frequently than does the *balanced* (Equation 6.3) formulation. Users that read more number of updates, read some updates that are read less frequently by

other users. The unbalanced formulation awards a higher $P(read)$ to such less frequently read updates, and thus, Equation 6.4 increases the likelihood of less frequently read updates to be included into a probability-based pool, thus favoring users that read more updates in general.

6.2.1 Depth Pools with Balanced and Unbalanced Probabilities

Balanced and unbalanced probabilities were estimated for each update of each run. Updates were then ordered by their probabilities. At various depths (top-k probabilities), pools were constructed by selecting the most probably read updates as ordered by the balanced probabilities. Similarly pools were also constructed by using the unbalanced probabilities at various depths of k. Figure 6.7 shows that there is minimal difference in terms of overlap between the pools created using balanced probabilities and the pools created using unbalanced probabilities, even at a pool depth of 1000.

Thus, even though the absolute values of the probabilities may differ, most updates are common to both types of pools. Therefore, for our remaining experiments, we utilized the unbalanced probabilities for constructing pools because of the simple underlying theory and ease of implementation.

6.3 Score vs. Probability Based Pooling

In standard depth-pooling, documents are ordered by the system assigned scores for each document, and the top-k documents are selected into the pool. Probability based pooling

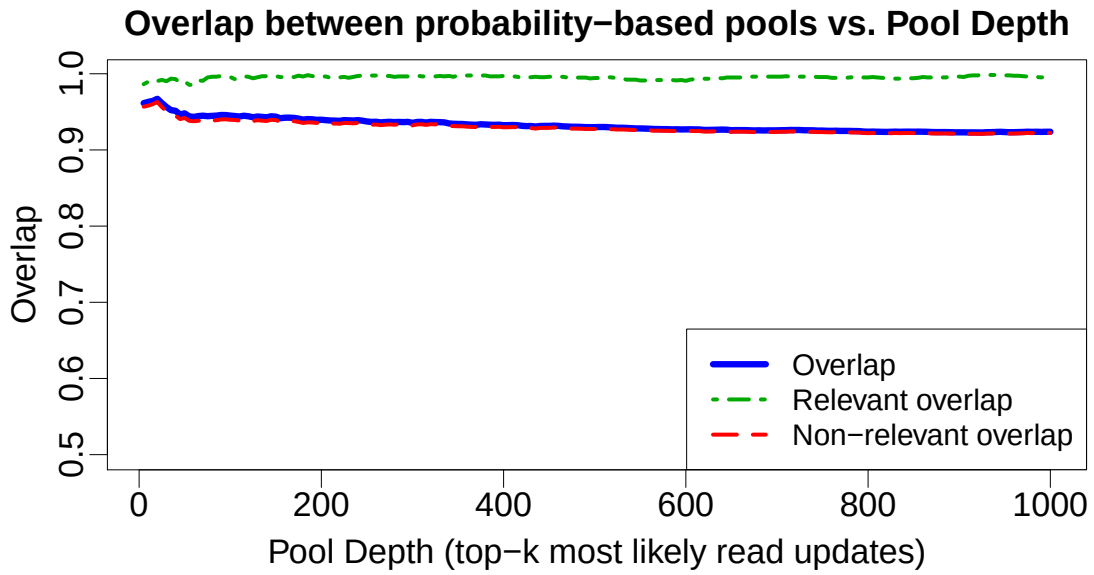


Figure 6.7: Overlap between the top-k probability-based pools created with balanced and unbalanced probabilities.

utilizes the probabilities of updates being read ($P(read)$) as the pool membership criterion, rather than the system assigned document score. Updates are thus ordered by $P(read)$, and the top-k updates are selected into the evaluation pool, for probability based pooling.

We vary k from 1 to 1,000 (in steps of 5) and construct probability-based pools as well as score-based pools. Figure 6.8 illustrates the difference between the 2 pooling methods at various pool depths. It can be seen that many updates are not shared between the pooling methods. The overlap remains below 45% between the probability-based pools and the score-based pools constructed using the top-k updates ordered by $P(read)$ and system assigned scores respectively. The overlap of relevant documents remains below 70% even at a pool depth of 1000. Note that the TST uses a pool depth of 60 to construct evaluation pools.

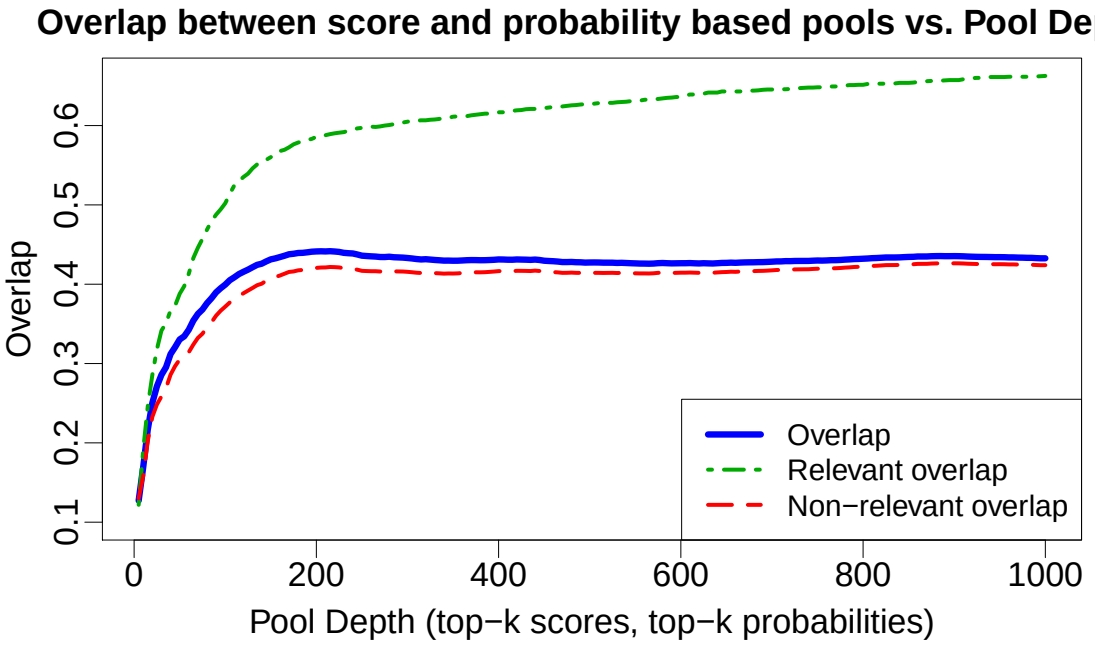


Figure 6.8: Overlap between the top-k score-based and top-k unbalanced probability-based pools.

Although score-based pooling (or depth-pooling) is performed in order to have a better chance of finding relevant documents, perhaps, this method is not entirely suitable for user-model oriented evaluation measures. As was seen in Figure 6.4, very few relevant updates are read by users. Also, since the overlap between the probability-based and score-based pools is low, it is quite likely that the updates in the probability based pools may contain other unseen relevant updates. We leave further investigation to this regard for future work.

6.4 Pooling using Probability Mass Cover

Given probabilities $P(read)$, alternative pooling methods based on probability mass cover can be explored. For instance, assessors may decide to adjudicate updates so that a probability mass of 0.2 is covered for each run. In such a scenario, updates having the highest probabilities of being read would be included into the pool. Additionally, these updates' probabilities would constitute 0.2 of the probability mass of $P(read)$ distributed across all updates in the run. If the updates in a given run are ordered by $P(read)$, then assessors could adjudge updates in order till a 0.2 probability mass cover for $P(read)$ is achieved for the run. Since the probability mass cover is different for each run, and because a designated level of probability mass is covered for each run, we term this method of pooling as *local* probability mass pooling.

Alternatively, the probability mass may be distributed across all the runs, in which case, the pool would consist of updates that have a higher likelihood of being read in multiple systems. In other words, the local probabilities of each update can be averaged across the number of submitted runs. This would result in updates that are common to multiple runs having higher *global* probabilities of being read. Then, to achieve a designated probability mass cover, we order all submitted updates (across all runs) by their global probabilities and adjudicate updates until the designated probability mass is covered. We term this method as *global* probability mass pooling.

Depending on the size of individual runs, and the availability of assessing resources, the amount of work and effort required for *probability mass cover based* pooling would be different. Figure 6.9 shows probability mass covered as a function of pool size for global

as well as local pools. As can be seen, the local probability mass pools require many more updates to cover a specified probability mass, than global probability mass pools.

This large difference in size of the respective pools is due to the large variation in the size (the number of updates submitted by a run) of runs. As Table 3.7 shows, the average number of updates returned per topic varies from 22 to 312,863. Figure 6.10 shows the change in the local probability mass cover as a function of the global probability mass cover, for each run submitted to TST 2013. Lines to the left of the plot indicate probability mass cover for smaller runs, and lines to the right indicate the probability mass cover for longer runs. As can be seen, the local probability mass grows much slower than the global probability mass for larger runs (runs that returned a large number of updates). In other words, updates with a higher probability of being read across all runs contribute more to the global probability mass pooling method in terms of pool membership.

Figure 6.11 depicts the overlap between the local and global probability pools as the desired probability mass cover varies from 0 to 1. The overlap is low for a lower specified probability mass and it grows to 100% as the probability mass cover reaches 1, at which point both pooling methods would include all submitted updates into the pool.

Pooling based on probability mass cover is an interesting alternative to depth pooling based on probabilities of updates being read (cf. 6.3). It allows the judged set of sentences to have increased cover for the subset of updates read by users. However, the utility of probability mass cover based pooling is yet to be investigated and is left as an interesting avenue of future work.

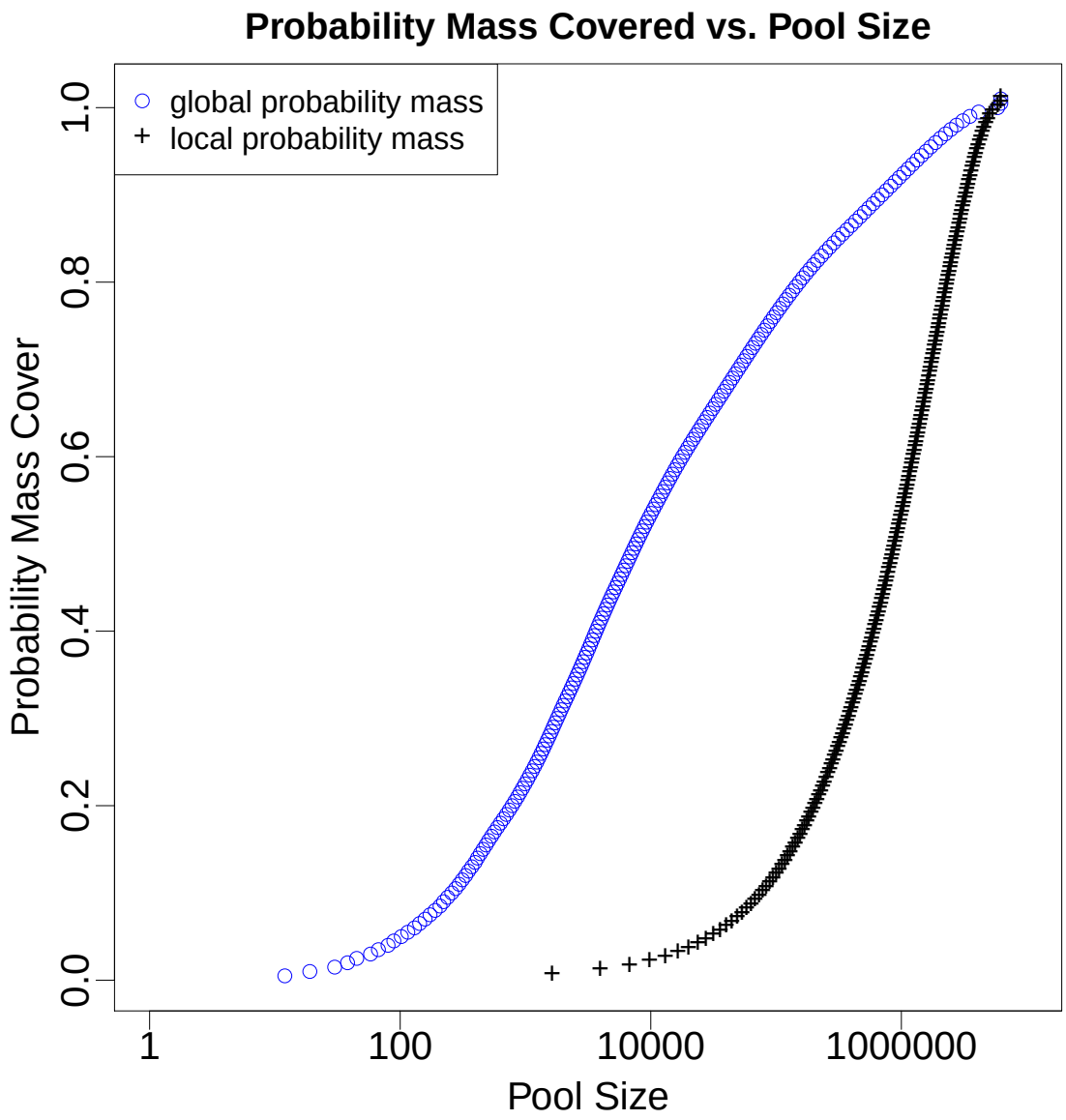


Figure 6.9: Comparison of pool size and probability mass covered for both local and global probability mass pooling strategies.

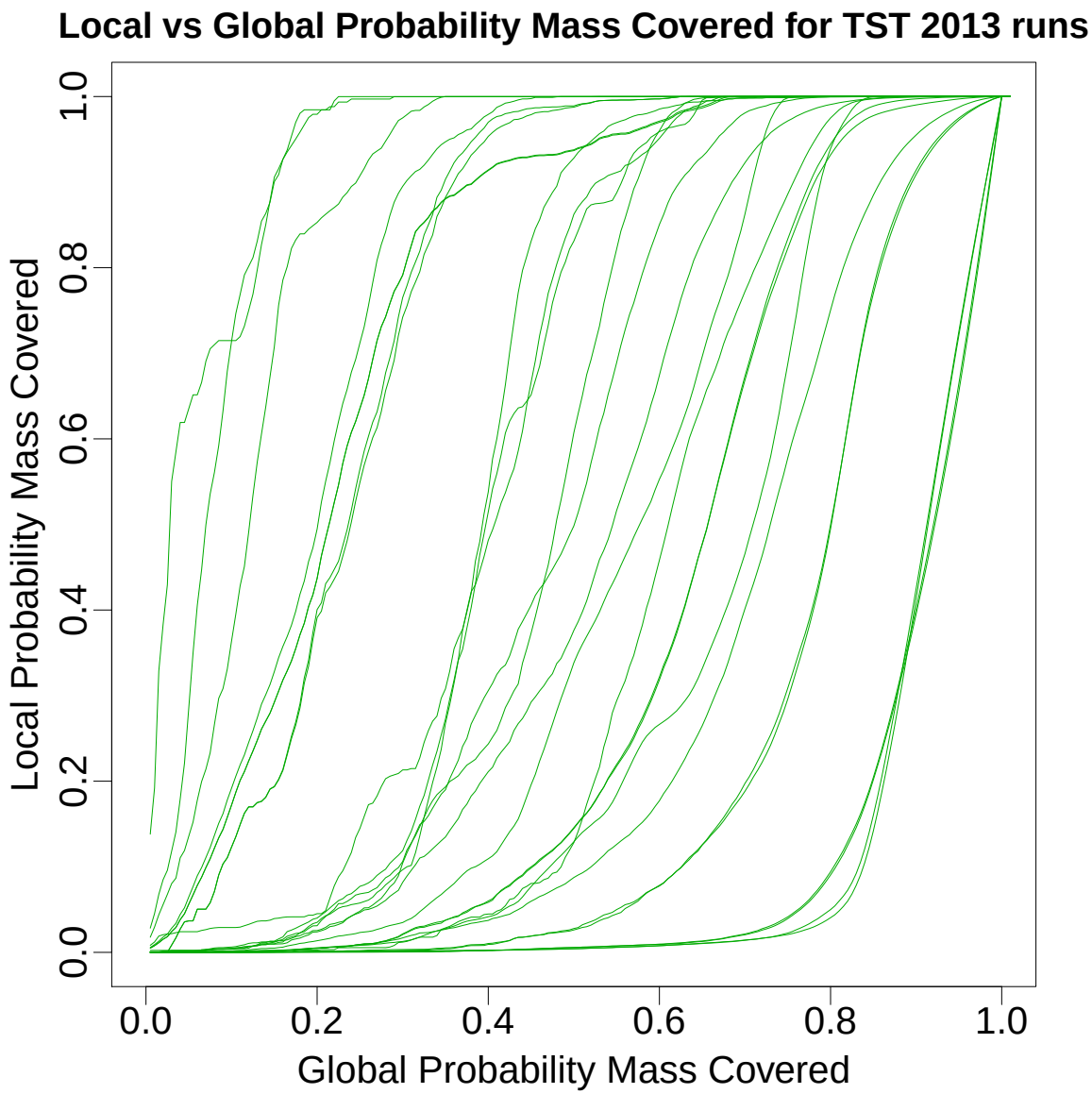


Figure 6.10: Local probability mass covered for all 26 runs submitted to TST 2013, when global probability mass pooling is performed.

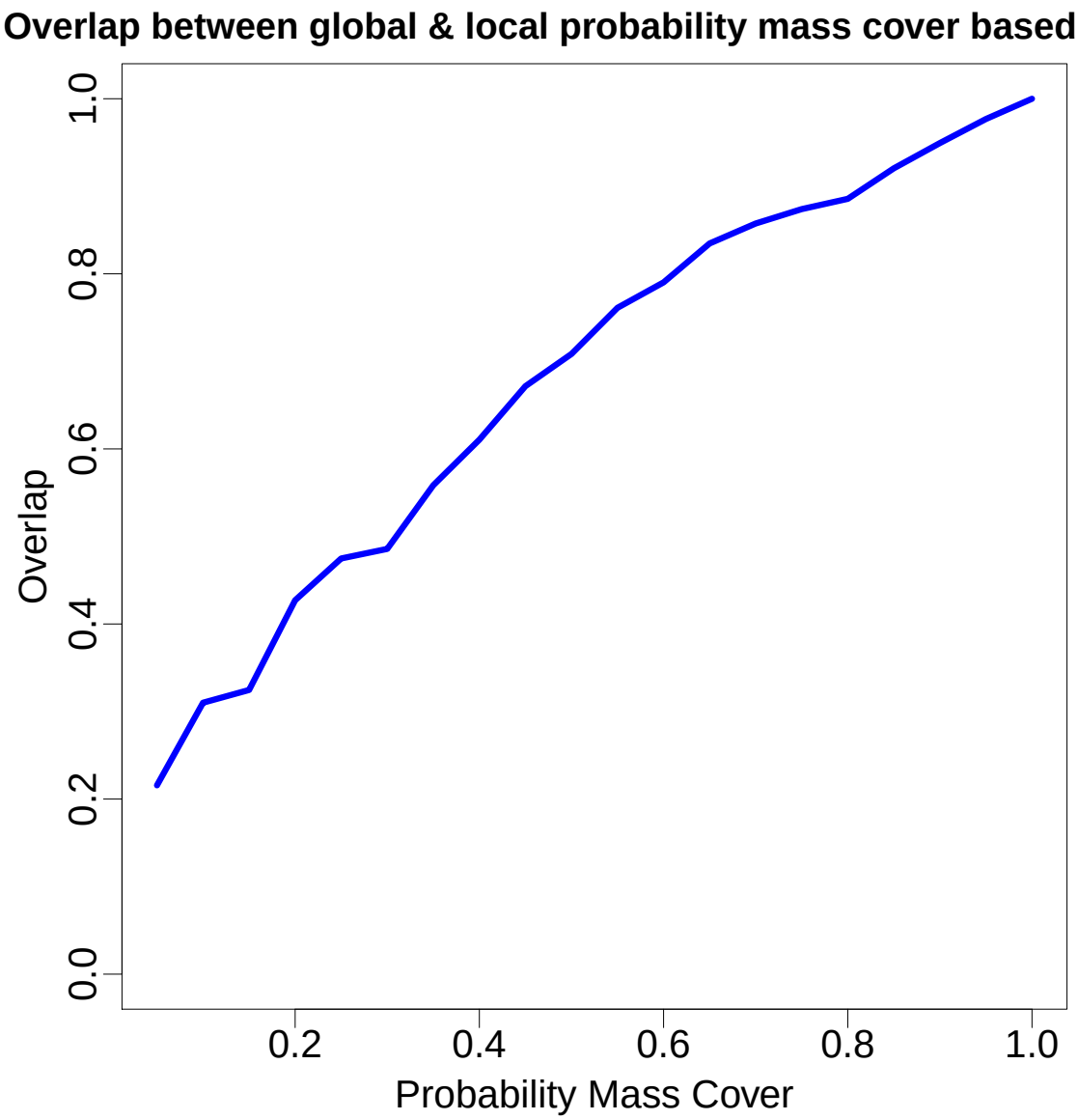


Figure 6.11: Overlap between pools created using global and local probability mass pooling strategies when both target the same probability mass cover.

6.5 Discussion and Future Work

How effective and reusable are probability-based pooling methods as compared to the standard score-based pooling method, is an interesting question for future work. Traditional research on pooling focusses on depth-pooling based on system assigned scores for documents (Cormack, Palmer, and Clarke, 1998; Sparck-Jones and Van Rijsbergen, 1975; Carterette, Allan, and Sitaraman, 2006; Voorhees, Harman, et al., 2005). Furthermore, it is yet unclear how analyses on the quality of pools (Cormack and Lynam, 2007; Buckley et al., 2007) can be applied to probability based pooling methods.

One potential disadvantage of the probability based pools is that, some users may never encounter relevant updates. Since the probabilities are averaged across users and runs, the resultant pools may not contain updates that are read by a very small fraction of users. Some of these updates could be relevant but since they would be absent from the evaluation pool, MSU may report inaccurate values of system performance. One possible solution could be to combine $P(read)$ and the system assigned score for every update to compute a "pool membership" score. A pool can then be formed by using pool membership scores for every update.

Relevance assessments are required for the probability-based pools to enable pertinent effectiveness analyses and further experiments. Any line of future work is hindered considerably by the lack of relevance judgements for the probability based pools. Procuring relevance judgements for the probability based pooling methods is essential for comparing probability based pooling with standard (score-) depth-based pooling.

Chapter 7

Conclusions and Future Work

In this thesis, we mainly looked at how we can evaluate systems that produce a stream of news updates, from a user-oriented perspective. We also explored factors that might affect the evaluation of such systems.

7.1 Summary

Following from the user-model oriented evaluation paradigm (Clarke and Smucker, 2014; Clarke et al., 2013; Smucker and Clarke, 2012d), we developed modeled stream utility (Baruah, Smucker, and Clarke, 2015) (MSU), a user-oriented evaluation measure for the evaluation of news filtering systems. We demonstrated our measure using the participant systems of the temporal summarization track (TST) 2013. MSU differs considerably from ELG and LC, the evaluation measures developed for TST.

We developed a simple user model for the behavior of a user accessing information a stream of updates. Essentially, the MSU user model simulates a user alternating between spending time reading updates, and spending some time away from the system. We simulated users reading content from the stream of updates produced by a system and measured gain for every information nugget contained in relevant updates that were read. Our experiments show that, the performance of a system can vary considerably depending on the users that use the system. Different characteristic user behaviors (users that spend different amounts of time reading, and away from the system), lead to different amounts of gain experienced by users. This finding shows that system developers would greatly benefit by knowing the behavior of their target user population.

We observed that duplicate sentences can exist in very large numbers in a web-scale corpus, specifically, the KBA stream corpus (Frank et al., 2014). We investigated the effect of including the duplicates of judged sentences into the qrels, on the evaluation of stream filtering systems, specifically the participating systems at TST 2013 and TST 2014. We compared the evaluation with duplicates included, to the respective tracks' evaluation, for the measures ELG, LC and MSU. Our key finding was that the relative ranking of runs does not change significantly, which is noteworthy given the fact that the duplicates of judged updates in the corpus can number 1000 times the judged updates. However, even though the relative performance of systems is not appreciably affected, the absolute scores of the systems changes significantly for over half the systems across the three different metrics for TST 2013. In contrast, TST 2014 runs do not show much change in performance over the track's measures even when duplicates are added to the qrels. This is mainly because, TST 2014 evaluation does not elide unjudged sentences but instead considers them to be

non-relevant. However, the performance of TST 2014 runs does change when evaluated using MSU when duplicates are added to the qrels.

The MSU user model essentially simulates users reading updates in sessions. The number of updates each modeled user reads in a session depends on the respective user's reading speed. If the user encounters a relevant update, the user gets gain. We observed that a large proportion of simulated users do not encounter all the judged relevant updates. About a third of the judged relevant updates are read by less than 1% of the modeled users. For evaluating a system, it may be beneficial to judge those updates that are read by most modeled users. Accordingly, we investigated pooling of updates for adjudication, using the probability of an update being read by users, as the pooling selection criterion. We explored alternative formulations for the probability of an update being read, and investigated depth-pooling based on probabilities as well as pooling based on probability mass cover.

We found that pools constructed using the TREC-standard depth based pooling based on updates' scores, have less than 45% overlap with the pools constructed using probabilities of updates being read. Furthermore, the overlap for the number of known relevant updates between the two pools does not exceed 70%, even for pools with a depth of 1000. Ascertaining the usefulness of pooling based on user-model induced probabilities is an interesting avenue of future work. Although such pooling methods may help to alleviate retrieval algorithm bias, they may turn out not to be reusable if the user model changes.

7.2 Limitations

Our evaluation measure was demonstrated with the help of a parameter sweep (Section 5.3). The parameter sweep allowed us to see how different types of users might experience gain differently using the stream filtering systems. However, we have no clear idea yet about the actual parameter values that reflect a real (and observed) user population. An obvious future work for MSU is its calibration to observed user behavior (Section 7.3.1).

MSU produces an easily interpretable score, e.g. an MSU of 8.3 for a system indicates that "users of the system can expect to get 8.3 units of gain on average". For TST 2013, all the topics had a uniform query duration length of exactly 10 days. This facilitated the computing of the MSU for a system as a simple average of the MSU scores for the topics of the track. However, for TST 2014, the topics have varying query durations. Although MSU computes the average expected gain over topics, the varying length of the query duration does not necessarily make MSU scores comparable across topics. Some form of normalization would be needed in order to compare MSU across topics and query durations.

Pooling using probabilities of sentences being read, or in the general case, pooling based on user-model induced probabilities, though an interesting thought experiment, can have many issues in reality. The primary issue is that the resulting test-collection could become unusable if the user model changes. A user may change his browsing behavior depending on the time of day, interest in the topic, the importance of the content read. At the very least, pooling by probabilities should be initiated only after calibration of the evaluation measure.

7.3 Future Work

7.3.1 MSU Calibration and Extensions

To our knowledge, an estimate of the distributions of time spent away from an IR system specifically when topics deal with evolving events, is not yet known. Estimating actual distributions of session and away times may require user studies and/or sessions log analyses. The sudden onset of news events makes user studies or recording user browsing behaviour in a live-setting logistically difficult. Search engine query-log based session analysis for news related information needs, may provide hints for actual user behaviour during an evolving news event, however, such data is hard to obtain and, identifying topic related queries may need manual annotation.

Search Engine Query Logs Analysis

Search engine query logs could help to identify patterns of search for breaking news events. There has been previous work on temporal analysis given a query log. Jones and Diaz (2004) classified queries as being atemporal, temporally ambiguous/unambiguous. Kulkarni et al. (2011) investigated how query intent changes with time. Query logs have also been used to detect trends, e.g. Ginsberg et al. (2009) modeled the spread of an influenza epidemic using queries submitted to Google. We may need to combine ideas from temporal analysis of query logs (Kanhabua, Ngoc Nguyen, and Nejdl, 2015) with session detection techniques (Gayo-Avello, 2009; Hagen, Stein, and Rb, 2011; Lucchese et al., 2011), in order to estimate parameters for MSU's user model.

Search log analysis may also help provide insight into the session/away behavior of users. News topics can be bursty, i.e., there can be periods of intense activity leading to availability of information at a faster rate than normal, especially immediately following an event. Correspondingly, users may have frequent sessions during bursts. On the other hand there would be quieter periods of user activity during nights/early morning hours. Investigations on how MSU's user model would change depending on successful calibration of the metric and an elaborate analysis of realistic user behavior.

User Browsing History

Web browsing histories of a sample of users could also help augment our understanding of user search behaviour when searching for breaking news and/or evolving events. However, this approach may be ethically nonviable due to privacy issues. An alternative approach would be to deploy a stream filtering system and have users use it. The data thus gathered may shed some light on actual user browsing behavior for streaming information access.

Novelty vs. Redundancy Tuning for Evaluation

Based on requirements of users, it is possible to envision a tuning parameter for evaluation of filtered streams that balances novelty and redundancy. (Baruah, Roegiest, and Smucker, 2014) found that there could be a large number of exact duplicates in the KBA corpus. Indeed, it is possible that many near-duplicates exist as well. Although, updates that do not contain new information do not contribute gain, reading the same information repeatedly could cause a loss in perceived usefulness of a system.

Previous work that balances novelty/redundancy with relevance includes, the application of maximal marginal relevance for evaluating summaries by Goldstein and Carbonell (1998), redundancy measures and their application to adaptive filtering techniques by Zhang, Callan, and Minka (2002). Allan, Wade, and Bolivar (2003) found that measuring novelty depends on the quality of relevance results. We could also apply the α-NDCG measure proposed by Clarke et al. (2008) to reward those systems that return novel updates, however, the effect of user sessions and/or presentation order of updates might need to be investigated for application of α-NDCG to MSU.

Normalization and Preference Ratios

MSU reports the expected gain a user may experience when using a system. However, MSU does not differentiate between varying lengths of topic query durations (which is indeed the case, for TST 2014). Computing the mean MSU across topics that have different query durations can potentially obfuscate the effect of the query duration on expected gain. The standard way to deal with such an issue would be to normalize the computed MSU with the "ideal" MSU.

Defining ideal MSU is ambiguous. If ideal MSU is considered to be the number of nuggets found (all delivered in time), then it could be hard to compare between a system that returns only 1 nugget, with a system that returns more than one. Another definition of ideal could be the number of unique nuggets found across all runs. Normalizing MSU with the number of unique nuggets found by all runs, essentially tells us the percentage of retrievable nuggets that can be expected to be found by the system in the specified query

duration.

A yet another way of normalization could be to divide the computed MSU by time units, e.g. MSU/day. However, day is an arbitrary unit of time and may not work well for query durations spanning a few hours. Section 5.3.3 demonstrates that normalization with time units may not reflect true system performance. Perhaps, an appropriate way to compare MSU across topics is to compute a preference ratio. Clarke and Smucker (2014) describe a pairwise comparison between 2 systems using preference ratios, which is the fraction of topics for which simulated users prefer one system over another.

7.3.2 Modeled Stream Utility for Different User Behaviors

Gain with different presentation orderings

There are variations possible in the order in which the updates are presented to the user. A system may present the updates in chronological order, reverse chronological order, or order them by a confidence score (ranked order) that indicates the importance of an update. Depending on factors that affect user behaviour the user may experience different amounts of gain for different result presentation orders. However, each change in presentation order necessitates a corresponding change in the user interfaces and thereby the user-model. For instance, Comarela et al. (2012) studied user interactions with their Twitter timeline and identified factors that influence replies to tweets and re-tweets. They investigated changes to the default reverse-chronological tweet presentation order. They found that their method of re-ranking tweets increases the percentage of replies and re-tweets by more than 50%.

For the news updates filtering task, we are more interested in finding updates that are of value to the user. The reverse-chronological order for presenting retrieved updates was modeled in our current approach (Section 5.1.4). A chronological order for updates may be desirable when the user wishes to know how the story evolved in the time spent away from the system. In comparison, a (system generated) ranked ordering may be preferable to users who are concerned with getting the most important information earliest. A ranked interface might also need careful analysis of the effect of graded levels of relevance assigned to respective nuggets.

Using *push* models with MSU

With the proliferation of *push notifications* by web and mobile applications to update users about latest content, developing a model of user behaviour with regards to push notifications is an exciting and a pertinent prospect. Online social networks, like Twitter[1] and Facebook[2], routinely push notifications of updates to users' Timelines[3] and NewsFeeds[4] respectively. News services and applications likewise, push notifications for news events. The TREC 2015 Microblog track had a specific task concerning the timeliness of push notifications for tweets about a topic (Lin et al., 2015).

The MSU user-model, in its current form, is a realization of a *pull* model. The users decide when they want to get updated about news events, i.e., the model is entirely user driven. Consider a user wanting to browse event related tweets using their mobile phone.

[1] https://www.twitter.com
[2] https://www.facebook.com
[3] https://support.twitter.com/articles/164083
[4] https://www.facebook.com/help/210346402339221

If the application does not push any notifications, and, the user opens the application to browse event related tweets, then the current MSU user model is applicable (after due calibration). A *push* user model involves considering the system as an active component that can influence user behavior. On receiving a push notification, a user may decide to (i) open the application immediately to read the updates, (ii) open the application at a later point in time, or (iii) ignore the notification. The characteristic user behavior changes considerably due to every single push notification because the user may have consciously made a note of the notification. The frequency of notifications and the amount of content presented with each notification may affect the gain experienced by the users. From an evaluation perspective, MSU not only needs to measure the gain encountered but also gauge the worthiness of the notifications.

References

[1] Rafik Abbes, Karen Pinel-Sauvagnat, Nathalie Hernandez, and Mohand Boughanem. "IRIT at TREC Temporal Summarization 2014." In: *TREC*. 2014.

[2] Eugene Agichtein, Eric Brill, and Susan Dumais. "Improving Web Search Ranking by Incorporating User Behavior Information". In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '06. Seattle, Washington, USA: ACM, 2006, pp. 19–26. ISBN: 1-59593-369-7. DOI: 10.1145/1148170.1148177. URL: http://doi.acm.org/10.1145/1148170.1148177.

[3] Azzah Al-Maskari, Mark Sanderson, Paul D. Clough, and Eija Airio. "The good and the bad system: does the test collection predict users' effectiveness?" In: *SIGIR*. 2008, pp. 59–66.

[4] James Allan, ed. *Topic Detection and Tracking: Event-based Information Organization*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN: 0-7923-7664-1.

[5] James Allan, Rahul Gupta, and Vikas Khandelwal. "Temporal Summaries of News Topics." In: *SIGIR*. 2001, pp. 10–18.

[6] James Allan, Courtney Wade, and Alvaro Bolivar. "Retrieval and Novelty Detection at the Sentence Level". In: *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. SIGIR '03. Toronto, Canada: ACM, 2003, pp. 314–321. ISBN: 1-58113-646-3. DOI: 10.1145/860435.860493.

[7] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. "Topic detection and tracking pilot study final report". In: (1998).

[8] Javed A. Aslam, Virgiliu Pavlu, and Robert Savell. "A unified model for metasearch, pooling, and system evaluation." In: *CIKM*. 2003, pp. 484–491.

[9] Javed A. Aslam, Virgiliu Pavlu, and Emine Yilmaz. “A statistical method for system evaluation using incomplete judgments.” In: *SIGIR*. 2006, pp. 541–548.

[10] Javed A. Aslam and Emine Yilmaz. “Inferring document relevance from incomplete information.” In: *CIKM*. 2007, pp. 633–642.

[11] Javed A. Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Fernando Diaz, and Tetsuya Sakai. “TREC 2013 Temporal Summarization.” In: *TREC*. 2013.

[12] Javed A. Aslam, Matthew Ekstrand-Abueg, Virgil Pavlu, Fernando Diaz, Richard McCreadie, and Tetsuya Sakai. “TREC 2014 Temporal Summarization Track Overview.” In: *TREC*. 2014.

[13] Leif Azzopardi. “Modelling Interaction with Economic Models of Search”. In: *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR ’14. Gold Coast, Queensland, Australia: ACM, 2014, pp. 3–12. ISBN: 978-1-4503-2257-7. DOI: 10.1145/2600428.2609574.

[14] Leif Azzopardi. “Usage based effectiveness measures: monitoring application performance in information retrieval.” In: *CIKM*. 2009, pp. 631–640.

[15] Leif Azzopardi and Krisztian Balog. “Towards a living lab for information retrieval research and development”. In: *Multilingual and Multimodal Information Access Evaluation*. Springer, 2011, pp. 26–37.

[16] Leif Azzopardi and Guido Zuccon. “Building and Using Models of Information Seeking, Search and Retrieval: Full Day Tutorial”. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 2015, pp. 1107–1110. DOI: 10.1145/2766462.2767874.

[17] Ricardo Baeza-Yates, Julien Masanès, and Marc Spaniol. *Proceedings of the 3rd International Temporal Web Analytics Workshop (TempWeb 2012)*. 2012.

[18] Ricardo Baeza-Yates, Julien Masanès, and Marc Spaniol. *Proceedings of the 3rd International Temporal Web Analytics Workshop (TempWeb 2013)*. 2013.

[19] Krisztian Balog, David Elsweiler, Evangelos Kanoulas, Liadh Kelly, and Mark D. Smucker. “Report on the CIKM workshop on living labs for information retrieval evaluation.” In: *SIGIR Forum* 48.1 (2014), pp. 21–28.

[20] Gaurav Baruah, Adam Roegiest, and Mark D. Smucker. “Pooling for User-Oriented Evaluation Measures.” In: *ICTIR*. 2015, pp. 341–344.

[21] Gaurav Baruah, Adam Roegiest, and Mark D. Smucker. "The effect of expanding relevance judgements with duplicates". In: *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*. 2014, pp. 1159–1162. DOI: 10.1145/2600428.2609534.

[22] Gaurav Baruah, Mark D. Smucker, and Charles L. A. Clarke. "Evaluating Streams of Evolving News Events." In: *SIGIR*. 2015, pp. 675–684.

[23] Gaurav Baruah, Rakesh Guttikonda, Adam Roegiest, and Olga Vechtomova. "University of Waterloo at the TREC 2013 Temporal Summarization Track". In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[24] Feza Baskaya, Heikki Keskustalo, and Kalervo Järvelin. "Modeling behavioral factors in interactive information retrieval". In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. CIKM '13. San Francisco, California, USA: ACM, 2013, pp. 2297–2302. ISBN: 978-1-4503-2263-8. DOI: 10.1145/2505515.2505660.

[25] Feza Baskaya, Heikki Keskustalo, and Kalervo Järvelin. "Time drives interaction: simulating sessions in diverse searching environments." In: *SIGIR*. 2012, pp. 105–114. URL: http://doi.acm.org/10.1145/2348283.2348301.

[26] Nicholas J. Belkin and W. Bruce Croft. "Information Filtering and Information Retrieval: Two Sides of the Same Coin?" In: *Commun. ACM* 35.12 (1992), pp. 29–38.

[27] Klaus Berberich, James Caverlee, Miles Efron, Claudia Hauff, Vanessa Murdock, Milad Shokouhi, and Bart Thomee. "SIGIR 2015 Workshop on Temporal, Social and Spatially-aware Information Access (#TAIA2015)". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: ACM, 2015, pp. 1149–1150. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767860.

[28] Yaniv Bernstein and Justin Zobel. "Redundant documents and search effectiveness." In: *CIKM*. 2005, pp. 736–743.

[29] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[30] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. "Hierarchical Topic Models and the Nested Chinese Restaurant Process." In: *NIPS*. 2003, pp. 17–24.

[31] Chris Buckley and Ellen M. Voorhees. "Evaluating evaluation measure stability." In: *SIGIR*. 2000, pp. 33–40.

[32] Chris Buckley and Ellen M. Voorhees. "Retrieval evaluation with incomplete information." In: *SIGIR*. 2004, pp. 25–32.

[33] Chris Buckley, Darrin Dimmick, Ian Soboroff, and Ellen M. Voorhees. "Bias and the limits of pooling for large collections." In: *Inf. Retr.* 10.6 (2007), pp. 491–508.

[34] Stefan Büttcher, Charles Clarke, and Gordon V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press, 2010. ISBN: 0262026511, 9780262026512.

[35] Stefan Büttcher, Charles L. A. Clarke, Peter C. K. Yeung, and Ian Soboroff. "Reliable Information Retrieval Evaluation with Incomplete and Biased Judgements". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '07. Amsterdam, The Netherlands: ACM, 2007, pp. 63–70. ISBN: 978-1-59593-597-7. DOI: 10.1145/1277741.1277755. URL: http://doi.acm.org/10.1145/1277741.1277755.

[36] Jaime Carbonell and Jade Goldstein. "The use of MMR, diversity-based reranking for reordering documents and producing summaries". In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1998, pp. 335–336.

[37] Ben Carterette, James Allan, and Ramesh K. Sitaraman. "Minimal test collections for retrieval evaluation." In: *SIGIR*. 2006, pp. 268–275.

[38] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. "Expected reciprocal rank for graded relevance." In: *CIKM*. 2009, pp. 621–630.

[39] Moses Charikar. "Similarity estimation techniques from rounding algorithms." In: *STOC*. 2002, pp. 380–388.

[40] Lei Chen, Hainan Zhang, Siying Li, Zhiyuan Ji, Qian Liu, Yue Liu, Dayong Wu, and Xueqi Cheng. "ICTNET at Temporal Summarization Track TREC 2014." In: *TREC*. 2014.

[41] Charles L. A. Clarke and Mark D. Smucker. "Time well spent". In: *Fifth Information Interaction in Context Symposium, IIiX '14, Regensburg, Germany, August 26-29, 2014*. 2014, pp. 205–214. DOI: 10.1145/2637002.2637026.

[42] Charles L. A. Clarke, Luanne Freund, Mark D. Smucker, and Emine Yilmaz. "Report on the SIGIR 2013 workshop on modeling user behavior for information retrieval evaluation (MUBE 2013)." In: *SIGIR Forum* 47.2 (2013), pp. 84–95.

[43] Charles LA Clarke, Gordon V Cormack, and Thomas R Lynam. "Exploiting redundancy in question answering". In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2001, pp. 358–365.

[44] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. "Novelty and Diversity in Information Retrieval Evaluation". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Singapore, Singapore: ACM, 2008, pp. 659–666. ISBN: 978-1-60558-164-4. DOI: 10.1145/1390334.1390446.

[45] Cyril Cleverdon. "The Cranfield tests on index language devices". In: *Aslib proceedings*. Vol. 19. 6. MCB UP Ltd. 1967, pp. 173–194.

[46] Giovanni Comarela, Mark Crovella, Virgilio Almeida, and Fabrcio Benevenuto. "Understanding factors that affect response rates in twitter." In: *HT*. 2012, pp. 123–132. URL: http://doi.acm.org/10.1145/2309996.2310017.

[47] Gordon V Cormack, Charles LA Clarke, and Stefan Büettcher. "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2009, pp. 758–759.

[48] Gordon V. Cormack and Thomas R. Lynam. "Power and bias of subset pooling strategies." In: *SIGIR*. 2007, pp. 837–838.

[49] Gordon V. Cormack, Christopher R. Palmer, and Charles L. A. Clarke. "Efficient Construction of Large Test Collections". In: *SIGIR*. 1998, pp. 282–289.

[50] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[51] Anita Crescenzi, Diane Kelly, and Leif Azzopardi. "Time Pressure and System Delays in Information Search." In: *SIGIR*. 2015, pp. 767–770.

[52] Aron Culotta. "Towards detecting influenza epidemics by analyzing Twitter messages". In: *Proceedings of the first workshop on social media analytics*. ACM. 2010, pp. 115–122.

[53] Hoa Trang Dang, Jimmy Lin, and Diane Kelly. "Overview of the TREC 2006 Question Answering Track". In: *Proceedings of TREC 2006*. 2007.

[54] Zoltan Dezsö, Eivind Almaas, András Lukács, Balázs Rácz, István Szakadát, and A-L Barabási. "Dynamics of information access on the web". In: *Physical Review E* 73.6 (2006), p. 066132.

[55] Fernando Diaz. "Experimentation Standards for Crisis Informatics." In: *SIGIR Forum* 48.2 (2014), pp. 22–30.

[56] Fernando Diaz, Susan Dumais, Miles Efron, Kira Radinsky, Maarten de Rijke, and Milad Shokouhi. "SIGIR 2013 Workshop on Time Aware Information Access (#TAIA2013)". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '13. Dublin, Ireland: ACM, 2013, pp. 1137–1137. ISBN: 978-1-4503-2034-4. DOI: 10.1145/2484028.2491802.

[57] Fernando Diaz, Claudia Hauff, Vanessa Murdock, Maarten de Rijke, and Milad Shokouhi. "SIGIR 2014 Workshop on Temporal, Social and Spatially-aware Information Access (#TAIA2014)". In: *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '14. Gold Coast, Queensland, Australia: ACM, 2014, pp. 1298–1298. ISBN: 978-1-4503-2257-7. DOI: 10.1145/2600428.2600740.

[58] Fernando Diaz, Susan Dumais, Kira Radinsky, Maarten de Rijke, and Milad Shokouhi. "#TAIA2012". In: *SIGIR Forum* 46.2 (2012), pp. 102–106. ISSN: 0163-5840. DOI: 10.1145/2422256.2422270.

[59] Laura Dietz, Jeffrey Dalton, and Krisztian Balog. "Time-aware evaluation of cumulative citation recommendation systems". In: *Proceedings of SIGIR 2013 Workshop on Time-aware Information Access, TAIA*. Vol. 2013. 2013.

[60] Gavin Doherty, Mieke Massink, and Giorgio Faconti. "Reasoning about interactive systems with stochastic models". In: *Interactive Systems: Design, Specification, and Verification*. Springer, 2001, pp. 144–163.

[61] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. "Towards recency ranking in web search". In: *Proceedings of the third ACM international conference on Web search and data mining*. ACM. 2010, pp. 11–20.

[62] Paul Earle, Michelle Guy, Richard Buckmaster, Chris Ostrum, Scott Horvath, and Amy Vaughan. "OMG earthquake! Can Twitter improve earthquake response?" In: *Seismological Research Letters* 81.2 (2010), pp. 246–251.

[63] *Evaluation script for Temporal Summarization 2014*. http://trec.nist.gov/data/tempsumm/2014/tseval.py. 2014.

[64] Jonathan G. Fiscus and George R. Doddington. "Topic Detection and Tracking". In: ed. by James Allan. Norwell, MA, USA: Kluwer Academic Publishers, 2002. Chap. Topic Detection and Tracking Evaluation Overview, pp. 17–31. ISBN: 0-7923-7664-1.

[65] John R Frank, Steven J Bauer, Max Kleiman-Weiner, Daniel A Roberts, Nilesh Tripuraneni, Ce Zhang, Christopher Re, Ellen Voorhees, and Ian Soboroff. *Evaluating Stream Filtering for Entity Profile Updates for TREC 2013 (KBA Track Overview)*. Tech. rep. DTIC Document, 2013.

[66] John R. Frank, Max Kleiman-Weiner, Daniel A. Roberts, Ellen M. Voorhees, and Ian Soboroff. "Evaluating Stream Filtering for Entity Profile Updates in TREC 2012, 2013, and 2014." In: *TREC*. 2014.

[67] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *J. Comput. Syst. Sci.* 55.1 (Aug. 1997), pp. 119–139. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504. URL: http://dx.doi.org/10.1006/jcss.1997.1504.

[68] Norbert Fuhr. "A probability ranking principle for interactive information retrieval". In: *Information Retrieval* 11.3 (2008), pp. 251–265.

[69] Daniel Gayo-Avello. "A survey on session detection methods in query logs and a proposal for future evaluation." In: *Inf. Sci.* 179.12 (2009), pp. 1822–1843.

[70] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. "Detecting influenza epidemics using search engine query data". In: *Nature* 457.7232 (2009), pp. 1012–1014.

[71] Jade Goldstein and Jaime Carbonell. "Summarization: (1) Using MMR for Diversity - Based Reranking and (2) Evaluating Summaries". In: *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*. TIPSTER '98. Baltimore, Maryland: Association for Computational Linguistics, 1998, pp. 181–195. DOI: 10.3115/1119089.1119120.

[72] Qi Guo, Fernando Diaz, and Elad Yom-Tov. "Updating Users about Time Critical Events." In: *ECIR*. 2013, pp. 483–494.

[73] Matthias Hagen, Benno Stein, and Tino Rb. "Query session detection as a cascade." In: *CIKM*. 2011, pp. 147–152. URL: http://doi.acm.org/10.1145/2063576.2063602.

[74] Jiyin He, Marc Bron, Arjen de Vries, Leif Azzopardi, and Maarten de Rijke. "Untangling Result List Refinement and Ranking Quality: A Framework for Evaluation and Prediction". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: ACM, 2015, pp. 293–302. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767740.

[75] Liangjie Hong, Byron Dom, Siva Gurumurthy, and Kostas Tsioutsiouliklis. "A time-dependent topic model for multiple text streams". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2011, pp. 832–840.

[76] Frank Hopfgartner, Benjamin Kille, Andreas Lommatzsch, Till Plumbaum, Torben Brodt, and Tobias Heintz. "Benchmarking News Recommendations in a Living Lab." In: *CLEF*. 2014, pp. 250–267.

[77] Muhammad Imran, Shady Mamoon Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. "Extracting information nuggets from disaster-related messages in social media". In: *Proc. of ISCRAM, Baden-Baden, Germany* (2013).

[78] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated Gain-based Evaluation of IR Techniques". In: *ACM Trans. Inf. Syst.* 20.4 (2002), pp. 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418.

[79] Kalervo Järvelin, Susan L. Price, Lois M. L. Delcambre, and Marianne Lykke Nielsen. "Discounted Cumulated Gain Based Evaluation of Multiple-query IR Sessions". In: *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*. ECIR'08. Glasgow, UK: Springer-Verlag, 2008, pp. 4–15. ISBN: 3-540-78645-7, 978-3-540-78645-0.

[80] Hideo Joho, Adam Jatowt, and Roi Blanco. "NTCIR temporalia: a test collection for temporal information access research." In: *WWW (Companion Volume)*. 2014, pp. 845–850.

[81] Hideo Joho, Adam Jatowt, and Roi Blanco. "Temporal information searching behaviour and strategies." In: *Inf. Process. Manage.* 51.6 (2015), pp. 834–850.

[82] Hideo Joho, Adam Jatowt, Roi Blanco, Hajime Naka, and Shuhei Yamamoto. "Overview of NTCIR-11 Temporal Information Access (Temporalia) Task." In: *NTCIR*. 2014.

[83] Rosie Jones and Fernando Diaz. *Temporal profiles of queries*. Tech. rep. ACM Trans. Inf. Syst, 2004.

[84] Nattiya Kanhabua, Roi Blanco, and Kjetil Nørvåg. "Temporal Information Retrieval". In: *Found. Trends Inf. Retr.* 9.2 (2015), pp. 91–208. ISSN: 1554-0669. DOI: 10.1561/1500000043.

[85] Nattiya Kanhabua, Tu Ngoc Nguyen, and Wolfgang Nejdl. "Learning to Detect Event-Related Queries for Web Search". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15 Companion. Florence, Italy: ACM, 2015, pp. 1339–1344. ISBN: 978-1-4503-3473-0. DOI: 10.1145/2740908.2741698.

[86] Evangelos Kanoulas, Ben Carterette, Paul D. Clough, and Mark Sanderson. "Evaluating multi-query sessions." In: *SIGIR*. 2011, pp. 1053–1062.

[87] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. "Dynamic feature space and incremental feature selection for the classification of textual data streams". In: *Knowledge Discovery from Data Streams* (2006), pp. 107–116.

[88] Makoto P. Kato, Matthew Ekstrand-Abueg, Virgil Pavlu, Tetsuya Sakai, Takehiro Yamamoto, and Mayu Iwata. "Overview of the NTCIR-10 1CLICK-2 Task." In: *NTCIR*. 2013.

[89] *KBA Stream Corpus*. http://trec-kba.org/kba-stream-corpus-2013.shtml. 2013.

[90] Chris Kedzie, Kathleen McKeown, and Fernando Diaz. "Predicting Salient Updates for Disaster Summarization." In: *ACL (1)*. 2015, pp. 1608–1617.

[91] Diane Kelly. "Methods for evaluating interactive information retrieval systems with users". In: *Foundations and Trends in Information Retrieval* 3.12 (2009), pp. 1–224.

[92] Tom Kenter, Krisztian Balog, and Maarten de Rijke. "Evaluating document filtering systems over time." In: *Inf. Process. Manage.* 51.6 (2015), pp. 791–808.

[93] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. "Detecting change in data streams". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment. 2004, pp. 180–191.

[94] Jon Kleinberg. "Bursty and Hierarchical Structure in Streams". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. Edmonton, Alberta, Canada: ACM, 2002, pp. 91–101. ISBN: 1-58113-567-X. DOI: 10.1145/775047.775061.

[95] Jon Kleinberg. "Temporal Dynamics of On-Line Information Streams". In: *IN DATA STREAM MANAGEMENT: PROCESSING HIGH-SPEED DATA*. 2006.

[96] Anagha Kulkarni, Jaime Teevan, Krysta Marie Svore, and Susan T. Dumais. "Understanding temporal query dynamics." In: *WSDM*. 2011, pp. 167–176.

[97] Chung-Hong Lee and Tzan-Feng Chien. "Leveraging microblogging big data with a modified density-based clustering approach for event awareness and topic ranking". In: *Journal of Information Science* (2013), p. 0165551513478738.

[98] *Lemur Project*. http://www.lemurproject.org/.

[99] Xiaoyan Li and W. Bruce Croft. "Time-based Language Models". In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. CIKM '03. New Orleans, LA, USA: ACM, 2003, pp. 469–475. ISBN: 1-58113-723-0. DOI: 10.1145/956863.956951.

[100] Chin yew Lin. "Rouge: a package for automatic evaluation of summaries". In: *Text summarization branches out: Proceedings of the ACL-04 workshop*. 2004, pp. 25–26.

[101] Jimmy Lin and Dina Demner-Fushman. "Automatically evaluating answers to definition questions". In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2005, pp. 931–938.

[102] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. "Overview of the TREC-2014 Microblog Track". In: *TREC 2014, Gaithersburg, Maryland, USA, 2014*. 2014.

[103] Jimmy Lin, Miles Efron, Yulu Wang, Garrick Sherman, and Ellen Voorhees. "Overview of the TREC-2015 Microblog Track". In: *TREC 2015, Gaithersburg, Maryland, USA, 2015*. 2015.

[104] Jimmy J Lin. "Is Question Answering Better than Information Retrieval? Towards a Task-Based Evaluation Framework for Question Series." In: *HLT-NAACL*. 2007, pp. 212–219.

[105] Qian Liu, Yue Liu, Dayong Wu, and Xueqi Cheng. "ICTNET at Temporal Summarization Track TREC 2013". In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[106] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. "Identifying task-based sessions in search engine query logs." In: *WSDM*. 2011, pp. 277–286.

[107] Gregory Marton and Alexey Radul. "Nuggeteer: Automatic Nugget-Based Evaluation using Descriptions and Judgements." In: *HLT-NAACL*. 2006.

[108] Richard McCreadie, Craig Macdonald, and Iadh Ounis. "Incremental Update Summarization: Adaptive Sentence Selection based on Prevalence and Novelty". In: (2014).

[109] Richard McCreadie, M-Dyaa Albakour, Stuart Mackie, Nut Limsopatham, Craig Macdonald, Iadh Ounis, and Bekir Taner Dinçer. "University of Glasgow at TREC 2013: Experiments with Terrier in Contextual Suggestion, Temporal Summarisation and Web Tracks". In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[110] Richard McCreadie, Romain Deveaud, M-Dyaa Albakour, Stuart Mackie, Nut Limsopatham, Craig Macdonald, Iadh Ounis, Thibaut Thonet, and Bekir Taner Diner. "University of Glasgow at TREC 2014: Experiments with Terrier in Contextual Suggestion, Temporal Summarisation and Web Tracks." In: *TREC*. 2014. URL: http://trec.nist.gov/pubs/trec23/papers/pro-uogTr_cs-ts-web.pdf.

[111] Donald Metzler, Rosie Jones, Fuchun Peng, and Ruiqiang Zhang. "Improving Search Relevance for Implicitly Temporal Queries". In: *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: ACM, 2009, pp. 700–701. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1572085.

[112] Tomas Mikolov, Kai Chen 0010, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint arXiv:1301.3781* (2013).

[113] Alistair Moffat, William Webber, and Justin Zobel. "Strategic system comparisons via targeted relevance judgments." In: *SIGIR*. 2007, pp. 375–382.

[114] Alistair Moffat and Justin Zobel. "Rank-biased precision for measurement of retrieval effectiveness." In: *ACM Trans. Inf. Syst.* 27.1 (2008).

[115] Peter Oram. *WordNet: An electronic lexical database. Christiane Fellbaum (Ed.). Cambridge, MA: MIT Press, 1998. Pp. 423.* 2001.

[116] Iadh Ounis, Craig Macdonald, Jimmy Lin, and Ian Soboroff. "Overview of the TREC-2011 microblog track". In: *TREC 2011, Gaithersburg, Maryland, USA, 2011*. 2011.

[117] Michael J Paul and Mark Dredze. "You are what you Tweet: Analyzing Twitter for public health." In: *ICWSM*. 2011, pp. 265–272.

[118] Virgiliu Pavlu, Shahzad Rajput, Peter B. Golbus, and Javed A. Aslam. "IR system evaluation using nugget-based test collections." In: *WSDM*. 2012, pp. 393–402.

[119] Alexei Pozdnoukhov and Christian Kaiser. "Space-time dynamics of topics in streaming text". In: *Proceedings of the 3rd ACM SIGSPATIAL international workshop on location-based social networks*. ACM. 2011, pp. 1–8.

[120] Yuanyuan Qi, Qinlong Wang, Chuchu Huang, Bo Tang, Weiran Xu, Guang Chen, and Jun Guo. "The Information Extraction systems of BUPT_PRIS at TREC2014 Temporal Summarization Track." In: *TREC*. 2014.

[121] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. "How does clickthrough data reflect retrieval quality?" In: *CIKM*. 2008, pp. 43–52.

[122] Shahzad Rajput, Matthew Ekstrand-Abueg, Virgiliu Pavlu, and Javed A. Aslam. "Constructing test collections by inferring document relevance via extracted relevant information." In: *CIKM*. 2012, pp. 145–154.

[123] Stephen Robertson. "Threshold Setting and Performance Optimization in Adaptive Filtering". In: *Information Retrieval* 5.2 (2002), pp. 239–256. ISSN: 1573-7659. DOI: 10.1023/A:1015702129514.

[124] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Found. Trends Inf. Retr.* 3.4 (Apr. 2009), pp. 333–389. ISSN: 1554-0669. DOI: 10.1561/1500000019. URL: http://dx.doi.org/10.1561/1500000019.

[125] J. Rogstadius, M. Vukovic, C.A. Teixeira, V. Kostakos, E. Karapanos, and J.A. Laredo. "CrisisTracker: Crowdsourced social media curation for disaster awareness". In: *IBM Journal of Research and Development* 57.5 (2013), 4:1–4:13. ISSN: 0018-8646. DOI: 10.1147/JRD.2013.2260692.

[126] Tetsuya Sakai. "Comparing Metrics Across TREC and NTCIR:: The Robustness to Pool Depth Bias". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. Singapore, Singapore: ACM, 2008, pp. 691–692. ISBN: 978-1-60558-164-4. DOI: 10.1145/1390334.1390454. URL: http://doi.acm.org/10.1145/1390334.1390454.

[127] Tetsuya Sakai. "Comparing metrics across TREC and NTCIR: the robustness to system bias." In: *CIKM*. 2008, pp. 581–590.

[128] Tetsuya Sakai. "Evaluating evaluation metrics based on the bootstrap." In: *SIGIR*. 2006, pp. 525–532.

[129] Tetsuya Sakai and Zhicheng Dou. "Summaries, ranked retrieval and sessions: a unified framework for information access evaluation." In: *SIGIR*. 2013, pp. 473–482.

[130] Tetsuya Sakai and Noriko Kando. "On information retrieval metrics designed for evaluation with incomplete relevance assessments." In: *Inf. Retr.* 11.5 (2008), pp. 447–470.

[131] Mark D Smucker and James Allan. "Find-similar: similarity browsing as a search tool". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2006, pp. 461–468.

[132] Mark D. Smucker, James Allan, and Ben Carterette. "A comparison of statistical significance tests for information retrieval evaluation." In: *CIKM*. 2007, pp. 623–632.

[133] Mark D. Smucker, James Allan, and Blagovest Dachev. "Human question answering performance using an interactive document retrieval system". In: *Information Interaction in Context: 2012, IIix'12, Nijmegen, The Netherlands, August 21-24, 2012*. 2012, pp. 35–44. DOI: 10.1145/2362724.2362735.

[134] Mark D. Smucker and Charles L. A. Clarke. "Modeling user variance in time-biased gain". In: *Human-Computer Information Retrieval Symposium, HCIR 2012, Cambridge, MA, USA, October 4-5, 2012*. 2012, p. 3. DOI: 10.1145/2391224.2391227.

[135] Mark D. Smucker and Charles L. A. Clarke. "Stochastic simulation of time-biased gain." In: *CIKM*. 2012, pp. 2040–2044.

[136] Mark D. Smucker and Charles L. A. Clarke. "The Fault, Dear Researchers, is not in Cranfield, But in our Metrics, that they are Unrealistic." In: *EuroHCIR*. 2012, pp. 11–12.

[137] Mark D. Smucker and Charles L. A. Clarke. "Time-based calibration of effectiveness measures". In: *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*. 2012, pp. 95–104. DOI: 10.1145/2348283.2348300.

[138] Mark D. Smucker and Chandra Prakash Jethani. "Human performance and retrieval precision revisited." In: *SIGIR*. 2010, pp. 595–602.

[139] Mark D Smucker, Gabriella Kazai, and Matthew Lease. *Overview of the TREC 2013 Crowdsourcing Track*. Tech. rep. 2013.

[140] Ian Soboroff. "Overview of the TREC 2004 Novelty Track." In: *TREC*. 2004.

[141] Ian Soboroff and Donna Harman. "Overview of the TREC 2003 Novelty Track." In: *TREC*. 2003, pp. 38–53.

[142] Ian Soboroff, Iadh Ounis, Craig Macdonald, and Jimmy Lin. "Overview of the TREC-2012 Microblog Track". In: *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012*. 2012.

[143] Marc Spaniol, Julien Masanès, and Ricardo Baeza-Yates. "The 4th Temporal Web Analytics Workshop (TempWeb'14)". In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14 Companion. Seoul, Korea: ACM, 2014, pp. 863–864. ISBN: 978-1-4503-2745-9. DOI: 10.1145/2567948.2579047.

[144] K. Sparck-Jones and C.J. Van Rijsbergen. "Report on the need for and provision of an "ideal" information retrieval test collection". In: (1975).

[145] Russell Swan and James Allan. "Automatic Generation of Overview Timelines". In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '00. Athens, Greece: ACM, 2000, pp. 49–56. ISBN: 1-58113-226-3. DOI: 10.1145/345508.345546.

[146] Jaime Teevan, Kevyn Collins-Thompson, Ryen W White, Susan T Dumais, and Yubin Kim. "Slow Search: Information Retrieval without Time Constraints". In: *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*. ACM. 2013, p. 1.

[147] *TREC KBA Stream Corpus*. http://trec-kba.org/kba-stream-corpus-2014.shtml. 2014.

[148] Andrew Trotman and Dylan Jenkinson. "IR evaluation using multiple assessors per topic". In: *ADCS* (2007).

[149] Olga Vechtomova. "A Semi-supervised Approach to Extracting Multiword Entity Names from User Reviews". In: *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search*. JIWES '12. Portland, Oregon, USA: ACM, 2012, 2:1–2:6. ISBN: 978-1-4503-1601-9. DOI: 10.1145/2379307.2379309.

[150] Sarah Vieweg, Amanda L. Hughes, Kate Starbird, and Leysia Palen. "Microblogging During Two Natural Hazards Events: What Twitter May Contribute to Situational Awareness". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 1079–1088. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753486.

[151] Ellen M. Voorhees. "Overview of the TREC 2004 Question Answering Track". In: *Proceedings of TREC 2004*. 2005.

[152] Ellen M. Voorhees. "Overview of the TREC 2005 Question Answering Track". In: *Proceedings of TREC 2005*. 2006.

[153] Ellen M. Voorhees. "Variations in relevance judgments and the measurement of retrieval effectiveness." In: *Inf. Process. Manage.* 36.5 (2000), pp. 697–716.

[154] Ellen M. Voorhees and Chris Buckley. "The effect of topic set size on retrieval experiment error." In: *SIGIR*. 2002, pp. 316–323.

[155] Ellen M Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*. Vol. 1. MIT press Cambridge, 2005.

[156] Yulu Wang, Garrick Sherman, Jimmy Lin, and Miles Efron. "Assessor Differences and User Preferences in Tweet Timeline Generation". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2015, pp. 615–624.

[157] Yaoyi Xi, Bicheng Li, Jie Zhou, and Yongwang Tang. "ZZISTI at TREC2013 Temporal Summarization Track". In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[158] Tan Xu, Douglas W. Oard, and Paul McNamee. "HLTCOE at TREC 2013: Temporal Summarization". In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[159] Yiming Yang and Abhimanyu Lad. "Modeling Expected Utility of Multi-session Information Distillation". In: *Advances in Information Retrieval Theory, Second International Conference on the Theory of Information Retrieval, ICTIR 2009, Cambridge, UK, September 10-12, 2009, Proceedings*. 2009, pp. 164–175. DOI: 10.1007/978-3-642-04417-5_15.

[160] Zhen Yang, Fei Yao, Huayang Sun, Yun Zhao, Yingxu Lai, and Kefeng Fan. "BJUT at TREC 2013 Temporal Summarization Track". In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[161] Emine Yilmaz and Javed A. Aslam. "Estimating average precision with incomplete and imperfect judgments." In: *CIKM*. 2006, pp. 102–111.

[162] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. “A new rank correlation coefficient for information retrieval.” In: *SIGIR*. 2008, pp. 587–594.

[163] Elad Yom-Tov and Fernando Diaz. “Out of sight, not out of mind: on the effect of social and physical detachment on information need.” In: *SIGIR*. 2011, pp. 385–394.

[164] Chunyun Zhang, Weiyan Xu, Fanyu Meng, Hongyan Li, Tong Wu, and Lixin Xu. “The Information Extraction Systems of PRIS at Temporal Summarization Track”. In: *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*. Ed. by Ellen M. Voorhees. Vol. Special Publication 500-302. National Institute of Standards and Technology (NIST), 2013.

[165] Yi Zhang, James P. Callan, and Thomas P. Minka. “Novelty and redundancy detection in adaptive filtering.” In: *SIGIR*. 2002, pp. 81–88.

[166] Yun Zhao, Fei Yao, Huayang Sun, and Zhen Yang. “BJUT at TREC 2014 Temporal Summarization Track.” In: *TREC*. 2014.

[167] Yunyue Zhu and Dennis Shasha. “Efficient elastic burst detection in data streams”. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2003, pp. 336–345.

[168] Justin Zobel. “How Reliable Are the Results of Large-Scale Information Retrieval Experiments?” In: *SIGIR*. 1998, pp. 307–314.