Tracking Events in Social Media

by

Luchen Tan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2017

Abstract

Tracking topical events in social media streams, such as Twitter, provides a means for users to keep up-to-date on topics of interest to them. This tracking may last a period of days, or even weeks. These events and topics might be provided by users explicitly, or generated for users from selected news articles. Push notification from social media provides a method to push the updates directly to the users on their mobile devices or desktops.

In this thesis, we start with a lexical comparison between carefully edited prose and social media posts, providing an improved understanding of word usage within social media. Compared with carefully edited prose, such as news articles and Wikipedia articles, the language of social media is informal in the extreme. By using word embeddings, we identify words whose usage differs greatly between a Wikipedia corpus and a Twitter corpus.

Following from this work, we explore a general method for developing succinct queries, reflecting the topic of a given news article, for the purpose of tracking the associated news event within a social media stream. A series of probe queries are generated from an initial set of candidate keywords extracted from the article. By analyzing the results of these probes, we rank and trim the candidate set to create a succinct query. The method can also be used for linking and searching among different collections.

Given a query for topical events, push notification to users directly from social media streams provides a method for them to keep up-to-date on topics of personal interest. We determine that the key to effective notification lies in controlling of update volume, by establishing and maintaining appropriate thresholds for pushing updates. We explore and evaluate multiple threshold setting strategies. Push notifications should be relevant to the personal interest, and timely, with pushes occurring as soon as after the actual event occurrence as possible and novel for providing non-duplicate information. An analysis of existing evaluation metrics for push notification reflects different assumptions regarding user requirements. This analysis leads to a framework that places different weights and penalties on different behaviours and can guide the future development of a family of evaluation metrics that more accurately models user needs.

Throughout the thesis, rank similarity measures are applied to compare rankings generated by various experiments. As a final component, we develop a family of rank similarity metrics based on maximized effectiveness difference, each derived from a traditional information retrieval evaluation measure. Computing this maximized effectiveness difference (MED) requires the solution of an optimization problem that varies in difficulty, depending on the associated measure. We present solutions for several standard effectiveness measures, including nDCG, MAP, and ERR. Through experimental validation, we show that MED reveals meaningful differences between retrieval runs. Mathematically, MED is a metric, regardless of the associated measure. Prior work has established a number of other desiderata for rank similarity in the context of search, and we demonstrate that MED satisfies these requirements. Unlike previous proposals, MED allows us to directly translate assumptions about user behavior from any established effectiveness measure to create a corresponding rank similarity measure. In addition, MED cleanly accommodates partial relevance judgments, and if complete relevance information is available, it reduces to a simple difference between effectiveness values.

Acknowledgements

First of all, I would like to express my sincere gratitude and immeasurable appreciation to my supervisor, Prof. Charles L. A. Clarke, a leading Information Retrieval scientist and a great mentor during my doctoral program life at the University of Waterloo. I have been extremely lucky to have a supervisor who provides instructive guidance and tremendous support throughout these years. His critical thinking, insightful academic advice, extensive knowledge and positive attitude towards work and life consistently inspired and motivated me.

I would also like to thank Prof. Jimmy Lin for his knowledge, inspiration, and encouragement. He provided a lot of valuable comments, insightful discussions and collaborations in part of this work.

My deepest gratitude also goes to Prof. Jimmy Huang from the York University for serving as my external examiner and providing many valuable suggestions. I would also like to express my appreciation to my other thesis examining committee members, Prof. Jimmy Lin, Prof. Gordon V. Cormack, and Prof. Mark D. Smucker for taking their time and effort to participate in my committee and giving their instrumental comments.

Many thanks to my great and lovely colleagues in the IR/PLG group for making such a wonderful environment in the lab. In particular, I would like to thank Adam Roegiest, Haotian Zhang, Bahareh Sarrafzadeh, Adriel Dean-Hall, Alexandra Vtyurina, Amira Ghenai, Dr. Gaurav Baruah, Dr. Ashif Harji, Dr. Aiman Al-Harbi and Dr. Matt Crane. I hereby want to thank my friends in the University of Waterloo, Cong Guo, Pei Pei, Yahui Chen and Lanjun Wang for their enormous support during my PhD program.

I want to thank my beloved family for their unconditional and endless love and support. My love and gratitude goes to my parents for always supporting me and taking good care of me. My most and deepest appreciation goes to my husband, the forever love of my life, Dr. Teng Wu. He always stands on my side, respects and understands me. Without his love, I cannot finish this work. I would also like to thank my lovely, sweet daughter Tanya, for all the happiness she brought to me and all the moving, warm moments she spent with me.

Last but not the least, I would like to acknowledge the Cheriton School of Computer Science and Dr. David R. Cheriton for the David R. Cheriton Graduate Scholarship, and the Government of Ontario and Bell Emergis for the Go-Bell Scholarship.

Dedication

I would like to dedicate my thesis to

my husband Dr. Teng Wu,

my parents Mr. Ming Tan and Ms. Li Song,

and my lovely daughter Tanya Wu.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Topic tracking in news streams became a hot research topic about twenty years ago [5, 8, 54]. Automatic algorithms were developed for discovering new topics in a stream of broadcast news or newswire. These topics were then used to follow the news events over time [55]. At that time, social media had not yet been created, but many of the issues exposed by the original research continue today.

Social media provides a platform for the general public to publish real-time news and comments on topics of personal and general interest. Social media is currently one of the most attention-attracting aspects of the internet. For example, it provides a way to learn opinions from other people about a given news event.

An important feature of social media is recency, with most information flowing through the social media closely related to the time when it is posted. For example, when someone reads an interesting news article on a website, he or she may want to follow this news event on social media for a period of time, perhaps learning about the progress of the event, receiving reports from different media sources, and reading comments from other people. Although it is possible to follow specific hashtags, for example, many of the most popular social media platforms currently do not generally support the continuous and real-time tracking of specific events and topics.

1.1 Problem Statement

Twitter[1], which started in 2006, is currently one of the most popular online social networking service. Users on Twitter can send and read short messages which are accessible all over the world. It has been reported that a large partition of tweets (short messages created by Twitter users) posted everyday are related to news events. Moreover, Twitter provides a friendly streaming API to external developers, which means that obtaining tweet data is fairly straightforward. Thus, tracking news topics obtained from news articles in Twitter stream becomes an interesting research problem.

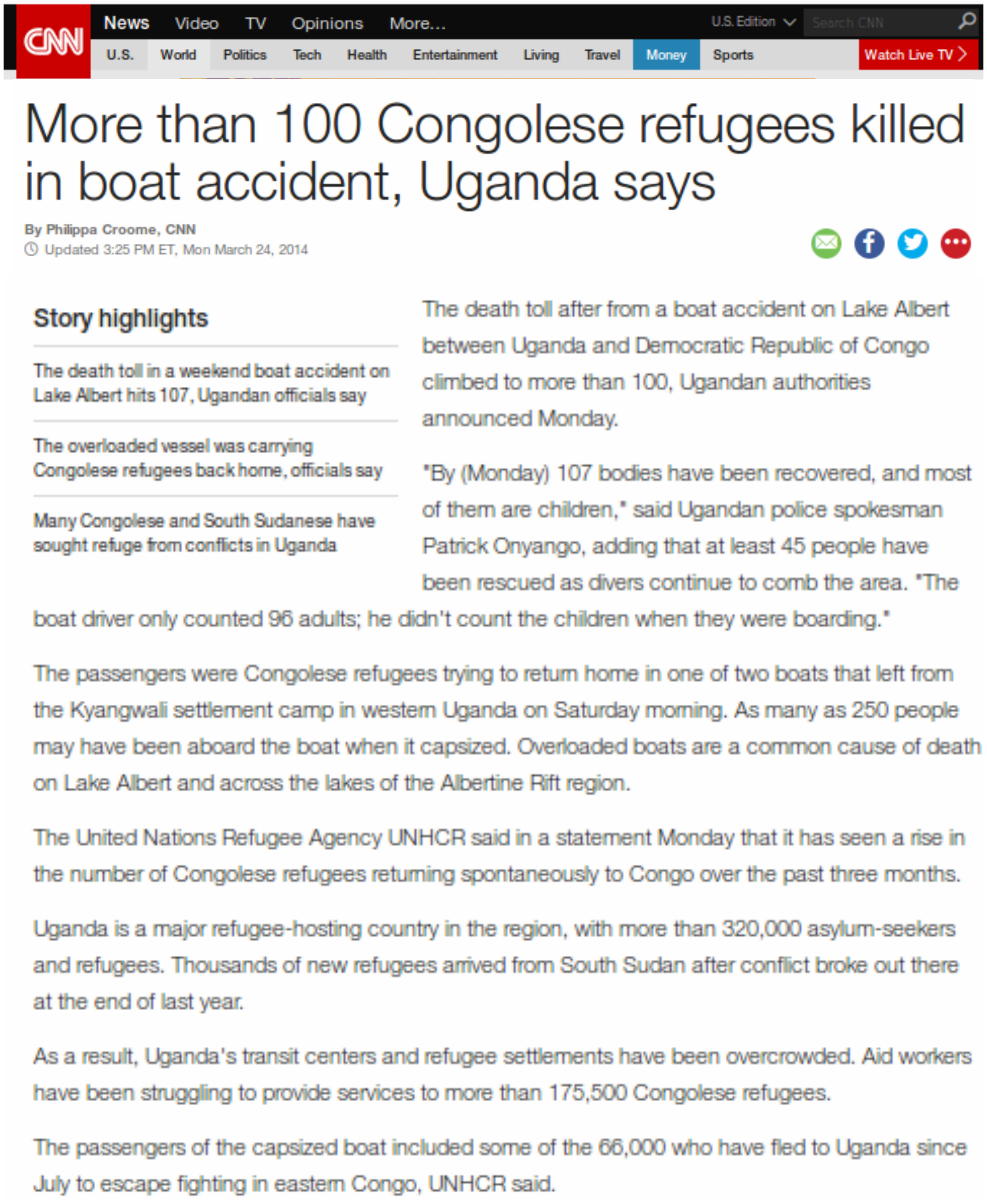
Unfortunately, due to the length limitation of tweets, which can be a maximum of 140 characters, tracking news events in the Twitter stream is more difficult than tracking them in general news streams, where full articles are available. Figure 1.1 is an example of news article. If we wish to generate a query to follow the topic of this article we have several choices, such as: the title of the news article, the content of the news article, key phrases extracted from the article, etc. However, if we wish to query Twitter (or some other services) to follow this topic, even the title may be inappropriate or too long for use with the commercial Twitter search engine, or another search engine. Ideally we would simply be able to indicate that we wish to follow the topic, and then begin to receive updates.

Once we have obtained a query, or some other indication of topical interest, from a user, we can monitor streams of social media posts, and discover posts that the user may care about. These updates might be sent to the user periodically, perhaps through email, or pushed immediately and directly to the user through notifications on his or her mobile device or desktop. For example, a user might be interested in poll results for the 2016 U.S. presidential elections and wishes to be notified whenever new results are published. In the following discussion, we define the problems in terms of five tasks. By solving these tasks, we are able to address the overall problem of tracking news events in social media streams.

According to the problems stated above, the following research contributions are reported in this thesis:

- **C1**: Improved understanding of word usage in social media.

[1] https://twitter.com



More than 100 Congolese refugees killed in boat accident, Uganda says

By Philippa Croome, CNN
Updated 3:25 PM ET, Mon March 24, 2014

Story highlights

The death toll in a weekend boat accident on Lake Albert hits 107, Ugandan officials say

The overloaded vessel was carrying Congolese refugees back home, officials say

Many Congolese and South Sudanese have sought refuge from conflicts in Uganda

The death toll after from a boat accident on Lake Albert between Uganda and Democratic Republic of Congo climbed to more than 100, Ugandan authorities announced Monday.

"By (Monday) 107 bodies have been recovered, and most of them are children," said Ugandan police spokesman Patrick Onyango, adding that at least 45 people have been rescued as divers continue to comb the area. "The boat driver only counted 96 adults; he didn't count the children when they were boarding."

The passengers were Congolese refugees trying to return home in one of two boats that left from the Kyangwali settlement camp in western Uganda on Saturday morning. As many as 250 people may have been aboard the boat when it capsized. Overloaded boats are a common cause of death on Lake Albert and across the lakes of the Albertine Rift region.

The United Nations Refugee Agency UNHCR said in a statement Monday that it has seen a rise in the number of Congolese refugees returning spontaneously to Congo over the past three months.

Uganda is a major refugee-hosting country in the region, with more than 320,000 asylum-seekers and refugees. Thousands of new refugees arrived from South Sudan after conflict broke out there at the end of last year.

As a result, Uganda's transit centers and refugee settlements have been overcrowded. Aid workers have been struggling to provide services to more than 175,500 Congolese refugees.

The passengers of the capsized boat included some of the 66,000 who have fled to Uganda since July to escape fighting in eastern Congo, UNHCR said.

Figure 1.1: A news article example.

- **C2**: Succinct query generation for linking different resources.
- **C3**: Real-time tracking social media stream and push notification algorithms.
- **C4**: Evaluation of push notification systems to mobile devices.
- **C5**: Maximized effectiveness difference rankings.

1.2 Research Tasks and Contributions

1.2.1 Improved Understanding of Word Usage in Social Media

The first task is research on an improved understanding of word usage within social media. Social media posts are created and edited by individual users. Thus, the language employed by users of social media tends to be highly informal. It includes slang, jargon, acronyms, typos, deliberate misspellings, abbreviation and interjections. These nonstandard usage of language brings problems when applying standard NLP tools and text retrieval techniques (Chapter 3).

We compare lexical usage between carefully edited prose and social media posts by using word embeddings. Mikolov et al. [115] proposed a novel neural network model to train continuous vector representation for words. The high-quality word vectors obtained from large data sets achieve high accuracy in both semantic and syntactic relationships [57]. The word embeddings are applied in this task to translate between the formal English of carefully edited texts, such as articles in Wikipedia, and the informal English of social media posts, specifically tweets.

The translation is achieved by generating a transformation matrix between two vector spaces representing two different language usage corpora. We identify the most different used words between the two corpora by computing the distances in the transformed spaces. Meanwhile, a rank overlapping method is applied as a validation method to verify the distance of most similar words ranking lists in two corpora.

1.2.2 Succinct Query for Linking Different Resources

The second task is given a news article of interest or a similar source document, a succinct query representing the content of the source document is expected to be generated. Furthermore, the query is intended to be able to find related material in a target resource such as social media. A specific and straightforward application of this succinct query method is to track a news article, news story in social media streams over a period of days, or even weeks (Chapter 4).

The problem is similar to a prior research problem: Query by Document [165]. However, the case of our second task is different from either traditional topic tracking task or query by document problem. In a traditional topic tracking task, an explicit query or a set of terms are provide for the tracking task. The tracking is produced in a fixed tracking platform without any change as time going on. The existing solutions for query by document problem extract key words and phrases from the source document, forming queries from these extracted terms. But they tend to produce large and complex queries.

We present an alternative approach for generating succinct queries from a candidate set of extracted key words. The succinct query comprises perhaps four or five terms. Starting with the candidate terms, we execute a series of *probe queries* over a sample of contemporaneous social media. By analyzing the results of the probes, and comparing them to the language of the source document, we rank the terms according to their ability to retrieve related material.

1.2.3 Live Tracking and Pushing Notifications

The third task assumes that an explicit user interest profile or a set of topic terms expressing user's information needs is provided, e.g., as created by the succinct query generation method above. A system that monitors the live social media stream (e.g., the live Twitter stream) automatically pushes updates to the user through notifications on mobile devices or desktops. The push notification system should control the frequency and volume of updates, avoiding indiscriminate and unwanted notifications to the users (Chapter 5).

The push notification problem is not strictly an information retrieval problem or an information filtering problem. Such systems should be able to identify relevant content from the streaming data. Moreover, they should avoid pushing duplicate contents to users, as well as controlling the frequency and time that they deliver the notifications.

We approach this problem by following the scenario settings of the TREC 2015 Microblog Track. This track provides pre-defined user interest profiles and an evaluation platform for participating systems. We demonstrate the crucial importance of controlling the volume of pushes and the need to avoid pushing non-relevant information. Through experiments over various strategies for establishing and maintaining thresholds for pushing, we achieve the best known results reported in the literature.

1.2.4 Ranking Push Notification Systems by Metrics

The forth task is a straightforward follow-up research problem to the third task. When comparing different systems for live stream tracking and push notification, what is a good metric for evaluating these different systems? We determine three significant components for evaluating these systems from the settings of the pushing notification scenario: *relevant* to the person's interests; *timely*, with pushes occurring as soon as possible after the actual event occurrence; and *novel*, providing non-duplicate information (Chapter 6).

The official evaluation metrics of TREC 2015 Microblog track were adapted from those of the TREC Temporal Summarization track. The major metric was expected latency-discounted gain (ELG) and the secondary metric was normalized cumulative gain (NCG). The gain of each push was determined by the three components discussed above: relevance to the user's information need, redundancy to the other tweets from the same "content cluster", and the time elapsed between a tweet's creation time and the time the tweet is delivered.

We analyze the official metrics, modify them with by making different assumptions about the user and then re-assess submitted systems. We present the novel and surprising finding that any number of reasonable evaluation metrics give rise to significantly different system rankings. We discuss and analyze why, tracing the issue to the handling of days for which there are no relevant tweets. We then examine the effects of latency penalties by considering the impact of different metric variants on runs submitted. Finally, we generate a framework of evaluation metrics using the same underlying contingency table,but placing different weights and penalties based on different user models.

1.2.5 Maximized Effectiveness Difference of Different Rankings

The fifth task concerns ranking similarity measurement which is used throughout this thesis work. We realize the unfortunate truth that most computation of information retrieval measures depends on the existence of explicit relevance judgement. It is said that, to measure the retrieved ranking list of a given query, we must know for each document in the list, whether or not it is relevant to the query. Creation of these judgements involves either substantial effort on the part of assessors or large volumes of interaction data, which limits the number of queries over which the measures may be computed (Chapter 7).

To compare different ranking results, we might also use standard rank correlation coefficient methods, such as Kendall's τ. In prior works focusing on the problem of comparing search result lists, Webber et al. [157] defined rank biased overlap (RBO) as a measurement for indefinite and incomplete ranked lists. However, we may want to measure differences between ranked lists under the assumptions of specified information retrieval effectiveness measure. We propose a family of distance measures, each directly derived from an associate information retrieval effectiveness measure, including RBP, nDCG, MAP and ERR.

This family of measures, called *maximized effectiveness difference* measures (MED), measures the maximum difference of two ranked lists in their effectiveness scores possible under a specified effectiveness measure. Computing this maximized effectiveness difference (MED) requires the solution of an optimization problem that varies in difficulty, depending on the associated measure. We present solutions for several standard effectiveness measures, including nDCG, MAP, and ERR. Through experimental validation, we show that MED reveals meaningful differences between retrieval runs. Mathematically, MED is a metric, regardless of the associated measure. Prior work has established a number of other desiderata for rank similarity in the context of search, and we demonstrate that MED satisfies these requirements. Unlike previous proposals, MED allows us to directly translate assumptions about user behavior from any established effectiveness measure to create a corresponding rank similarity measure. In addition, MED cleanly accommodates partial relevance judgments, and if complete relevance information is available, it reduces to a simple difference between effectiveness values.

Chapter 2

Background and Related Work

2.1 Topic Detection and Tracking

Topic Detection and Tracking (TDT) research develops automatic algorithms for discovering new topics in a stream of broadcast news or newswire, and then following these topics over time. The notion of a **topic** in TDT is defined as “a specific event or activity, along with all directly related events and activities” [55], where an **event** means “some special thing happening at some special time and special place”. An article in newswire is called a **story**, which is defined by the community as “a topically cohesive segment of news that includes two or more declarative independent clauses about a single event”. For example, the disappearance of Malaysia Airlines Flight 370(MH370) while flying from Kuala Lumpur International Airport to Beijing Capital International Airport on March 8th, 2014 is considered to be an event, whereas the investigation and search for wreckage that followed is considered to be the topic.

The TDT research program started from a pilot study between September 1996 and October 1997. The U.S. National Institute of Standards and Technology (NIST) provided an evaluation platform and ran a workshop every year between 1998 and 2004. The pilot study was designed and conducted by researchers from DARPA, Carnegie Mellon University (CMU), Dragon Systems, and the University of Massachusetts at Amherst (UMass). The purpose of the pilot study was to advance and assess state of art technologies in Information Retrieval (IR) and Information

Filtering (IF) for exploitation by TDT. Although the detection and tracking task are similar to standard IR and IF tasks respectively, IR and IF technologies can not solve all the TDT problems [168]. For the topic detection task, TDT systems lack any knowledge or understanding about topics. The systems have to automatically identify and detect new topics. However, IR systems usually have some kinds of initial user information need, or at least a query. For topic tracking task, topics are indicated by one to four sampled related news stories which contains implicit queries. While explicit profiles should be given for IF problems. The TDT systems were restricted to make online decisions, i.e. systems must decide about one story before looking at any subsequent stories [8] without any supervised feedback.

2.1.1 TDT Tasks

Under this program, TDT research was factored into five technical tasks [156]:

- **Story Segmentation Task**: Segment a continuous stream of text into its constituent stories
- **Topic Detection Task**: Detect and thread new topics, with no knowledge of the topics to be detected
- **Topic Tracking Task**: Follow topics to find additional stories about them
- **First-Story Detection Task**: Identify the first story of unknown topics in a stream of stories
- **Link Detection Task**: Determine whether two stories are topically linked

These tasks are not independent. For example, the link detection task is the foundation of the detection and tracking tasks. Indirect evaluation of the segmentation task is measured by topic tracking performance. In this section, we will focus on technologies for the topic tracking task only, which is closely related to our work.

2.1.2 TDT Corpora

Five corpora were used for the TDT research project: the TDT Pilot study corpus (TDT-Pilot), the TDT Phase 2 corpus (TDT2) for TDT 1998 Workshop, the TDT Phase 3 corpus (TDT3) for TDT 1999-2001 Workshop, the TDT Phase 4 (TDT4) corpus for TDT 2002 and 2003 Workshops, and the TDT Phase 5 corpus (TDT5) for TDT 2004 Workshop[1]. Some characteristics and statistical information for these corpora are provided in Table 2.1.

Corpora	Story Period	Languages(#Stories)	Broadcast News
TDT Pilot	Jul, 1994 to Jun, 1995	ENG(16K)	Transcripts
TDT2	Jan, 1998 to Jun, 1998	ENG(53.6K), MAN(18.7K)	Audio and Text
TDT3	Oct, 1998 to Dec, 1998	ENG(31.2K), MAN(12.8K)	Audio and Text
TDT4	Oct, 2000 to Jan, 2001	ENG(28.4K) ,MAN(27.1K), ARB(42.7K)	Audio and Text
TDT5	Apr, 2003 to Sep, 2003	ENG(278K), MAN(56.5K), ARB(72.9K)	None

Table 2.1: Summary of the TDT Corpora

The first TDT Pilot Corpora was created by the pilot study researchers. It contains approximately 16K of English news stories, which were either taken from the Reuters news service or manual transcriptions of broadcast news from CNN. The stories span a time period from July 1994 to June 1995. 25 events of different types were manually labelled as annotated topics for detection and tracking. Each story in the corpus was flagged YES/NO/BRIEF for each event.

The TDT2 corpus and TDT3 corpus were collected daily from six news sources, including two newswires services, two radio news programs and two television news programs, such as New York Times News service, VOA world news and CNN Headline News [54]. Both corpora contain news stories in two languages: American English (ENG) and Mandarin Chinese (MAN). For TDT2, 100 English topics were completely annotated, and 20 of them were also labelled as Mandarin topics; for TDT3, 240 topics were annotated. All English broadcast transcripts were created manually or through automatic speech recognition (ASR), and all Mandarin stories were automatically translated by machine translation tools.

The TDT4 corpora and TDT5 corpora were collected daily from 20 news sources and 15 news sources, respectively. They contain stories in three languages: American English, Man-

[1]All data can be obtained from the Linguistic Data Consortium (https://catalog.ldc.upenn.edu)

darin Chinese and Modern Standard Arabic (ARB). A total number of 100 topics were manually labelled for TDT4, and 250 topics for TDT5.

2.1.3 Successful Techniques for Topic Tracking

For the topic tracking task, researchers were interested in the following key research issues:

- Topic representation models
- News story representation models
- Similarity between news stories and topics
- Threshold setting for detection and filtering

Pilot Study

In the pilot study, the notion of a *topic* was narrowed down to an *event*, which was defined by N_t stories that discuss the event [5], where $N_t = 1, 2, 4, 8, 16$ were considered. Different participating groups attempted different approaches. It had been shown from the evaluation that simple IR techniques could achieve high quality results, although to reduce errors, more work was needed.

The UMass group used methods based on Information Filtering [5, 8]. N_t positive and $100N_t$ negative training stories were exploited to generate a short query for representing the event and a threshold for comparison between story and that query. The queries constructed in this way were represented by a tf-idf weighted vectors. Dimensions in these vectors were either the top 10 to 100 most commonly occurring words or extracted nouns and noun phrases appearing in positive training stories. **Adaptive tracking** was first tested by UMass to handle drifting queries using methods similar to pseudo-relevance feedback. If a story was selected for tracking, it would be added to training story set and queries would be regenerated.

CMU treated the task as a binary classification problem. Two classifiers k-Nearest Neighbour (kNN) and Decision-Tree Induction were developed. They converted each story (including training stories) into a vector and compared vectors by cosine similarity. An incoming story would be voted YES/NO by k nearest training vectors or by one positive and one negative nearest training vector. The voting decision was made by a pre-trained threshold. Features for Decision Tree classifiers were selected with maximal information gain (IG).

Dragon's event tracker was derived from their segmentation algorithm, which preformed segmentation and topic assignment simultaneously [161]. One hundred background topic models were build from an external corpus, by clustering the corpus using k-means algorithm and unigram statistics. An event language model was built from N_t training stories. Additionally, it was smoothed by the mixture of the best approximated background topic models. A Hidden Markov Model (HMM) approach was utilized to model probabilities of transition or duration of topics.

Between 1998 and 2004

During the evaluation years for 1998 to 2004, many research groups exploited different technologies to achieve better performance, some of them reduced costs, while the others brought occasional improvements, or even harmed the total detection and tracking. As in traditional IR applications, news stories were mainly represented by two types of models: vector space models (VSM) and language models (LM).

Although there are some limitations of the vector space model, it performed better than other alternate techniques like language modelling and machine learning. With the vector space model, each document was represented by a vector of weighted features. These features can be single terms, phrases or entities. The weights can be either statistical TF-IDF family weights or binary weighting (one of terms present and zero for terms absent). UMass systems [6, 7, 36] persisted with a vector model for representing stories and topics. Each dimension in the vector was a single stemmed word, and the weighting scheme was raw term frequency value multiplying the InQuery engine's IDF component value, which was:

$$idf_{comp} = \frac{log(N/df)}{log(N+1)} \tag{2.1}$$

The IDF component was the logarithm of the inverse probability of the term in a collection, where N denotes the total number of documents in the collection. The dimension of each vector was chosen to be 1000, which contained almost the full story. Similarly, CMU [24, 163, 164] defined their vector space model weights to be:

$$w(t, \vec{d}) = \frac{(1 + \log_2 tf(t, \vec{d})) * \log_2 (N/n_t)}{\|\vec{d}\|} \tag{2.2}$$

where, $w(t, \vec{d})$ is the weight of term t in document $\vec{d}$; $tf(t, \vec{d})$ is the within document term frequency; N is the number of documents in training set and n_t is the number of documents in training set which contains t; $\log_2 (N/n_t)$ is the inverse document frequency (IDF); and $\|\vec{d}\|$ is the L2-norm of document vector $\vec{d}$. In this model, topics were also represented by vectors, where each topic's vector was an average of the vectors of the stories about that topic (i.e. cluster centroid). If there were both possitive and negative examples of each topic, the Rocchio relevance feedback algorithm(formula 2.3) was also applied [36, 163].

$$\vec{c}(D, \gamma) = \frac{1}{|R|} \sum_{\vec{y} \in R} \vec{y} + \gamma \frac{1}{|S_n|} \sum_{\vec{z} \in S_n} \vec{z} \tag{2.3}$$

where $\vec{c}(D, \gamma)$ is the representation of the topic, D is the entire training set, R is the positive examples in training set D, $S = D - R$ is the negative examples in D, and S_n contains n most similar negative examples, γ is the weight for negative component.

Cosine similarity shown to be stable and provided substantial benefits when measuring similarity under vector space model. Other functions like weighted sum, cross-entropy and Kullbach-Leiblar divergence were tested and found to be worse and less stable [7]. Thus, cosine similarity was widely used by those groups applying vector space models.

Like the vector space model, language models represent another popular information retrieval model. Some groups participating in TDT developed language model approaches. Dragon's tracking system [160, 162] built language models for each topic, each story, and the background material. When comparing the tracking score of an incoming story to a topic, they defined it as the log ratio between the probability that the story was generated from the topic and the probability that the story was generated by the background model. UMass extended their relevance model, which was a modified language model technique to achieve success in TDT tasks [87]. Relevance models were first defined by Lavrenko and Croft [88] to estimate the probability of observing a word in a document that was relevant to the query in a collection of documents. Without any relevance judgement, typically the probability was approximated by co-occurrence between the query and the word. Suppose Q is a query, and each d is a document in collection C, the relevance model for estimating a word w relevant to Q is :

$$P(w|Q) = \sum_{d \in C} P(w|d)P(d|Q) \tag{2.4}$$

where $P(w|d)$ denotes the probability of word w in document d, and $P(d|Q)$ denotes the posterior probability of producing the query, which is computed as:

$$P(d|Q) = \frac{P(d)\prod_{q \in Q} P(q|d)}{\sum_{d' \in C} P(d')\prod_{q \in Q} P(q|d')} \tag{2.5}$$

In the TDT task, a story was used as a query, i.e., $S = q_1 q_2 ... q_k$, where each q_i was a word in the story. Training documents for each topic was considered as a collection. Thus, tracking score between a topic and a story can be computed using equation 2.4.

Adaptive Tracking was also tested in the pilot study. Stories in the stream that had been considered as track-able with respect to the topics were added to adapt the representation of the topics. However, since no relevance judgement were provided, this approach would bring risks of false alarms. If immediate relevance feedback was assumed to provide by users, there would be no danger of adapting.

Since three different language sources were provided by TDT corpora, multilingual topic tracking was also a popular research area. Larkey et al. [86] built a language model for each language and proved by experiments that building language-specific topic models separately for each language performed more effectively than only training English topic models with machine translated scripts.

After 2004

After 2004, NIST ended the TDT evaluation project. Some groups, although not as many as during the evaluation period, are still working on solving TDT problems. Utilizing more news specific features, such as temporal information [75, 90] and location information [69] has been proved to help TDT tasks. New models such as probabilistic topic modelling, online LDA [9] were applied to track topics over time. TDT technologies were transferred and modified to other domains, such as email [38], web video [100], and news web pages [119]. Also social media, especially website Twitter has become an emerging news media platform. Thus moving TDT to social media platform is a straight forward research trend [17, 99].

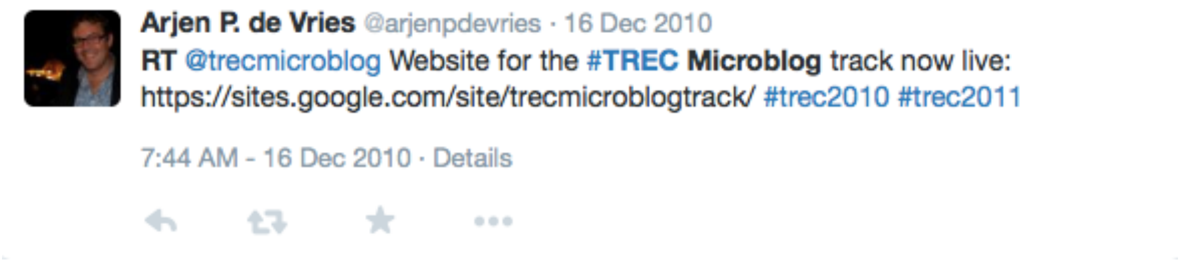
2.2 Topic Tracking and Filtering in Microblog Stream

2.2.1 TREC Microblog Track

The Microblog track was first introduced in TREC 2011, and was still running in 2016. The major goal of this task is to improve technologies for satisfying user information needs in microblog environments such as Twitter[2]. Users in the Twittersphere post messages (called *tweets*) which are strictly limited to up to 140 characters. Some well-defined terminology is unique to Twitter and tweets: the term "RT" is short for *retweet*, which means that a users has re-posted the content of another user's tweet with or without their own comments;the character '@' in a tweet followed by a user's screen name is mentioning that user; the character '#' followed by a word or sequence of words (which may or may not be words appearing in dictionaries) is called a *hashtag*. Hashtags are created organically by users for marking keywords or topics in tweets [85]. Hastags typically appear as new topics emerge, and are then included in future tweetes on that topic.

Figure 2.1 shows an example tweet from the user "@arjendevries", which mentions the user "@trecmicroblog", with tags indicated that it is related to "#trec2010" and "#trec2011'. The average length of tweets is around 10 words, which is much shorter than normal documents in other information retrieval tasks. Also due to the hard length limitation, users are more likely to use special terms such as abbreviations, shortened words and slang when generating tweets, which leads to highly varied document quality. Traditional IR technologies often cannot be directly applied to tweets successfully [52]. Thus, new methodologies for both searching and evaluation are examined in the TREC Microblog Track.

[2]Twitter Statistics: http://www.statisticbrain.com/twitter-statistics/


Arjen P. de Vries @arjenpdevries · 16 Dec 2010
RT @trecmicroblog Website for the #TREC Microblog track now live: https://sites.google.com/site/trecmicroblogtrack/ #trec2010 #trec2011

7:44 AM - 16 Dec 2010 · Details

Figure 2.1: A tweet example.

Microblog Adhoc Search Task

Tasks in the Microblog track are still evolving and improving. Since 2011, three tasks were conducted: an adhoc search task, a filtering task and a tweet timeline generation task. The adhoc search task ran between 2011 and 2014. In this task, a user's information need was defined as a query, along with a specific query time. Systems were required to search for the query and find most relevant and recent tweets posted up to that time. An example of a TREC Microblog track topic is as follows:

```
  <top>
  <num> Number: MB01 </num>
  <title> Wael Ghonim </title>
  <querytime> 25th February 2011 04:00:00 +000 </querytime>
  <querytweettime> 3857291841983981 </querytweettime>
 </top>
```

According to the task definition in the overview reports [96, 123, 140], the `<querytweettime>` tag contains the timestamp of the query in both human and machine readable ISO standard formats, while the `<querytweettime>` tag contains the timestamp of the query in terms of the chronologically nearest tweet id in the corpus. The `<title>` tag indicates the information need for the topic. Unlike for some TREC tasks, no narrative or description tags were provided in these topics.

Corpora Most tweets are visible to the public on Twitter's web site. Twitter also provides friendly searching and streaming APIs that any Twitter user can access to read and write Twitter data. Due to Twitter's term of service, any reproduction of tweets is forbidden. Therefore, the way of obtaining a corpus of tweets for the track is dramatically different from other TREC tracks. There were two corpora provided by the microblog track organizers till now: the Tweets2011 corpus (used in TREC 2011 and 2012) and the Tweets2013 corpus (used in TREC 2013 and 2014).

The Tweets2011 corpus contained a list of unique identifiers (tweet ids) of about 1% sample of tweets from January 23, 2011 to February 7, 2011. The total number of tweet ids was approximately 16 million. Each tweet id can be mapped to a URL at *twitter.com*. Participants can download a copy of the corpus by a given tool provided by TREC organizers. When downloading the corpus, each participated group might obtain different tweets since users might delete their tweets or restrict their tweets to private only. The Tweets2013 corpus was crawled by the organizers through Twitter public streaming API between February 1 and March 31, 2013. There were approximately 243 million tweets.

Normal Process Typical steps for adhoc search on the Twittersphere are similar to those of other traditional IR search tasks: 1) data pre-processing; 2) with a given text scoring function, retrieve original tweets; 3) query expansion or document expansion; 4) re-search tweets by expanded terms; and 5) re-rank the search results. The filtering task could be achieved by exploiting the relevance score computed by the retrieval method and filtering out low relevance results with thresholds. In 2014, groups who submitted experimental runs to the tweet timeline generation task were also required to submit adhoc search task results, which meant that the TTG task also included an adhoc search task. In this subsection, we introduce some approaches and models used by top ranked research groups for these different steps.

Pre-processing Data By definition, retweets and tweets written in a language other than English are judged as not relevant for the Microblog Track. As a result, most groups removed retweets and non-English tweets in a pre-processing step. Some groups removed retweets after the first round of retrieval before query expansion. Retweets were easy to detect, since these tweets were either started with 'RT' or returned a 302 status code when crawled by HTML tool. Non-English tweets could also be filtered out after first retrieval or added as a feature in a learning-to-rank re-ranking step. Groups such as PRIS [93] determined if a tweet was English or non-English by applying an English vocabulary word list. If more than half the words in a tweet were English words, the tweet would be kept as English tweet. Other groups like Clarity, HIT and University of Indonesia [52, 61, 102] utilized language classifiers or identifier tools to detect non-English tweets. Other typical NLP technologies popular in IR tasks were also attempted by some groups, such as stopword removal and stemming.

Text scoring The difficulty of defining a function for measuring relevance between queries and tweets is exacerbated by the shortness and varied quality of tweets. Some search engines' built-in models were directly used for this task, such as the Markov random field (MRF) retrieval model [111], built into Indri[3], which was used by several groups [93, 110, 178]. MRF can combine text matching features from term, phrase, and proximity-based matching, measuring dependencies between them. TF-IDF based methods have been widely demonstrated to be effective and efficient for many IR tasks. The Clarity group [52] modified the Okapi BM25 model based on the short nature of tweets. They removed the penalty against longer documents from the original BM25 model (equation 2.6)by setting the parameters $k1$ and b to 0. Here tf_t is either 0 or 1 representing absent or present in the tweet respectively.

$$Score_{bm25}(q,d) = \sum_{t \in q} log(\frac{N - df_t + 0.5}{df_t + 0.5}) * \frac{(k_1 + 1)tf_t}{k_1((1-b)) + b\frac{dl}{avdl}) + tf_t} \tag{2.6}$$

The Kullback-Leibler divergence (KL-div) retrieval model is another typical language modeling approach for retrieval tasks. K-L divergence measures the information distance between two probability models. Group HIT [61] used this model to estimate the difference between a query model and a document model. Similarly, group FUB, IASI-CNR, UNIVAQ [10] exploited it as term-message weighting.

Query Expansion The most common approaches for dealing with vocabulary mismatch problems deriving from short documents are query expansions and document expansions. About one tenth of the tweets contain at least one web link in its content. Thus, we might be able to use web page contents from links appearing in tweets to expand tweet content, as used by group

[3] http://www.lemurproject.org/indri/

PRIS [93]. For query expansions, several different models were used by different groups, such as the Latent Concept Expansion [110], parameter free Bo1 model built into Terrier [10], a word activation force algorithm and term similarity metric based on electric resistance network [93], top ranked snippets or titles from Google search results [102, 178], synonym of query terms from WordNet [171] and pseudo-relevance feedback [52, 61]. Key parameters in this step were the number of tweets selected to process query expansion (one to hundreds) and the number of expanded terms (one to dozens).

Re-ranking After these processes, usually a re-ranking or re-weighting process would be performed by the systems. Two kinds of methodologies were reported successfully when processing this step: learning to rank methods and temporal-based methods. Widely used features for learning to rank methods can be categorized into three types: text-based features, temporal-based features and non-text based features.

- **Text-based features**: tweet relevance scores, ratio of terms that are out-of-vocabulary, key word ratio, expanded word ratio, named entities
- **Non-text based features**: has-url, has-hashtag, is-reply, retweet-count, tweet-length, user-followers-count
- **Temporal-based features**: time difference from query time

Real-time Filtering Task

TREC 2012 A filtering task was first run at TREC 2012 as the reverse task to the adhoc search task, which was modelled on the TREC 2002 adaptive filtering task. The information need was also defined as a query and a specific time, with participating systems required to filter relevant tweets in stream of tweets after that time. The organizers reused topics from the TREC 2011 microblog track and re-tagged the topic files. The original `<querytweettime>` from the adhoc searching task was provided as the endpoint for the collection that systems receive before the streaming started, which was named as `<querynewesttweet>`. An additional `<querytweettime>` tag indicated the earliest known relevant tweet for each topic in the corpus. Systems would judge relevance of each tweet from the query tweet time to the query newest time one-by-one, and decide whether or not to show it to the user. If a system decided to show a tweet to the user, it might be able to obtain an relevance feedback to improve subsequent judgements.

Each topic in the task was represented by a query and a single start tweet. Since queries consisted of several key words and tweet was much shorter than a typical document such as news story, most participating groups followed a process similar to the first three steps of the adhoc retrieval task: 1) pre-processing tweets, 2) building text-score function, and 3) expanding queries and tweets. Instead of re-ranking as the fourth step, thresholds were set for adaptive filtering.As it turned out, appropriate setting of these thresholds was a key to success in the task.

The HIT group [61] applied the same retrieval model as they used for their adhoc search runs over the tweet collection ahead of the start tweet time for each topic. They retrieved the top m tweets as their relevant set. The filtering threshold for determining whether an incoming tweet was relevant or not was the retrieval score of the m-th tweet in the relevant set. They tested different ways of setting the parameter m, such as fixed, dynamic or combined fixed with dynamic ms. Manually adjustment of the thresholds also achieved success by the PRIS group [171]. They manually initiated the threshold for each topic. While running the filtering system, they manually label the correctness of automatic judgements. Later they would adjust the threshold for each topic according to the ratio of the number of their correctly labeled or wrongly labeled tweets. The University of Glasgow group [94] trained their filtering thresholds and explored an approach to adjust them according to various features, such as whether the tweet contained URLs and/or hashtags.

TREC 2015 TREC 2015 Microblog Track included a completely different task from previous years, which was the real-time filtering task. The goal of this task was to monitor the real Twitter streaming data and determine whether or not push each tweet to a user. Each user's interest profile was given in the format of a traditional TREC topic, as shown in the following example:

```
<top>
<num> Number: MB10001
<title> crossword puzzle tournaments
<desc> Description:
```

```
Return announcements of and commentary regarding
crossword puzzle tournaments.
<narr> Narrative:
The user likes to do crossword puzzles and intends to
participate in upcoming crossword puzzle tournaments.
She wants to see any Tweets that relate to a tournament:
Tweets that announce a tournament or give logistical
information; Tweets about a tournament from its participants
including Tweets that express anticipation of the tournament
or traveling to/from the tournament; Tweets that comment
on the quality of a tournament; etc.
</top>
```

Two task scenarios were studied:

- **Scenario A**: Push notifications on a mobile phone. Participating systems should identify interesting tweets based on the user's interest profile(topic), and determine whether or not push a notification on the user's mobile phone. Such notification is expected to be triggered within 100 minutes after the tweet is created. A maximum of 10 tweets could be delivered by a system to a single user per day.
- **Scenario B**: Periodic email digest. Participating Systems should identify tweets based on the user's interest profile, and aggregate them into an email. The email should be periodically sent (i.e., every day) to the user.

Scenario A is a real-time filtering task, but it does not require on-line decision. It means that participating systems do not need to decide whether or not push notification for a tweet before seeing the subsequent tweets. A 100-minute latency time is allowable. Thus, in addition to the normal retrieval processes, pushing strategy is also significant. However, Scenario B is more like an adhoc retrieval task based on a one-day tweet collection.

TREC 2016 TREC 2016 Real-Time Summarization (RTS) Track [4] took place from August 2, 2016 00:00:00 UTC to August 11, 2016 23:59:59 UTC. Similarly to TREC 2015 Microblog

[4] http://trecrts.github.io/TREC2016-RTS-guidelines.html

Track, all participating systems were asked to listen to the Twitter sample stream using the Twitter streaming API themselves and perform the evaluation tasks in real time. The user interest profiles were designed in the same format as previous track. They were ad hoc style TREC topics.

There were also two scenarios in 2016 RTS track: push notifications and periodic email digest. The periodic email digest scenario (Scenario B) was almost the same as TREC 2015 Microblog track. Systems identified up to 100 ranked tweets per day per interest profile. It was expected that systems computed the results in a relatively short amount of time after each day ended. The final submission was uploaded to the evaluation platform after evaluation period ended.

The push notification scenario (Scenario A) was designed differently from Scenario A in TREC 2015 Microblog track. In this scenario, content that was identified as relevant by a system based on the user's interest profile would be pushed to the TREC RTS evaluation broker via a REST API in real-time. These notifications were immediately routed to the mobile devices of a group of human assessors. Two different evaluation judgments were made for the push notifications: real human assessors with mobile devices and NIST assessors at the end of the evaluation period.

2.2.2 Topic Tracking and Filtering in Twitter Streams(Beyond TREC)

Topic Tracking in Twitter Streams

Since Twitter has become a new type of news media, both news agents and readers read, tweet and retweet news from Twitter. Thus, it is a straightforward idea to import topic tracking research to the Twitter environment. Similar to traditional topic tracking, topic tracking in Twitter also faces the problem of insufficient initial data, i.e. the cold start problem, as well as the chance variation of topics, i.e. topic drift. In addition to these common issues, tweets also have their own specific features that leads to more topic tracking issues. Due to the length limitation of a single tweet (140 characters), it is more difficult to track topics in tweets than in traditional topic tracking of news articles. Terms seldom repeat in a tweet, which often makes the traditional family of TF-IDF weighting methods fail. Moreover, the quality of tweets vary based on different user styles, which makes it even harder to extract topics from tweets.

Lin et al. [99] manually selected 10 hashtags based on popularity as topics, and built a topic-specific language model for each topic. The language models were then exploited to compute relevance scores of incoming tweets in tweet stream and filter out non-relevance tweets. Four different smoothing methods were examined on topic-specific language models with background models to solve the sparsity zero-probability problem. The selected topics were all stable, coherent and non-advance topics over time, which means the topic drift problem was skipped in this research.

Duan et al. [44] proposed a graph-based approach, in a graph optimization framework, for classification of tweets into six broad topics: entertainment, politics, science and technology, business, lifestyle, and sports. Related tweets, the ones that share either the same hashtag or URL, were used to enrich the representation of each tweet and adaptively update the trained model using the hashtags as a surrogate for user feedback.

Hong et al. [63] and Fei et al. [51] followed the work of Lin et al., with an additional consideration of topic drift over time. Hong et al. developed a more complex background model with a foreground model to handle the cold start problem They adapted semantic features and quality features to build a content model, which enriched document content and measured the quality of tweets. A fixed-width temporal sliding window was adopted to extract tweet features as well as providing pseudo-relevance feedback. The feedback model was then proved by experiments to effectively solve the topic drift issue. However, the size of the window was fixed which may not portray the drift properly as the emerging of topic drift was unexpected and irregular. Fei et al. [51] proposed a cluster-based subtopic detection algorithm to deal with topic drift over time.

Magdy et al. [107] proposed an unsupervised approach for tracking short messages from Twitter that were relevant to broad and dynamic topics, which initially obtained a set of user-defined fixed accurate (Boolean) queries that covered the most static part of the topic and updates a binary classifier to adapt to dynamic nature of the tracked topic automatically. However, it was not easy to find such user-defined queries to capture emerging subtopics of a broad topic.

Dan et al. [39] aimed at following tweets that are related to specific TV shows. They proposed a bootstrapping approach that uses domain-knowledge and a two-stage semi-supervised training of a classifier. Chen et al. [32] realized that keyword-based Boolean filtering was not effective for tracking tweets that express customer opinions about commercial brands (e.g., Delta Airlines). They leveraged crowd-sourcing resources to label tweets that satisfy predefined queries to train a supervised binary classifier. Nishida et al. [122] detected changes in word probabilities over time in a probabilistic classification approach. Hashtags on baseball teams or television networks were used as labels for training the classifier.

Real-time Filtering in Twitter Streams

Phuvipadawat and Murata [126] focused on tracking and detection of breaking news using predefined search queries, e.g., #breakingnews and "breaking news". They adopted an unsupervised filtering approach based on clustering similar incoming tweets with an emphasis on proper nouns, and significant nouns and verbs, to track and adapt to developing stories. Sriram et al. [141] used manually-labelled tweets to train a naive Bayes classifier to classify tweet streams into general broad topics such as news, opinions, events, deals, and private messages.

Albakour et al. [2] proposed an effective approach to deal with the sparsity and drift for real-time filtering in Twitter. In their approach, query expansion based on pseudo-relevance feedback was used to enrich the representation of user profile which improved the filtering performance a lot. Furthermore, a set of recent relevant tweets were utilized to represent the users' short time interests and tackle the drift issue. Three strategies were introduced to decide the size of tweets set: arbitrary adjustments, daily adjustments and event detection strategy based on CombSum voting technique and Grubb's test. The event detection significantly improved recall at the cost of a marginal decrease in the overall filtering performance.

Zhao and Tajima [175] considered the real-time filtering problem as a retweet recommendation problem. A special type of users, which were named as "Portal accounts", on Twitter was defined. Instead of posting original tweets, these accounts retweet tweets which were useful to their followers. Portal accounts should retweet a certain number of tweets and could not cancel the retweets. They considered the problem as a multi-choice secretary problem [78]. Four different methods were proposed and tested: history-based threshold algorithm, stochastic threshold algorithm, time-interval algorithm and every k-tweets algorithm.

Other Related Work

Comparison with other search Efron [47] defined two types of search in microblog systems: asking for information and retrieving information, which were corresponding to online Q&A and adhoc search in IR research problems respectively. Several key problems in microblog retrieval were described, including sentiment analysis and opinion mining, entity search, user-generated metadata, authority and influence, temporal issues. None of these problems was novel in IR research area. Approaches for solving these problems in previous IR research, as well as initial attempts to solve them in a microblog environment were reviewed. In the conclusion, possible applications of geographical information in microblog search and important issues of microblog search evaluation (relevance, corpora and recency) were presented. A text searching task is defined as matching *query* against a set of *documents* (tweets in microblog search).

To better understand microblog search, Teevan et al. [149] and Zhao et al. [174] analyzed characteristics of twitter queries and tweet collections respectively. Teevan et al. [149] compared differences between motivation, behaviour and results of microblog search and web search. From an analysis of a questionnaire study and millions of query logs, queries issued to twitter were found to be seeking more temporally and personally related information. Users tended to search shorter, more repeated and more popular queries to monitor content. Search results from twitter search engines included information about more social content and events.

A Twitter corpus and a traditional news article corpus were compared by Zhao et al. [174]. They exploited topic models to discover that the distributions of different topic categories and types were different between twitter corpus and traditional news corpus. The original tweets were found to focus more on personal life and pop culture, rather than world events, especially more celebrities and brands, while the retweets helped to spread important news and widely covered world events.

Key Information Extraction Extracting key information from documents is crucial for retrieval tasks. Since the length of tweets is much shorter than traditional documents and not every tweet contains useful informationand key information extraction is more challenging in a microblog environment. Hashtags as user generated labels can be a type of natural keywords.

Efron [46] demonstrated that hashtags benefited from query expansion during the relevance feedback process, where hashtag retrieval was treated as entity search.

Keywords and key phrases extraction methods were proposed in the work of Zhao et al. [173]. Keywords were ranked by a modified topical PageRank method, and key phrases were ranked by a principled probabilistic phrase ranking method. The natural language processing community attempted to apply traditional NLP tasks to twitter dataset, such as Part of Speech tagging (POS) and named entity recognition (NER) [101, 129]. Lacking sufficient context information, it is hard to identify entities and entity types in tweets. Thus, traditional POS and NER tools performed poorly on twitter corpora.

Liu et al. [101] combined a K-Nearest Neighbour classifier with a linear Conditional Random Fields model under a semi-supervised learning framework. By testing on manually annotated tweet dataset, their model outperformed the dictionary look-up baseline system.

Temporal and Geographical Information Utilizing temporal and geographical information to improve relevance in microblog retrieval has been attempted by some groups during The TREC microblog track. More research work outside TREC also discovered effective methods to take advantage of these metadata from tweets.

Choi and Croft [33] exploited the temporal distribution of users' retweet behaviour to propose a model for selecting a time period for query expansion. Experiments on TREC collections indicated that the time-based relevance model contributed to improving retrieval performance. This approach is derived from the temporal cluster hypothesis, which suggests that relevant documents tend to share similar temporal features,

Efron et al. [48] proposed methods to estimate temporal density of relevant documents and explored temporal feedback to benefit tweet search. A language model based on query-likelihood generated a ranked list of relevant documents. A log-linear model combined probability of relevance given temporal features with the word-based probability of relevance.

Several research papers [62, 79, 80] demonstrated that geography-aware topic models can discover patterns of language usage among different geographical locations. It has been found that queries were often geographically contextualized. Taken into account a tweets' geographical context, vocabulary mismatch problems in microblog retrieval can be solved through expanding tweets with related terms.

Hong et al. [62] assumed that each tweet was generated from three types of language models: a background language model, a per-region language model and a topical language model. For each tweet, latent region and location of this tweet was first chosen, then a topic was selected dependent on both the region and the author of the tweet. Different topical patterns across different geographical regions were observed from their experiments.

Kotov et al. [79, 80] expanded a geographically-aware extension of the Latent Variable Model (LVM) and the Latent Dirichlet Allocation (LDA) topic modelling approach to generate geographically special topics on language model based retrieval framework respectively.

Other works, such as Yuan et al. [169, 170] and Ma et al. [105, 106] recommended and searched hashtags for Twitter users.

2.3 Evaluation Methodology of Tracking and Filtering Tasks

2.3.1 TDT Evaluation

All five TDT tasks can be treated as detection tasks. In a detection task, a labelled target story can be correctly detected as a system response target, or can be missed as an error called *missed detection*. A labelled non-target story can be correctly detected as a system response non-target, or can be falsely detected as an error called *false alarm*. A contingency matrix of detection system responses is shown in Table 2.2 [55].

		Reference Annotation	
		Target	Non-Target
System Response	YES (a Target)	Correct	$False\ Alarm$
	NO (Not a Target)	$Missed\ Detection$	Correct

Table 2.2: Contingency matrix of detection system responses

The performance of a TDT system is measured based on costs to missed detection and false alarm errors exploiting two approaches: the detection cost function and the detection error tradeoff curve (DET). The detection cost function is defined as the following:

$$C_{Det} = C_{Miss} * P_{Miss} * P_{Target} + C_{FA} * P_{FA} * P_{Non-Target} \tag{2.7}$$

Here, C_{Miss} and C_{FA} are the costs of a missed detection and a false alarm, respectively. They are pre-defined parameters for the task. P_{Miss} and P_{FA} are the probabilities of a missed detection and a false alarm respectively. They are determined by system performance, where $P_{Miss} = \frac{\#Missed\ Detections}{\#Labelled\ Targets}$ and $P_{FA} = \frac{\#False\ Alarms}{\#Labelled\ Non-Targets}$. P_{Target} is a corpus statistical prior probability, specified by the task. It represents the quantity of stories about a particular topic in the training data. $P_{Non-Target}$ is also a prior probability, which equals to $1 - P_{Target}$. To visualize the system's tradeoff between P_{Miss} and P_{FA}, DET can be plotted by sweeping a threshold through the system's space of decision scores.

2.3.2 TREC 2012 Microblog Filtering Task Evaluation

TREC 2012 Microblog filtering task was built on the TREC 2002 adaptive filtering task [131], and used the same evaluation metrics: linear utility and Van Rijsbergen's F-measure. Filtering systems are expected to make binary decisions (relevant or non-relevant) with respect to each document in the filtered stream for each topic. Thus, the selected documents are considered as an unrated set in these two filtering tasks.

F-measure is a trade-off measure between precision and recall, combining them with a parameter β controlling the emphasis weighting of precision and recall equation 2.8. Here, precision denotes the number of documents retrieved from the collection, and recall is the total number of relevant documents within the collection. $\beta = 1$ means precision and recall are balanced; $\beta = 0.5$ corresponds to an emphasis on precision. A choice of $\beta = 0.5$ is used by previous TREC and TREC 2012.

$$F_\beta = \frac{(1 + \beta^2) * precision * recall}{\beta^2 * precision + recall} \tag{2.8}$$

Linear utility scores all retrieved documents by giving two points of reward to relevant ones and one point of penalty to the non-relevant ones:

$$T11U = 2 * |relevant\ tweets\ retrieved| - 1 * |non-relevant\ tweets\ retrieved| \tag{2.9}$$

Filtering according to a linear utility function is equivalent to filtering by estimated probability of relevance, i.e. to retrieve if $P(rel) > 0.33$ [140]. Since $T11U$ is unbounded, simple averaging of values across all the topics is impossible. The topics with larger retrieved sets will be weighted more than the ones with smaller retrieved sets. Hence, the utility value of a topic is normalized by the theoretical maximum possible utility for this topic ($MaxU = 2 * |relevant\ tweets\ retrieved|$) and bounded by an arbitrary minimum normalized utility value ($MinNU = -0.5$). Therefore, we have:

$$NormU = \frac{T11U}{MaxU}$$
$$T11SU = \frac{max(NormU, MinNU) - MinNU}{1 - MinNU}$$

2.3.3 TREC 2015 Microblog Filtering Task Evaluation

For TREC 2015, all submitted tweets by the participating groups were pooled and judged by NIST assessors as in previous years. Tweets are judged based on a four-point scale information need: *spam/junk*, *not interesting*, *somewhat interesting*, *very interesting*. The computation of score for each participated system is an average of the scores across all the topics. Additionally, the score of each topic is an average of the scores across all the evaluation days. Let T denotes the topic set where each $t \in T$ is a topic in the set and D is the date set of the evaluation period where each $d \in D$ is a day in the set. The final score for a system S is defined as the following:

$$Score(S) = \frac{1}{|T|}\sum_{t\in T} Score(t,S) = \frac{1}{|T|}\sum_{t\in T}\frac{1}{|D|}\sum_{d\in D} Score(t,d,S) \tag{2.10}$$

Evaluation of scenario A is based on an latency-discounted gain (LG) metric. It considers both the push notification latency time and the degree of relevance. The maximum acceptable delay is 100 minutes from the tweet creation time to the notification time that the tweet is considered to be pushed to the user. For a tweet t, the LG of t is defined as:

$$\begin{aligned} LG(t) &= L(c(t), n(t)) * g(t) \\ &= \frac{Max(0, 100 - delay)}{100} * g(t) \\ &= \frac{Max(0, 100 - (n(t) - c(t)))}{100} * g(t) \end{aligned}$$

where $c(t)$ denotes the created time of tweet t; $n(t)$ denotes the participating system pushing notification time of tweet t; $g(t)$ is the relevance gain of t, i.e. 0 for judged *spam/junk* or *not interesting* tweets, 0.5 for *somewhat interesting* tweets, 1 for *very interesting* tweets. For example, if a participating system notified a very interesting tweet within a minute of the created time of the tweet, the tweet adds 1 credit to the system. Another system delivered a somewhat interesting tweet after 50 minutes of the tweet being created, the system receives only 0.25 credit.

The primary metric is expected latency-discounted gain (ELG) and the secondary metric is normalized cumulative gain (nCG). The computation of ELG for a daily score of a topic is an average of LG of all the tweets delivered (denoted as TD), which is the following:

$$\frac{1}{|TD|} * \sum_{tweet \in TD} LG(tweet) \tag{2.11}$$

The nCG metric is similar to ELG, only differing in averaging the LG of all the tweets delivered per day by the maximum possible number of tweets delivered per day (which is 10 this year) instead of the real number of tweets.

Scenario B is evaluated as a normal retrieval task. The tweets retrieved per day are considered as a ranked list of tweets. Normalized discounted cumulative gain (nDCG) is computed for a daily score of each topic:

$$\begin{aligned} DCG@k &= \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)} \\ nDCG@k &= \frac{DCG@k}{Ideal\ DCG@k} \end{aligned}$$

where $rel_i \in \{0, 0.5, 1\}$ is the relevance value of tweets; $Ideal\ DCG@k$ is $DCG@k$ value computed under an ideal ordering of the retrieved list for the given topic.

Chapter 3

Data Collections

In this chapter, we describe the data collections we use for evaluating the algorithms in the remainder of this thesis. These data collections are all text collections. In total, four different collections are utilized in this thesis work: 1) the TREC 2015 microblog tweet collection, 2) a Reuters news collection, 3) a larger tweet collection gathered specifically for this thesis work, and 4) a Wikipedia article collection. In the first section of this chapter, details about each collection are provided. In the second section, we compare the lexical differences between Wikipedia collection and tweet collection, to better illustrate the differences between social media collections and collections of standard English text.

3.1 Collections

3.1.1 TREC 2015 Microblog Collection

The TREC 2015 Microblog Track required participating groups to collect tweets from the standard Twitter public streaming API from July 20, 2015 00:00:00 UTC to July 29, 2015 23:59:59 UTC. This API is available to all Twitter users. Any Twitter user can have a developer account through which he or she can gain access to the public streaming API.

The collection was not distributed before the start of track experiments. It was generated in real-time and gathered by each Twitter user account independently. The collection might appear to be a personalized collection since each Twitter user is using their own account and connecting to the public streaming API individually. However in the study of Paik and Lin [125], they demonstrated that multiple listeners to the Twitter public streaming API receive effectively the same tweets. They set up six independent listeners for three days in March 2015. The Jaccard overlap across these six different tweet collections is 0.999.

During the 10 days of TREC 2015 Microblog Track official evaluation period, there were approximately 40 million tweets published through the Twitter public streaming API. These tweets are in many languages. They include retweets and also tweets that have already been deleted by the original posting users. In this thesis, we use this collection as the main experimental collection for our real-time tracking and push notification algorithms.

3.1.2 Reuters News Collection

This news article collection was crawled and collected from Reuters[1]. Reuters is a large international news agency owned by the Thomson Reuters company. On its website, it releases news happening from all over the world every day.

We develop a web page crawler and HTML parser for Reuters news web pages. The news articles in our collection start from October 20, 2006 and end on February 28, 2015. For each day during the period, we first obtain the top news headlines and URLs. Then, we go to the web page of each news article and parse the HTML code of the web page. The title, date and body of each news article are recorded in our collection.

In total, we gathered 154,940 news articles. The overall vocabulary size of this collection is 264,811. This collection is used primarily as a background collection for our keyword extraction methods.

[1]http://www.reuters.com

3.1.3 Tweet Collection

We gathered a larger tweet collection through the Twitter Streaming API between November 2013 to March 2015. The Twitter Streaming API allows Twitter user to obtain a sample of approximately 1% of all public tweets. It also provides a language field for filtering tweets with specific language(s). However, the language fields are mainly indicated by Twitter users themselves, which may not be correct for each individual tweet. For example, a bilingual speaker of English and Mandarin may post a tweet with a mixture of English and Chinese characters. As a first step filter, we restrict tweets in our collection as English tweets on the basis of the language field.

Each tweet is delivered in JSON format by the Twitter API. The JSON structure provides information about tweet content, including the URLs contained in tweet text, and if the user is posting or re-posting the tweet. If the tweet is a retweet, information about the original tweet is also provided in the JSON structure.

3.1.4 Wikipedia Collection

The Wikipedia collection was downloaded from MediaWiki data dumps[2]. Wikipedia offers free downloads of all available content from their website. The collection we downloaded is a copy of all English-language Wikipedia pages available on March 04, 2015, i.e., it contains all the Wikipedia English article pages current at that time.

The data downloaded from Wikipedia is an XML dump which contains all the HTML code for each page. Along with the content text in UTF-8, each page includes additional elements, indicating page structure, Wikipedia internal links, references, images, tables, and other mark up. We developed a simple parser, based on regular expressions, to extract only the content text from each page.

Both our tweet collection and Wikipedia collection were collected for use as large background collections for our keyword extraction methods. In the next section, we compare the lexical usage and understanding for English terms between these two collections through the use

[2]`https://dumps.wikimedia.org/enwiki/20150304/`

of word embeddings. Along with providing insight into these individual collections, this work uncovers differences in language usage between social media and standard English.

3.2 Lexical Comparison Between Wikipedia and Twitter Corpora

Users of social media typically employ highly informal language, including slang, acronyms, typos, deliberate misspellings, and interjections [59]. This heavy use of nonstandard language, as well as the overall level of noise in social media, creates substantial problems when applying standard NLP tools and techniques [50]. For example, [72] apply machine translation methods to convert tweets to standard English in an attempt to ameliorate this problem. Similarly, [14] and [60] address this problem by generating corrections for irregularly spelled words in social media.

In this section, we continue this line of research, applying word embedding to the problem of translating between the informal English of social media, specifically Twitter, and the formal English of more carefully edited texts, such as those found in Wikipedia. Starting with a large collection of Tweets and a copy of Wikipedia, we construct word embeddings for both corpora. We then generate a transformation matrix, mapping one vector space into another. After applying a normalization based on term frequency, we use distances in the transformed space as an indicator of differences in word usage between the two corpora. This method identifies differences in usage due to jargon, contractions, abbreviations, hashtags, and the influence of popular culture, as well as other factors. As a method of validation, we examine the overlap in closely related words, showing that distance after transformation and normalization correlates with the degree of overlap.

3.2.1 Background

Mikolov et al. [115] proposed a novel neural network model to train continuous vector representation for words. The high-quality word vectors obtained from large data sets achieve high accuracy in both semantic and syntactic relationships [57, 92, 176, 177].

Some probabilistic similarity measures, based on Kullback-Leibler (KL) divergence (or relative entropy), give an inspection of relative divergence between two probability distributions of corpus [81, 144]. For a given token, KL divergence measures the distribution divergence of this word in different corpora according to its corresponding probability. Intuitively, the value for KL divergence increases as two distributions become more different. Verspoor et al. [153] found that KL divergence could be applied to analyze text in terms of two characteristics: the magnitude of the differences, and the semantic nature of the characteristic words.

Subašić and Berendt [142] applied a symmetrical variant of KL divergence, the Jensen-Shannon (JS) divergence [95], to compare various aspects of the corpora such as Language divergence, headline divergence, named-entity divergence and sentiment divergence. As for the applications derived from above methods, Tang et al. [148] studied the lexical semantics and sentiment tendency of high frequency terms in each corpus by comparing microblog texts with general articles. Baldwin et al. [14] analyzed non-standard language on social media in the aspects of lexical variants, acronyms, grammaticality and corpus similarity. Their results revealed that social media text is less grammatical than edited text.

3.2.2 Methods of Lexical Comparison

Mikolov et al. [114] construct vector spaces for various languages, including English and Spanish, finding that the relative positions of semantically related words are preserved across languages. We adapt this result to explore differences between corpora written in a single language, specifically to explore the contrast between the highly informal language used in English-language social media with the more formal language used in Wikipedia. We assume that there exists a *linear* transformation relationship between the vectors for the most frequent words from each corpora. Working with these frequent terms, we learn a linear projection matrix that maps source to target spaces. We hypothesize that usage of those words appearing far apart after this transformation differs substantially between the two corpora.

Let $a \in \mathbb{R}^{1\times d}$ and $b \in \mathbb{R}^{1\times d}$ be the corresponding source and target word vector representation with dimension d. We construct a source matrix $A = [a_1^T, a_2^T, ..., a_c^T]^T$ and a target matrix $B = [b_1^T, b_2^T, ..., b_c^T]^T$, composed of vector pairs $\{a_i, b_i\}_{i=1}^c$, where c is the size of the vocabulary common between the source and target corpora. We order these vectors according to frequency in the target corpus, so that a_i and b_i correspond to the ith most common word in the target corpus.

These vectors are used to learn a linear transformation matrix $M \in \mathbb{R}^{d \times d}$. Once this transformation matrix M is obtained, we can transform any a_i to $a'_i = a_i M$ in order to approximate b_i. The linear transformation can be depicted as:

$$AM = B \tag{3.1}$$

Following the solution provided by Mikolov et al. [114], M can be approximately computed by using stochastic gradient descent:

$$\min_M \sum_{i=1}^{n} \parallel a_i M - b_i \parallel^2 \tag{3.2}$$

where we limit the training process to the top n terms.

After the generation of M, we calculate $a'_i = a_i M$ for each word. For each a_i where $i > n$, we determine the distance between a'_i and b_i:

$$Sim(a'_i, b_i), n \leq i \leq c. \tag{3.3}$$

Let Z be the set of these words ordered by distance, so that z_j is the word with the jth greatest distance between the corresponding a' and b vectors. For the experiments reported in this section, we used cosine distance to calculate this Sim metric.

3.2.3 Experiments

In this subsection, we describe the results of applying our method to Twitter and Wikipedia.

Experimental Settings

As mentioned in the above section, the Wikipedia dataset for our experiments consists of all English Wikipedia articles downloaded from MediaWiki data dumps. The Twitter dataset was collected through the Twitter Streaming API from November 2013 to March 2015. We restricted the dataset to English-language tweets on the basis of the language field contained in each tweet. To obtain distributed word representation for both corpora, we trained word vectors separately by applying the *word2vec*[3] tool, a well-known implementation of word embedding.

Before applying the tool, we cleaned Wikipedia and Twitter corpora. The clean version of Wikipedia retains only normally visible article text on Wikipedia web pages. The Twitter clean version removes HTML code, URL, user mention(@), the # symbol of hashtags, and all the retweeted tweets. The sizes of document and vocabulary in both corpora are listed in Table 3.1.

Corpora	**# Documents**	**# Vocabulary**
Wikipedia	3,776,418	7,267,802
Twitter	263,572,856	13,622,411

Table 3.1: Corpora sizes

There are two major parameters that affect *word2vec* training quality: the dimensionality of word vectors, and the size of the surrounding words window. We choose 300 for our word vector dimensionality, which is typical for training large dataset with *word2vec*. We choose 10 words for the window, since tweet sentence length is 9.2 ± 6.4 [14].

Visualization

In Figure 1, we visualize the vectors for the some of the most common English words by applying principal component analysis (PCA) to the vector spaces. The words “and”, “is”, “was” and “by” have similar geometric arrangements in Wikipedia and in Twitter, since these common words are not key differentiators for these corpora. On the other hand, the pronouns “I” and “you”, are heavily used in Twitter but rarely used in Wikipedia. Despite this difference in term frequency, after transformation, the vectors for these terms appear close together.

[3]https://code.google.com/p/word2vec/

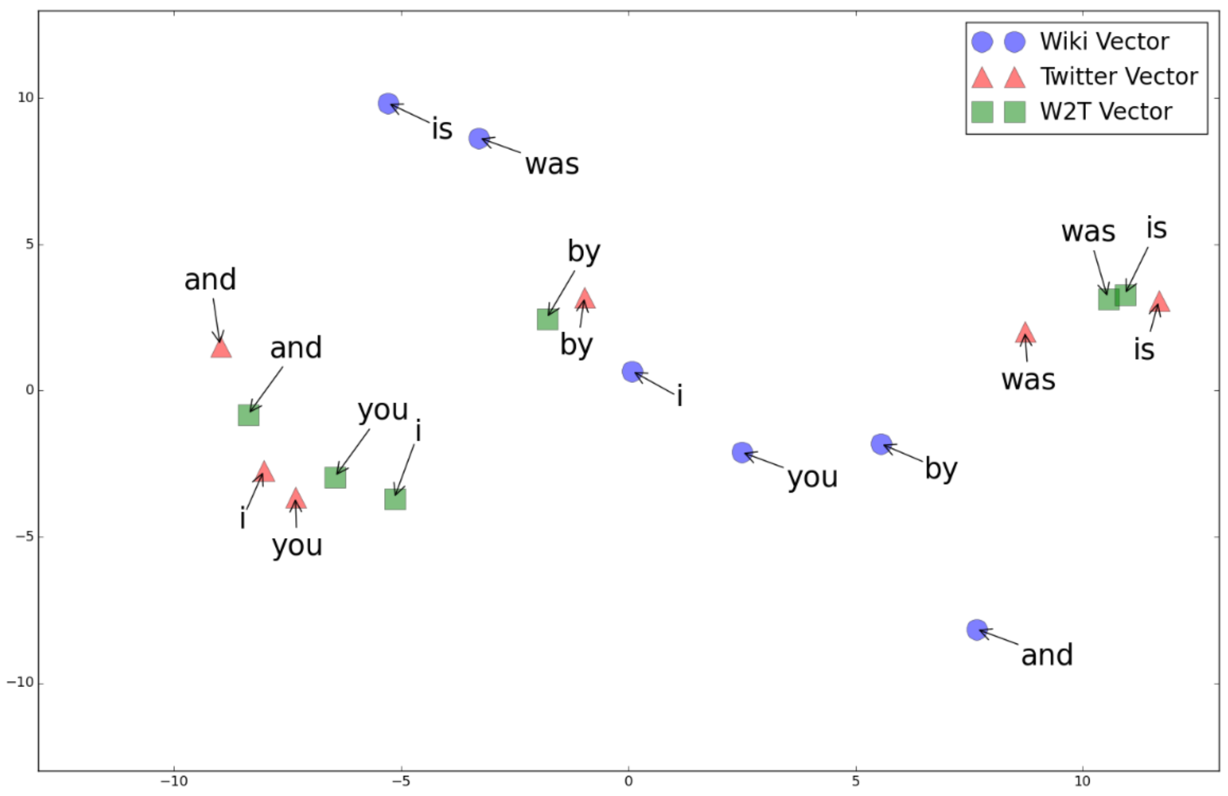


Figure 3.1: Word representations in Wikipedia, Twitter and transformed vectors after mapping from Wikipedia to Twitter.

Results

As our primary goal, we hope to demonstrate that our transformation method reflects meaningful lexical usage differences between Wikipedia and Twitter. To train our space transformation matrix, we used the top $n = 1,000$ more frequent words from the 505,121 words that appear in both corpora. The transformation can be either from Twitter to Wikipedia (*T2W*) or the opposite direction *W2T*. We observed that the two transformation matrices are not exactly the same, but they produce similar results. [116] suggest that a simple vector offset method based on cosine distance was remarkably effective to search both syntactic and semantic similar words. They also report that cosine similarity preformed well, given that the embedding vectors are all normalized to unit norm.

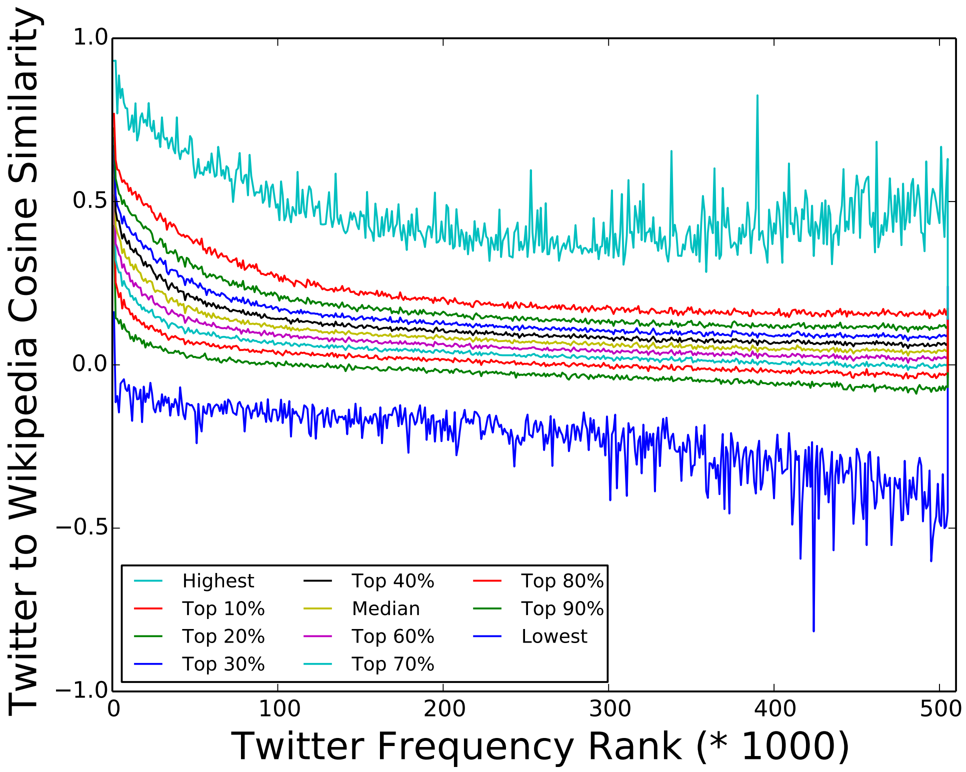


Figure 3.2: T2W transformated similarity curves.

Figure 3.2 illustrates how *T2W* word vectors are similar to their original word vectors. Similarly, Figure 3.3 shows how *W2T* word vectors are close to their original word vectors in Wikipedia collection. For the purpose of explaining Figure 3.2 and Figure 3.3, we define new notation as follows: Let $\mathcal{T}$ and $\mathcal{W}$ be the word sets of Twitter and Wikipedia respectively, and let $\mathcal{C} = \mathcal{T} \cap \mathcal{W}$. We denote the document frequency of a word t in the Twitter corpora as $df(t)$. Sorting the whole set $\mathcal{C}$ by $df(t)$ in an ascending order, we obtain a sequence $\bar{S} = \{c_0, \cdots, c_{m-1}\}$, where $c_i \in \mathcal{C}$; $m = 505,121$; and $df(c_i) \leq df(c_j), \forall i < j$. We partition the sequence $\bar{S}$ into 506 buckets, with a bucket size $b = 1000$. $B_i = \{c_{i*b}, \cdots, c_{(i+1)*b-1}\}$ represents the i-th bucket. We number the curves in Figure 3.2 and Figure 3.3 from the top to the bottom. The points on the i-th curve demonstrates the cosine similarity of the $(i-1) * 100$-th word in each bucket. From these figures, it is apparent that words with higher frequencies have higher average cosine similarity than those words with lower frequencies. Since our goal is to find words with lower than average similar, we apply the median curve of Figure 3.2 to adjust word distances.

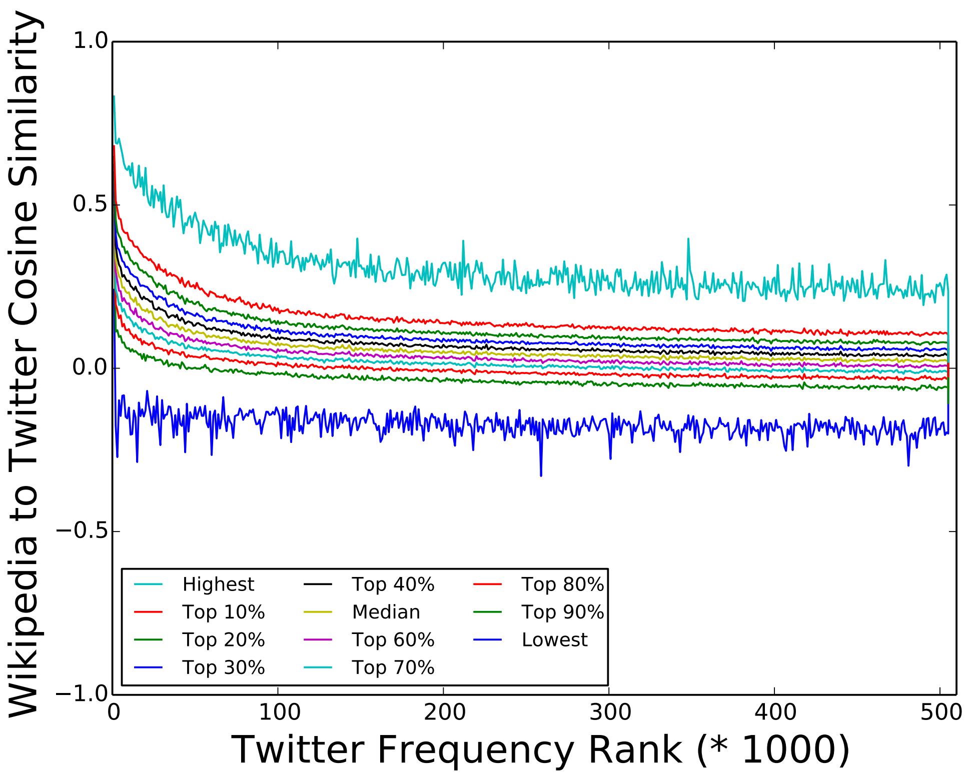


Figure 3.3: W2T transformated similarity curves.

Defining **adjusted distance** as $D_{adjusted}(t)$ of a given word t, we calculate the cosine distance between t and the median point c_{median} from its corresponding bucket B_i.

$$D_{adjusted}(t) = Sim(c_{median}) - Sim(t) \tag{3.4}$$

where the index of median point should be $i * b + b/2$. A negative adjusted distance value means the word is more similar than at least half of words in its bucket. On the other hand, the words that are less similar than at least half of words in their buckets have positive adjusted distance values. The larger an adjusted distance, the less similar the word between the corpora.

Examples

In the following table, it provides some examples of common words with large adjusted distance, suggesting that their usage in the two corpora are quite different. For each of these words, the example shows the closest terms to that word in the two corpora. In Twitter, “bc” is frequently an abbreviation for “because”, while in Wikipedia “bc” is more commonly used as part of dates, e.g. 900 BC. Similarly, in Twitter “ill” is often a misspelling of the contraction “I’ll”, rather than a synonym for sickness, as in Wikipedia. The other examples have explanations related to popular culture, jargon, slang, and other factors.

Word	Twitter Most Similar	Wikipedia Most Similar
bc	because bcus bcuz cuz cos	bce macedon hellenistic euthydemus ptolemaic
ill	ll imma ima will youll	unwell sick frail fated bedridden
cameron	cam nash followmecam camerons callmecam	gillies duncan mckay mitchell bryce
mentions	unfollow reply respond strangerswelcomed offend	mentions mentioned mentioning reference attested
miss	misss love missss missssss imiss	pageant pageants titlehoder titlehoders pageantopolis
yup	yep yupp yeah yea yepp	chevak yupik gwaii tlingit nunivak
taurus	capricorn sagittarius pisces gemini scorpio	poniatovii scorpio subcompact sagittarius chevette

3.2.4 Validation

To validate our method of comparing lexical distinctions in the two corpora, we employ a ranking similarity measurement. Within a single corpora, the most similar words to a word t can be generated by ranking cosine distance to t. We then determine the overlap between the most similar words to t from Twitter and Wikipedia. The more the two lists overlap, the greater the similarity between the words in the two corpora. Our hypothesis is that larger rank similarity correlates with smaller adjusted distance.

Rank biased overlap (RBO) a new rank similarity measure designed for comparisons between top-weighted, incomplete and indefinite rankings. Given two ranked lists, A and B, let $A_{1:k}$ and $B_{1:k}$ denote the top k items in A and B. RBO defines the overlap between A and B at depth k as the size of the intersection between these lists at depth k and defines the agreement between A and B at depth k as the overlap divided by the depth. [157] defines RBO as a weighted average of agreement across depths, where the weights decay geometrically with depth, reflecting the requirement for top weighting:

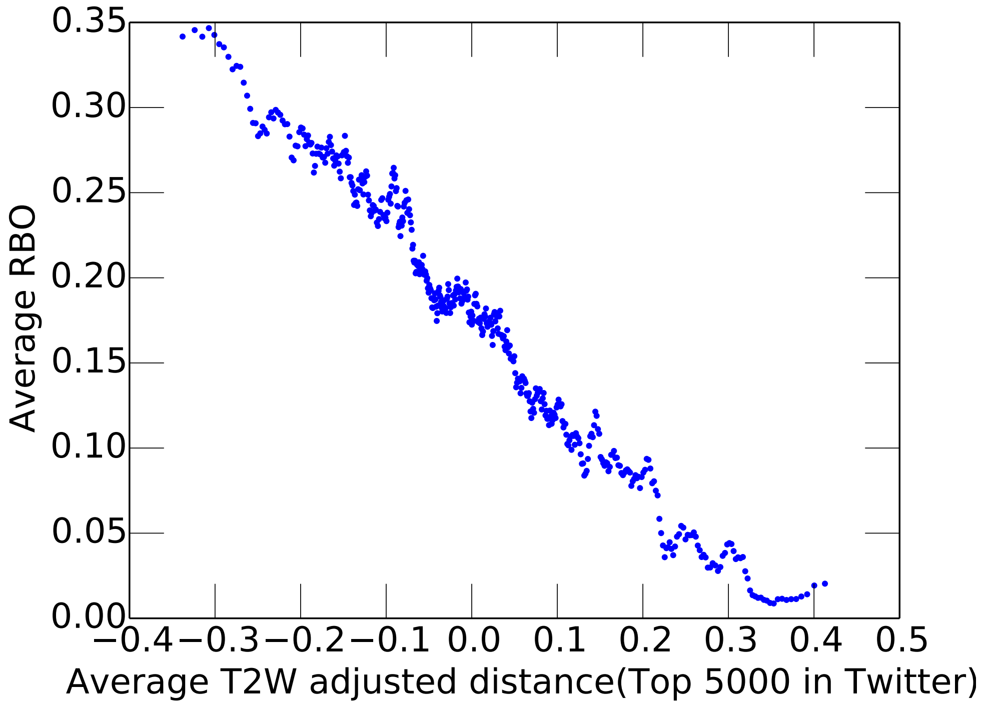
$$RBO = (1 - \varphi) \sum_{k=1}^{\infty} \varphi^{k-1} \frac{|A_{1:k} \cap B_{1:k}|}{k} \tag{3.5}$$

Here, φ is a persistence parameter. As suggested in that paper, we set $\varphi = 0.9$, a typical choice. In practice, RBO is computed down to some fixed depth K. We select $K = 50$ for our experiments. For a word t, we compute RBO value between its top 50 similar words in Wikipedia and top 50 similar words in Twitter.

In Figure 3.4, we validate consistency between results of our space transformation method and RBO. For the top 5,000 terms in the Twitter corpora, we sort them by their adjusted distance value. Due to properties of RBO, there are many zero RBO values. To illustrate the density of these zero overlaps, we smooth our plot by sliding a 100-word window with a step of 10 words. As shown sharply in the figure, RBO and adjusted distance is negatively correlated.

3.2.5 Conclusion

This section analyzed the lexical usage difference between Twitter microblog corpora and Wikipedia corpora. A word-level comparison method based on word embedding is employed to find the characterisic words that particularly discriminating corpora. In future work, we plan to introduce this method to normalize the nonstandard language used in Twitter.



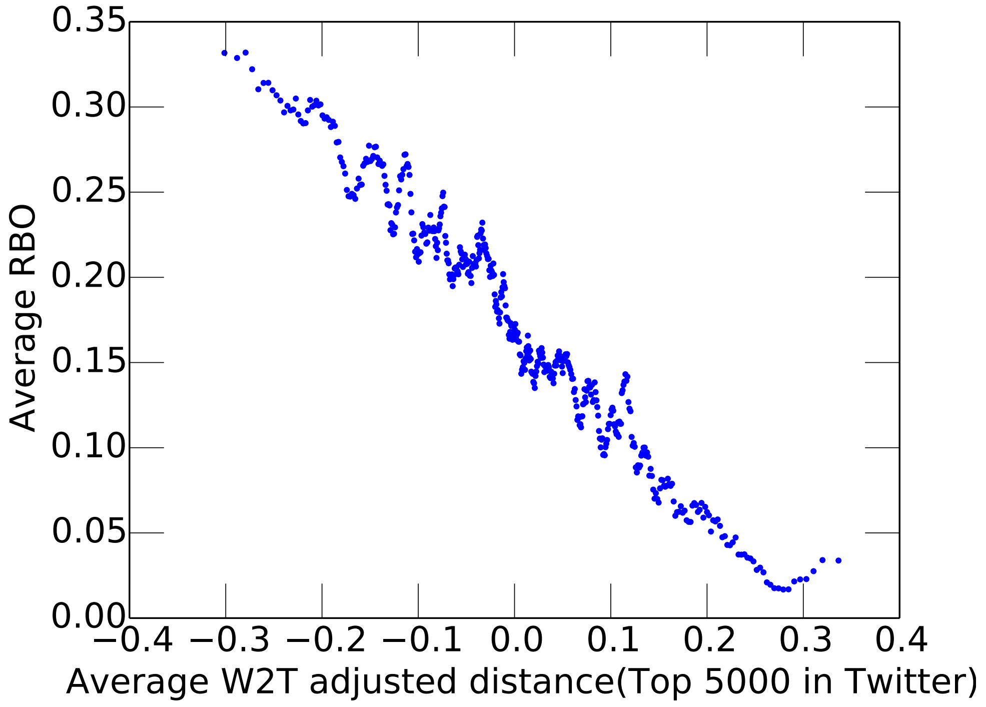


Figure 3.4: T2W and W2T negative correlation between adjusted distance and RBO.

Chapter 4

Succinct Query Generation for Tracking News

Given a news article of interest, we wish to create a succinct query reflecting its content, which may be used to follow the news story over a period of days, or even weeks. Alternatively, after reading a news article, or a similar source document, we may wish to find related material in social media, or in a similar target resource. In part, the need for succinct queries is occasioned by limitations of commercial social media search engines, which can perform poorly with longer queries. The case is different from traditional topic tracking task, where an explicit query or a set of terms are provided for tracking and there is no change of the tracking platform.

Prior research has approached this *Query by Document* problem [165] in a variety of ways. As one approach, we might extract key words and phrases from the source document, forming queries from these extracted terms. For example, given the news article "More than 100 Congolese refugees killed in boat accident, Uganda says[1]" we might extract the query "Lake Albert boat accident", which may produce higher precision results than querying with more general terms, such as "Uganda" or "Congo".

To query by document across blogs and other media, Yang et al. [165] employ a part-of-speech tagger to identify candidate terms in the source document. They rank these terms using

[1] http://www.cnn.com/2014/03/24/world/africa/uganda-boat-capsizes-death-toll

TF-IDF and mutual information, supplementing them with associated terms from Wikipedia. Similarly, Tsagkias et al. [151] explore query-by-document methods for linking news with social media. They employ multiple methods for extracting query terms from the source document, supplementing them with terms extracted from social media posts that explicitly reference the news article. They then separately execute the queries from each method, generating multiple ranked lists and merging them through late fusion.

Unfortunately, this approach can produce large and complex queries. For example, one method employed by Tsagkias et al. simply uses the full document as the query. Perhaps unsurprisingly, using full documents as queries produces the best results of any single method they explore, and it also provides a challenging baseline for evaluating late fusion.

The execution and fusion of multiple large and complex queries may not be feasible to support casual queries to find related material. Even if this process can be justified, re-querying to follow updates to an evolving news story requires the process to be repeated. Moreover, from a practical standpoint, commercial social media search engines may provide poor external support for long queries, requiring this approach to be built into the engine itself.

As an alternative, we present an approach for generating *succinct queries* from a candidate set of extracted terms. These succinct queries (comprising perhaps four or five terms in total) provide a lightweight way to follow the story over hours, days, or even weeks. Starting with the candidate terms, we execute a series of *probe queries* over a sample of contemporaneous social media. By analyzing the results of the probes, and comparing them to the language of the source document, we rank the terms according to their ability to retrieve related material.

The problem of finding additional documents related to a given source document has been a longstanding research topic within the information retrieval community [42, 112, 136, 158]. Generally these methods assume that the search engine will directly support a "find similar" feature, although Dandan et al. [40] consider the generation of queries for identifying near-duplicate documents by querying against a search engine. In this chapter, rather than finding similar documents within a given collection, we create succinct queries to efficiently search across collections and across time.

Other research considers the problem of trimming and re-weighting verbose queries, although without probe queries. Bendersky and Croft [18] train a classifier to recognize key words and

```
Input:    Primary source document $A$
          Secondary source collection $\beta = \{B_1, B_2, ...\}$
Output:   Ranked list of terms $Q = \{q_1, ..., q_n\}$

1) Extract initial candidate term set from A.
        $T = \{t_1, ..., t_m\}$

2) Select subsets of $T$ as probe queries.
        $P_1, P_2, ...$, where each $P_i \subseteq T$

3) Apply each probe query to rank documents in $\beta$;
        producing a probe ranking $\beta_i$ corresponding to each $P_i$.

4) Compute similarities between the source document $A$ and each probe ranking $\beta_i$.
        $s_i = \textit{similarity}(A, \beta_i)$

5) Estimate the term ranking that best explains the similarity values.
        $Q = \{q_1, ..., q_n\}$
```

Figure 4.1: Generating succinct queries. Any prefix of Q may be used as a succinct query.

phrases in queries consisting of a few sentences, as is typically seen in the topic descriptions from older TREC topics. Bendersky et al. [19] extended this approach to incorporate pseudo-relevance feedback and information derived from external sources. Lease et al. [89] re-weight these terms using simple term features in a learning to rank framework. Xue et al. [159] use conditional random fields to model query subset distributions and select the terms to keep.

Closer to our work, Kumaran and Carvalho [84] analyze term subsets from TREC topic descriptions. They use various query quality features to rank subsets, with some features based on the execution of subsets as probe queries. While we start with full documents and introduce a post-probe analysis step, we intend to expand our work with their ideas, extending them to social media.

Balasubramanian et al. [13] build on the work of Kumaran and Carvalho to improve the performance of longer Web queries, i.e., those with five terms or more. They employ query quality features to predict which term, if any, could be removed to improve the performance of these longer queries. Datta and Varma [41] further extend this work by probing with randomly selected subsets. Other related work includes methods for term select in pseudo-relevance feedback [23]. Jiang and Allan introduce the notion of necessary and frequent terms [68]. Kumaran and Allan [83] examine interactive query reduction.

Figure 4.1 presents our succinct query generation algorithm. While we express the algorithm in general terms, within this chapter the primary source document is a news article taken from a mainstream news source, while the secondary source collection is sample of tweets from a three-day window starting at the date of the news article. This secondary source collection is used to execute probe queries. Tweets for the secondary source collection were gathered through the Twitter Streaming API, which produces a maximum yield of 1% of the total tweet stream. Twitter's search engine cannot be used to execute probe queries, since it restricts the speed at which it accepts queries from a particular user.

Output from the algorithm is a ranked list of query terms intended to find social media content related to the news story. The term ranking is intended to reflect the expected value of the terms for this purpose. For the experiments reported in this chapter, we form a succinct query from the top five terms of Q, which are then executed on the main Twitter search service.

Each step of this algorithm could be implemented in numerous ways. In this chapter, we explore different implementations of these steps. The algorithm could be further generalized by repeating steps 2-5 multiple times, with the results of each iteration applied to suggest new subsets for probing in the next iteration, and with each iteration improving the estimated ranking. New terms might be extracted from the probe rankings through pseudo-relevance feedback and added to the candidate set. Figure 4.2 is showing an overview of the algorithm as an implemented system.

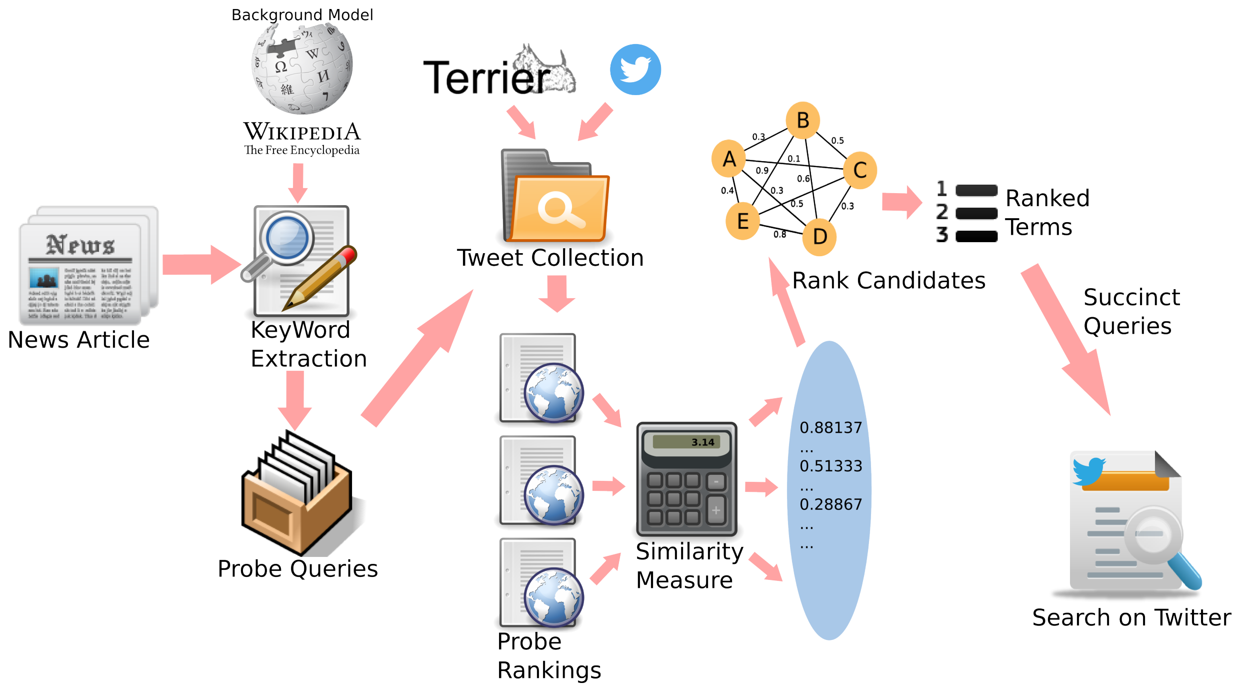


Figure 4.2: Overview of the succinct query generation system.

4.1 Extracting Candidate Terms

To extract an initial candidate term set (step 1) we apply multiple key phrase extraction methods to the news article. Each news article was pre-processed by stop-word removal and tokenization.

4.1.1 Pointwise Kullback-Leibler Divergence

The Kullback-Leibler (K-L) divergence is a measure in information theory. It measures the difference between two distributions: a posterior probability distribution or a observation distribution P and a prior probability distribution or a precisely calculated theoretical distribution Q. For discrete probability distributions P and Q, the K-L divergence of Q from P is defined as:

$$\sum_{x} P(x) * log(\frac{P(x)}{Q(x)}) \tag{4.1}$$

For each news article, we use pointwise K-L divergence [108, 150] to rank terms appearing in the article's full contents, which is:

$$p(t)\log(\frac{p(t)}{q(t)}), \tag{4.2}$$

where $p(t)$ is the relative frequency of term t in the article and $q(t)$ is the relative frequency of term t in the background model collection(cleaned Wikipedia collection described in Section 3.1.4). Relative frequency is the term's probability of occurrence according to the unigram language model. The value of pointwise K-L divergence for each term is used to measure how different the relative frequency of term t in the given article and the background model. The larger the value is, the bigger difference is. By ranking the scores of all terms in the article contents, a ranked list of article special terms is produced.

4.1.2 TextRank

The TextRank model [113] is a graph-based ranking model for determining the importance of text pieces (denoted as vertices in the graph), where global information (eg. knowledge drawn from an entire text) can be used for computing that importance, rather than only local information associated with individual vertices. TextRank models can be derived from different graph-based ranking models, such as Google's PageRank [124] and the Hyperlink-Induced Topic Search(HITS) algorithm [77]. Similar to the implementation of [113], in this chapter we rely on an algorithm derived from PageRank.

A formally graph-based ranking algorithm considers a directed, unweighted graph G with a set of vertives V and a set of edges E, where $G = (V, E)$ and E is a subset of set $V \times V$. The score of a vertex V_i is defined as:

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{4.3}$$

Here, $In(V_i)$ denotes the set of vertices that point to V_i and $Out(V_i)$ is the set of vertices that V_i points to. Let d be the damping factor which is the probability of randomly jumping from a given vertex to another one in the graph. Computation starts from arbitrarily assigned initial values to the vertices, and iterates until convergence under a given error rate. The error rate of convergence is usually approximated by the difference between scores of two iterations, which is $S^{K+1}(V_i) - S^k(V_i)$.

TextRank is built for natural language texts, and introduces a weight for indicating the ”strength” between two vertices. The score of vertex V_i with weighted edges in the graph is defined as:

$$WS(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j) \tag{4.4}$$

A typical application of TextRank model is to automatically extract keywords from documents. It was suggested in [113] that the following steps would help ranking nature language texts:

1. Identify text units as vertices, such as words, phrases, sentences or paragraphs. Add the vertices to the graph.
2. Assign initial values to all the vertices.
3. Identify relations that can be used as edges for linking the vertices in the graph. Edges can be weighted or unweighted, directed or undirected.
4. Iterate the ranking algorithm until convergence.
5. Sort vertices based on their final scores.

For the purpose of extracting news article keywords, we first considered single words as units. We used a sliding window size of 2 words to discover relationships between units. Weights were set to the number of co-occurrence within a widow. The damping factor d was set to 0.85, a value typically suggested in the research literature. The maximum number of iterations was set to 1000 and the error rate was set to 0.00001.

4.1.3 Evaluation of Candidate Term Extraction

We compared the performance of the pointwise K-L divergence method and the TextRank method for candidate term extraction purposes. The dataset we used for evaluating these key word extraction methods was the SemEval2010 training set [76]. This training set contains 144 scientific articles, with author-assigned and reader-assigned keyphrases. Traditionally, the performance of a keyword/keyphrase extraction system is determined by computing the proportion of top k candidates that match exactly the gold standard keywords or keyphrases. Thus, we evaluate the K-L divergence method and the TextRank method using this match evaluation metric. The following table (Table 4.1) shows an example of ranked lists from K-L divergence, TextRank, and the gold standard.

K-L divergence	TextRank	Gold Standard
svms	spam	support vector machine
spam	svms	content-based filtering
online	online	spam filtering
detection	data	blog
data	performance	splog
email	set	link analysis
splog	results	machine learning technique
performance	detection	link spam
filtering	email	content-based spam detection
blog	cost	bayesian method

Table 4.1: An example of key word ranking list of different methods.

The annotated key information are keyphrases in SemEval2010, while the output ranked list of our two methods are key words ranking lists. Thus, we first change the gold standard keyphrase set to a keyword set. We then calculate the average precision, recall and F-score(β=1), which are defined as follows:

$$Precision@k = \frac{\#gold\ standard\ key\ words\ in\ top\ k\ candidates}{k} \tag{4.5}$$

$$Recall@k = \frac{\#gold\ standard\ key\ words\ in\ top\ k\ candidates}{\#all\ glod\ standard\ key\ words} \tag{4.6}$$

$$F1@k = \frac{2 * Precision@k * Recall@k}{Precision@k + Recall@k} \tag{4.7}$$

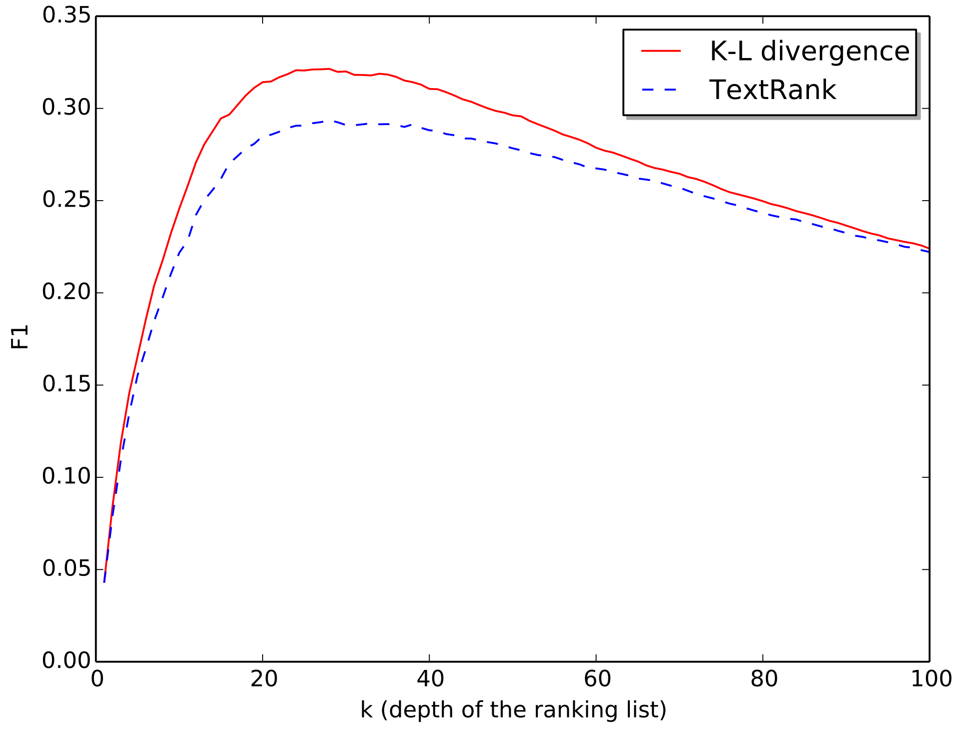


Figure 4.3: Average $F1$ scores of different depth of ranked lists.

Figure 4.3 shows $F1$ scores at different top k candidate key words averaged over articles. For all the different depths k, K-L divergence method performs equal to or better than TextRank method (p-value of a paired sign test is 0.02). The best $F1$ scores for both methods occurs at depth 27, where the $F1@27$ of K-L divergence is 32.14% and the $F1@27$ of TextRank is 29.36%.

We also computed the average running time for both methods to evaluate their efficiency. To give a sense of comparative performance, we compare the run times on a typical desktop machine. The average run time of K-L divergence for each article is 0.16 seconds (including the time spending on building the background model, which is only undertaken once), and the average run time of TextRank is 2.51 seconds. Thus, we chose K-L divergence as our key word extraction method for later experiments reported in this chapter.

Before doing the above comparison, we chose the top 20 terms based on preliminary experiments over a set of pilot news articles. We manually selected ten news articles from CBC news website [2], and employed a member of our research group to annotate the relevance in the generated key word ranking list for each news article.

The top-20 terms from pointwise K-L divergence ranking method for each news article formed a set L as our candidate term set for the following experiments. In later steps, we also use L as a simple language model for the news story. As suggested by the results of Tsagkias et al. [151], the non-stopword terms in the headline (term set denoted as H) provide a solid baseline query for linking news to social media.

In our evaluation experiments, we used terms from H as a baseline to compare with our succinct query generation algorithm. At the end of the first step of our algorithm, the initial candidate term set we used is term set T, where $T = L \cup H$.

4.1.4 Key Phrases Extraction

Alternatively, we could use key phrase extraction methods to extract candidate phrases from the news articles. Thus, we explored noun phrase chunking and named entity recognition to test the performance of key phrase extraction methods.

Noun Phrase Chunking Yang et al. [165] identifies candidate key phrases for documents using noun phrase patterns. Following their approach, the extraction of candidate phrases is preformed with the help of a part-of-speech (POS) tagger. We first split the original news article by sentences. A tokenizer and POS tagger are then applied to each sentence. We use the Natural

[2]http://www.cbc.ca/

Language Toolkit (NLTK) package [3] for these purposes. NLTK is a Python package that provides libraries for text processing, such as tokenization, stemming, classification, tagging and parsing.

Based on the POS tags, we extract all candidate noun phrases from each sentence. A set of noun phrase patterns (NPP) were defined by Yang et al. Each noun phrase is a sequence of terms whose POS tags match a NPP in the NPP set. Three types of terms are components of NPPs: nouns(N), adjectives(J) and conjunctions(C). The following table shows all NPPs we used, which originated from the work of Yang et al. which we partially modified by adding more combinations:

N	NCN	NN	NNCN	NNN
NNNCN	NNNN	NNNNN	JN	JNN
JNNCN	JNCNN	JNCN	JCJJN	JJJJN
JJJN	JJCJN	NCNNN	NCNN	JNNN
JNNNN	JJN	JJNCN	JJNN	JJNNN
JCJNN	JCJN			

After all candidate noun phrases were extracted, we scored the candidate phrases using an undirected weighted graph-based method. Each candidate phrase was treated as a node in the graph. Weights between two nodes were counted by the frequency of co-occurrence in the same sentence. The score of each node was defined as the sum of all weights of the node. We noticed that using this method, shorter noun phrases, and especially single noun terms, would be ranked higher. In addition, there were duplicates within the ranked lists. For example, a ranked list of noun phrases for the article “Google sells Motorola to Lenovo for $2.9B” [4] is shown in Table 4.2:

We observed from the result lists that if multiple terms were contained in a high ranking key phrase, each individual term was also ranked in high positions in key word ranking lists. Additionally, since our final goal is to discover succinct queries, we prefer shorter length of queries, which key word extraction methods tend to provide. Thus, we did not select the noun phrase chunking method as our candidate term extraction method.

[3] http://www.nltk.org/

[4] `http://www.cbc.ca/news/technology/google-sells-motorola-to-lenovo-for-2-9b-1.2516744`

Google sells Motorola to Lenovo for $2.9B	Noun Phrases
An expensive mistake by Google could turn into a golden opportunity for China's Lenovo Group as it expands beyond its success in the personal computer industry. Google is ridding itself of a financial headache by selling Motorola Mobility's smartphone business to Lenovo for $2.9 billion. The deal announced late Wednesday comes less than two years after Google bought Motorola Mobility for $12.4 billion in the biggest acquisition of Google's 15-year history. While Google Inc. is backpedaling, Lenovo Group Ltd. is gearing up for a major expansion. Already the world's largest PC maker, Lenovo is now determined to become a bigger player in smartphones as more people rely on them instead of laptop and desktop computers to go online. Lenovo already is among the smartphone leaders in its home country, but ...	motorola google lenovo inc smartphones company companies apple product apple inc patents mobility motorola mobility pcs smartphones technology companie lines product lines global product lines ...

Table 4.2: A noun phrase extraction example.

Named Entity Recognition Another candidate phrase extraction method we considered was named entity recognition. Since an event was defined as ”something that happens at a specific time and place along with all neccessary proconditions and unavoidable consequences by TDT tasks, Nallapati et al. [121] concluded that key words/phrases extracted from a news story can be categorized to that words answering the questions “who?”, “where?”, “what?” and “when?”.

We utilized the Stanford NER [53] which is a named entity recognizer[5]. It helps to label

[5]http://nlp.stanford.edu/software/CRF-NER.shtml

names of entities from sequences of words in a piece of text, such as persons, locations and company names. The recognized named entities are not ranked or weighted by any score function. For example, the named entities recognized by Stanford NER for the example article above ("Google sells Motorola to Lenovo for $2.9B") are as follows:

DATE	ORGANIZATION	MONEY	LOCATION	PERSON
Wednesday	Google	2.9 billion	China	Yang Yuanqing
this month	Motorola	2.3 billion	U.S.	Frank Gillett
2012	Lenovo	2.35 billion	Latin America	Gillett
Thursday	Lenovo Group	1.65 billion	Mountain View Calif.	
fourth-quarter	Motorola Mobility			
2007	Google Inc			
summer	Lenovo Group Ltd			
this year	BlackBerry Ltd			
	Associated Press			
	IBM Corp			
	Bebeto Matthews Associated Press Google			
	Arris Group Inc			
	Arris			
	Apple Inc			
	Forrester Research			

However, lacking of information about the importance of each named entity, we can not rank the named entities in each category. In addition, no explicit evidence indicates the significance of different categories. The average length of named entities is also too long for the goal of succinct queries. Therefore, we did not use this method for extracting candidate terms.

4.2 Selecting probe queries

Given that $|T| \geq 20$, we do not probe with all $2^{|T|}$ subsets. Instead, we first probe all pairs from T, which provided reasonable performance in our preliminary experiments. Probing all subsets of size three, four or five terms [84] is also feasible, especially given that probes queries are executed over a subset of tweets. Random selection of subsets is another possibility [41].

We tested all subsets of one, two, three, four and five terms. Since these probe queries were selected from the candidate term set, the terms in each probe query were not ordered. Different size of subsets produced different numbers of potential probe queries. Some combinations of the probe terms are meaningful and others are not. We then execute these probe queries on our collection and re-rank the candidate terms based on the scores computed for each probe query.

4.3 Executing probes

We executed each probe query P_i over the secondary source collection to produce a ranking β_i corresponding to each probe (step 3). The secondary source collection is the tweet collection we describe in 3.1.3.

We indexed our tweet collection using the Terrier IR Platform[6]. The Terrier IR Platform is an open source research search engine developed by the Terrier team from University of Glasgow. We rank tweets using the PL2 divergence from randomness formula, as implemented by Terrier, with the restriction that all probe terms are required to appear in all the retrieved tweets.

Since tweets are edited by individual Twitter users, the quality of tweets are extremely diverse. Tweets about a news article can be categorized as:

- Tweets posted by mainstream news organizations, such as BBC, CNN and CBC. Tweets of this type have a typical format, with a article's title followed by a url of the news content webpage. News organizations and news agents use Twitter as a platform to announce their news publications. Although tweets of this type are highly related to the news article, they

[6] http://terrier.org

only repeat the title of the news article, and contain no additional information otherwise.. For users tracking a news event in Twitter, they may not want to see the same news article title again and again.

- Retweets, i.e., re-posts of original tweets. Twitter users may be interested in the news title or content, so they simply re-post the original tweets from news agents or other users. Similarly to the above category, there is no extra information in this kind of tweet. As retrieval results for tracking, we should not repeatedly display tweets from this category.
- Comments on the news article content. Some users may comment on the events, persons, places and any part of the news article content. Sometimes they are providing additional information along with the retweets, including or not a url for the original news article.
- Junk tweets. To be retrieved by hot topic searching queries, some users include special terms or hashtags in their tweets. However, their tweets content may have no relationship with the terms or hashtags they are using.

Each β_i consists of up to the top-50 documents returned by Terrier, with the choice of 50 documents based on preliminary experimentation. Since retrieved tweets are required to contain all probe terms, some probes produce less than 50 documents. Since we seek material related to the news article, rather than re-tweets and re-postings of the article, we apply near-duplicate detection before computing similarities in the next step. We also assume that tweets containing all terms from the headline merely repeat the original story, and these tweets are removed from the β_i rankings.

4.4 Computing similarities

We analyze the probes by computing the similarity between the source document A and each β_i ranking (step 4):

$$s_i = \textit{similarity}(A, \beta_i) \tag{4.8}$$

The computation of this similarity is critical to the performance of the algorithm. In this section, we compute similarity using two different methods: one supervised-learning method and one unsupervised method.

4.4.1 SIM_1+SIM_2

For the unsupervised method, we compute similarity from two simple matching functions, both based on the weighted language model provided by L. As defined in 4.1, L is the simple language model extracted from the news story.

The first of these matching functions $\mathit{SIM}_1(L - P_i, \beta_i)$ is a precision-like method. It counts matches between terms in $L - P_i$ (i.e., L with the probe terms excluded) and the tweets in β_i taken as a group. It returns the proportion of terms in $L - P_i$ appearing in β_i. The bigger score from SIM_1 function is, the more other candidate terms except probe terms appeared in the probe ranking tweets (as shown in the following equation 4.9).

$$\mathit{SIM}_1(L - P_i, \beta_i) = \frac{|terms\ in\ L - P_i\ appearing\ in\ \beta_i|}{|L - P_i|} \tag{4.9}$$

The second matching function $\mathit{SIM}_2(L - P_i, \beta_i)$ compares each individual tweet in β_i to $L - P_i$. The number of matching terms is used to estimate a probability of relevance for each tweet. To compute these probabilities, we derived a mapping between the number of matching terms and the estimated probability of relevance from the results of our preliminary experiments.

In our preliminary experiments, we chose 11 news articles. All the tweets that contain at least one term from $L - P_i$ were gathered together. Then we randomly selected 100 tweets from each document. A member of our research group judged the relevance of each of these 1100 tweets. The results of this preliminary experiments are listing in the following table. The probability of number of matching terms in each document is shown in each cell of the table.

# Match	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10	Doc11
1	0.7272	0.5952	0.2069	0.2333	0	0.4565	0.8033	0.9167	0.4839	0	0.7356
2	0.9189	0.9444	1	0.75	0	0.8	0.7308	0.9762	0.9524	0.2857	1
3	1	1	1	0.6364	1	1	1	1	0.9643	0.4667	
4	1	1		1		1	1	1	1	1	
5	1	1		1		1	1	1	1	1	
6		1		1				1	1	0.9091	
7										1	
8										1	
9										1	
10										1	

Through these experiments, we then concluded the estimated probabilities as :

# matching terms	probability of relevance $Prob(t_j)$
tweet with 1 term in $L - P_i$	$Prob(t_j) = 0.5$
tweet with 2 terms in $L - P_i$	$Prob(t_j) = 0.75$
tweet with 3 terms in $L - P_i$	$Prob(t_j) = 0.9$
tweet with more than 3 terms in $L - P_i$	$Prob(t_j) = 1$

Table 4.3: The estimated probabilities of matching terms in each tweet.

These estimates are then summed and divided by $|\beta_i|$ to return an overall precision estimate for β_i. Thus, *SIM*$_2(L - P_i, \beta_i)$ is defined as :

$$\textit{SIM}_2(L - P_i, \beta_i) = \frac{\sum_{t_j \in \beta_i} Prob(t_j)}{|\beta_i|} \tag{4.10}$$

To compute similarity, we take a weighted linear combination of these two functions:

$$s_i = \alpha_1 \textit{SIM}_2(L - P_i, \beta_i) + \alpha_2 \textit{SIM}_2(L - P_i, \beta_i). \tag{4.11}$$

We exploited an unweighted linear combination, which equals to the case that $\alpha_1 = \alpha_2 = 0.5$ independent of later results. After collecting annotations as training data in the learning to rank subsection, we did a parameter sweep on α_1 and α_2.

4.4.2 Learning to Rank

To improve our unsupervised *SIM*$_1$+*SIM*$_2$ method, we developed a data set for supervised learning to rank method. The purpose of computing the similarities between the source document and the probe rankings is to rank the probe queries. Thus, we can consider the problem as a ranking problem and utilize learning to rank methods.

The data set we used for training and testing the learning to rank method are taken from the same news articles as we used for evaluating the *SIM*$_1$+*SIM*$_2$ method. From each article, we chose 10 probe queries. The chosen probe queries must have at least 5 tweets in their probe ranking and they must have *SIM*$_1$+*SIM*$_2$ scores greater than zero. To fairly sample from the probe queries, we carefully select the 10 probe queries using the following steps:

1. Split the selected probe query list into two equal halves. If there is a tie in splitting, we add the tie to the top half.
2. Randomly choose one probe query from the bottom half.
3. Drop the bottom half, and consider the top half as the selected probe query list.

We iterated the above steps until we have 10 selected probe queries. If there were not enough 10 selected probe queries, we randomly select the remainder from the whole set of remaining unselected probe queries. From each of these 10 selected probe queries, we took the top 5 tweets from each of the probe ranking lists. Therefore, 50 tweets were chosen from each news article.

A member of our research group was asked to judge relevance between each news article story and the 50 tweets. The judgement for each tweet is a binary judgement: 1 for relevant tweet and 0 for non-relevant tweet. To divide the 66 news articles into training and validation data sets, 11-fold cross validation was used. For each round of training and testing, we train on 60 news articles and test on 6.

The probe queries we selected were all pairs of candidate terms, which means there were exactly two terms in each probe query: term1 and term2. Considering the relatively small training dataset (60 articles * 10 probe queries), we extracted a small set of features to achieve the probe query ranking task. The extracted features were based on three different categories: news article (N), probe query/probe ranking tweets(P denotes for this type, Q for probe query, pt for probe ranking tweet list, $pt20$ for top 20 tweets in the probe ranking tweet list), the relationship between news articles and probe query/probe ranking tweets (N-P). A news article was represented by its language model, which was mentioned before as: $T = L \cup H$, where L is the body content language model of the news article and H is the non-stop words of the headline ranked by IDF value.

We chose two learning methods: RankSVM [70] and Linear Regression. RankSVM is an variant of the support vector machine algorithm. It employs pair-wise ranking method to solve

Category	Feature Description	Meta Data Used	Features
N	Probe query quality: Term rank in news article's language model	List of terms in H are ranked by IDF value generated from a background collection	term1's rank in L term1's rank in H term2's rank in L term2's rank in H
P	Probe ranking tweets quality: Statistics of tweet length in top 20 tweets of probe ranking		$\sum_{t\in pt20} \|t\|$ $avg_{t\in pt20}\|t\|$ $min_{t\in pt20}\|t\|$ $max_{t\in pt20}\|t\|$ $var_{t\in pt20}\|t\|$
P	Probe ranking tweets quality: Statistics of distinct stop word numbers in top 20 tweets of probe ranking	English stop word list	$\sum_{t\in pt20} \|stop\ words\ in\ t\|$ $avg_{t\in pt20}\|stop\ words\ in\ t\|$ $min_{t\in pt20}\|stop\ words\ in\ t\|$ $max_{t\in pt20}\|stop\ words\ in\ t\|$ $var_{t\in pt20}\|stop\ words\ in\ t\|$ $\sum_{t\in pt20} \frac{\|stop\ words\ in\ t\|}{\|t\|}$ $avg_{t\in pt20}\frac{\|stop\ words\ in\ t\|}{\|t\|}$ $min_{t\in pt20}\frac{\|stop\ words\ in\ t\|}{\|t\|}$ $max_{t\in pt20}\frac{\|stop\ words\ in\ t\|}{\|t\|}$ $var_{t\in pt20}\frac{\|stop\ words\ in\ t\|}{\|t\|}$

Table 4.4: Feature set description for learning to rank methods.

Category	Feature Description	Meta Data Used	Features
P	Probe ranking tweets quality: Statistics of term overlapping ratio between two tweets in top 20 tweets of probe ranking		$\sum_{t_i,t_j \in pt20} \frac{t_i \cap t_j}{t_i \cup t_j}$ $avg_{t_i,t_j \in pt20} \frac{t_i \cap t_j}{t_i \cup t_j}$ $min_{t_i,t_j \in pt20} \frac{t_i \cap t_j}{t_i \cup t_j}$ $max_{t_i,t_j \in pt20} \frac{t_i \cap t_j}{t_i \cup t_j}$ $var_{t_i,t_j \in pt20} \frac{t_i \cap t_j}{t_i \cup t_j}$
P	Number of returned tweets (cut at 50)		$\|pt\|$
N-P	Ratio of tweets containing key words from H		$\frac{\#t \in pt20 \ and \ \|H \cap t\|=1}{\|pt20\|}$ $\frac{\#t \in pt20 \ and \ \|H \cap t\|=2}{\|pt20\|}$ $\frac{\#t \in pt20 \ and \ \|H \cap t\|>=3}{\|pt20\|}$
N-P	Ratio of covered key words from H		$\frac{\|H \cap \bigcup_{t \in pt20} t\|}{\|H\|}$
N-P	Ratio of tweets containing key words from L		$\frac{\#t \in pt20 \ and \ \|L \cap t\|=1}{\|pt20\|}$ $\frac{\#t \in pt20 \ and \ \|L \cap t\|=2}{\|pt20\|}$ $\frac{\#t \in pt20 \ and \ \|L \cap t\|>=3}{\|pt20\|}$
N-P	Ratio of covered key words from L		$\frac{\|L \cap \bigcup_{t \in pt20} t\|}{\|L\|}$

Table 4.5: Feature set description for learning to rank methods(Continue).

ranking problems. If we assume the relationship between our features and the ranking scores is linear, we can format the learning problem as a linear regression problem. We utilized TreeRankSVM package [7] to train our RankSVM model and later used the trained model to predict ranking for each news article. The package for linear regression we used is from scikit-learn [8], which is a machine learning package in Python. We used the Ordinary Least Squares estimator.

For training and testing, we first employed 11-fold cross validation on only selected probe queries. We then trained on the entire selected probe query dataset and tested the trained model on all the probe queries from all of our 66 news articles. We also did experiments on two different label sets for both RankSVM and linear regression method: nDCG@5 and precision@5. When trained by linear regression, the predicted values were estimated nDCG@5 values or precision@5 values. The predicted values were then used for ranking the probe queries.

4.4.3 Evaluation of similarity computation

Kendall's τ rank distance Kendall's τ rank distance is a metric for measuring the similarity of the ranking of items when ranked by different quantities. With respect to a list with n items, we take each pair of items in the list that is ranked into consideration. If two quantities rank them in the same order, the pair is accepted as a concordant pair. In contrast, if the pair is ranked as different orders by two quantities, the pair is discordant. Here, n_c denotes the number of concordant pairs in two different rankings. Similarly, the number of discordant pairs in the two rankings is denoted by n_d. The total number of pairs of the n items is $n_0 = n(n-1)/2$. The Kendall's τ rank distance between two ranking results is defined as the equation in 4.12.

$$\tau = \frac{n_c}{n_0} - \frac{n_d}{n_0} = \frac{n_c - n_d}{n_0} \tag{4.12}$$

However, a big challenge with the original Kendall's τ measurement is that it does not adjust ties when the pair of items are ranked in the same position in one or both quantities.

Based on human judgements for each individual tweet from our selected probe ranking lists, we can rank the selected probe queries by the nDCG@5 value or precision@5 value of their

[7] http://staff.cs.utu.fi/~aatapa/software/RankSVM
[8] http://scikit-learn.org/stable/

probe ranking lists. The limited number of values of nDCG@5 and precision@5 bring ties in these selected probe query ranks. Thus, a extension of Kendall's τ, Kendall's τ_B appears to be better for our coefficient measurement case.

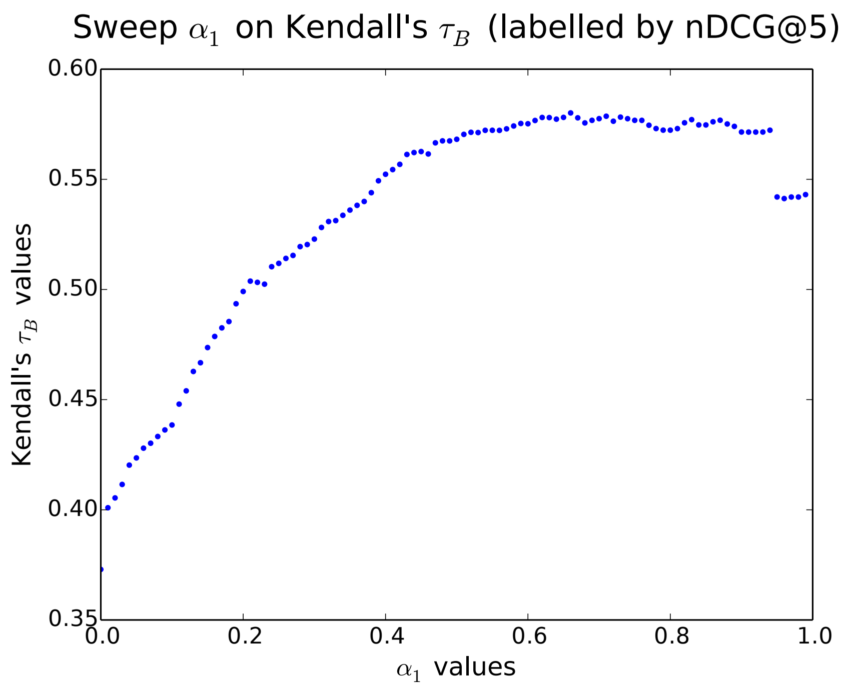
Kendall's τ_B The idea of Kendall's τ_B is to consider the number of tied values in each quantity. When computing the total number of item pairs, the tied value pairs are excluded from the total number of pairs. There might be different number of tie groups in each quantity. The numbers of tied values vary in different groups of ties. We can re-compute the total number of pairs from either the first quantity or the second quantity. To generate a normalized average of these two numbers, a geometric mean was computed. In the following equation, Kendall's τ_B is defined as:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}} \tag{4.13}$$

Here denote several quantities as follows: $n_0 = n(n-1)/2$ is the same as the denominator in τ_A; $n_1 = \sum_i t_i(t_i - 1)/2$ is the number of tie pairs in the first quantity, where t_i is the number of tied values in the i^{th} group of the first quantity; similarly, $n_2 = \sum_j u_j(u_j - 1)/2$ denotes the number of tie pairs in the second quantity, where u_j is the number of tied values in the j^{th} group of the second quantity.

Difference between ranking by nDCG@5 and precision@5 We first computed the different between rankings by nDCG@5 and precision@5 from human judgements. The Kendall's τ_B rank distance value is 0.9712 between the two rankings. They are highly coefficient when ranking the probe queries.

Parameter sweep of *SIM*$_1$ + *SIM*$_2$ To test our blind choice of weights for *SIM*$_1$ + *SIM*$_2$ method, we swept values of α_1 and α_2 on averaged Kendall's τ_B scores between *SIM*$_1$ + *SIM*$_2$ rankings and nDCG@5 or precision@5 rankings of the selected probe queries. We show the results of parameter sweep on α_1 of the two labels in Figure 4.4.



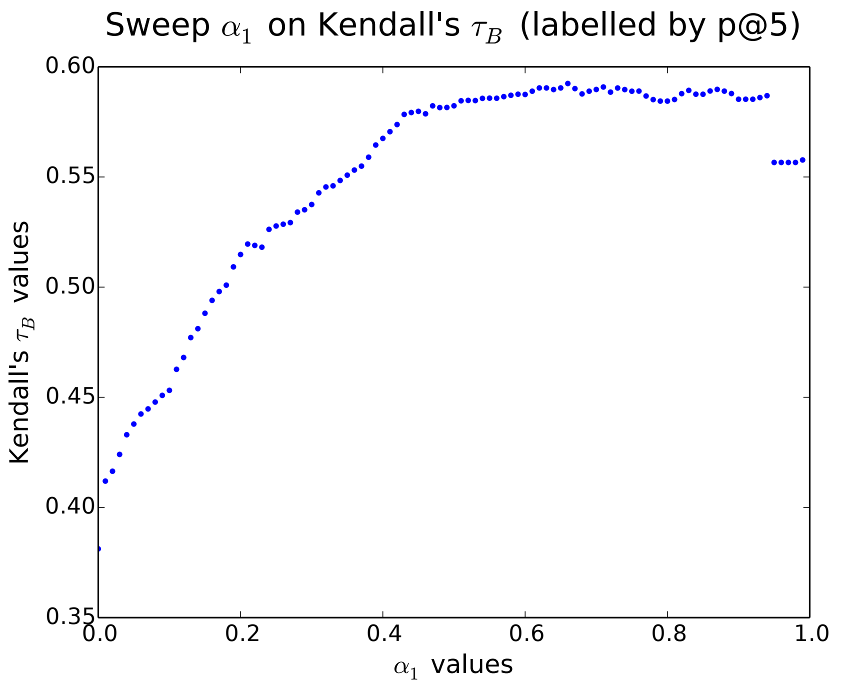


Figure 4.4: Parameter sweep on α_1 of SIM_1 + SIM_2.

SIM$_1$ + *SIM*$_2$ ranking result has higher rank coefficient with ranking labelled by precision@5 on Kendall's τ_B values. The best combination of weights for both label methods is $\alpha_1 = 0.66$ and $\alpha_2 = 0.34$. But the p value of a paired sign test is 0.7633, which means that the improvement is not significant. Therefore, in the following experiments, we retained our blind choice of $\alpha_1 = \alpha_2 = 0.5$.

Cross-Validation on selected probe queries In order to estimate the performance of different similarity measurements, we executed cross-validation on our selected probe queries dateset. In Table 4.6, we display the results of comparing different methods on ranking the selected probe queries based on different labels.

The *SIM*$_1$ + *SIM*$_2$ method is not a learning method, which we simply computed the Kendall's τ_B value between ranking list from *SIM*$_1$ + *SIM*$_2$ with ranking lists based on either nDCG@5 or precision@5 of human judgement. The distance between *SIM*$_1$ + *SIM*$_2$ method ranking list and nDCG@5 based ranking list from human judgements is 0.5682, compared with 0.5823 computed from the distance to precision@5 based ranking.

RankSVM method and Linear Regression(LR) method were trained by the corresponding two label sets. Both two training method improved the corresponding Kendall's τ_B values on both label sets. Significant t-test scores were calculated based on *SIM*$_1$ + *SIM*$_2$ method results. However, none of these improvements was significant.

Method	τ_B labelled by nDCG@5	τ_B labelled by precision@5
SIM$_1$ + *SIM*$_2$	0.5682	0.5823
RankSVM	0.5739(p=0.8895)	0.5933(p=0.7814)
LR	0.6060(p=0.3472)	0.6116(p=0.4646)

Table 4.6: Kendall's τ_B values between methods and human judgements.

Prediction on all probe queries We also tried to train our methods on all the selected probe queries, and tested the trained models on the entire probe query set. Since we did not have human judgements for individual tweets of all the probe queries, we measured the differences between *SIM*$_1$ + *SIM*$_2$ method and the trained methods. The Kendall's τ_B values are showing in table 4.7.

Method	Kendall's τ_B
RankSVM + nDCG@5	0.5744
RankSVM + precision@5	0.5909
LR + nDCG@5	0.6118
LR + precision@5	0.6137

Table 4.7: Comparison of the predictions on all the probe queries with *SIM*$_1$+*SIM*$_2$ method.

Although the RankSVM learning to rank method and Linear Regression learning method improved the rank coefficients between predicted rankings and human judgement rankings, the improvements were not significant, but had a dramatically increased calculation workload. As a results, we did not choose the learning to rank method or linear regression method in later evaluation experiments.

4.5 Ranking candidate terms

The similarity values $(s_1, s_2, ...)$ essentially represent a system of equations, each parametrized by a pair of probe terms. If we imagine a latent variable associated with each term, our goal is to estimate values for these variables, ranking T according to these values to produce Q. Since we need actual value for each probe query, the following approaches are based on *SIM*$_1$ + *SIM*$_2$ methods. For the learning methods, some methods can help us predict similarity values of the probe queries, which we can use them for this step. We explored several approaches, which often produced similar rankings.

4.5.1 PageRank-like

Our pagerank-like algorithm allows terms that produce high similarity value across many pairs to be properly recognized. We basically reused the TextRank equations. We create a Markov chain with each term in T represented by a state. Transition probabilities are derived from the similarity values associated with each term pair. Let $S(t_x, t_y)$ be the similarity value associated with a term pair $\{t_x, t_y\}$. We set the transition probability to

$$trans(t_x, t_y) = \frac{\delta}{|T|} + \frac{(1-\delta)S(t_x, t_y)}{\sum_{t \in T} S(t_x, t)}. \tag{4.14}$$

Following the example of pagerank, δ is a jump or transportation probability, which we set to 0.01. We then apply the power method to determine the stationary distribution, which ranks the terms in T.

4.5.2 Linear System

Linear system collects multiple linear equations involving the same set of variables. We treated each term in T as a variable, and the similarity values $S(t_x, t_y)$ as outputs of the linear equations. To rank the terms in T, we need to solve the linear system and find a solution of assigning a number for each variable. The linear system looks like:

$$\begin{cases} t_1 + t_2 & = S(t_1, t_2) \\ t_2 + t_3 & = S(t_2, t_3) \\ \dots \\ t_m + t_n & = S(t_m, t_n) \end{cases} \tag{4.15}$$

It is the simplest kind of linear system. For example of pair probe queries, each equation involves exactly two variables. The coefficients or weights of the system are all one. Since the variable set size are usually more than 20, there may not be a perfect solution for the linear system. We apply least-squares to find a solution.

4.5.3 Average Scores

The average scores method is a simplified version of the linear system method and pagerank-like method. We assume that within a probe query, each term contributes same value to the similarity score $S(t_x, t_y)$. It means that in the linear equation of the above system $t_1 + t_2 = S(t_1, t_2)$, we have $t_1 = t_2 = \frac{S(t_1,t_2)}{2}$. For each linear equation, we have similar solutions. We then sum up all the assigned values for each variable as the following:

$$t_i = \sum_{t_j \in T} \frac{S(t_i, t_j)}{2} \tag{4.16}$$

Similarly, using the values of each term in T, we can sort these terms.

4.5.4 Comparison of Re-Ranking Methods

To measure the differences among methods, we utilized a rank correlation metric: Rank Biased Overlap (RBO)[157]. RBO has already been introduced in Section 3.2.

$$RBO = (1 - \varphi) \sum_{k=1}^{\infty} \varphi^{k-1} \frac{|A_{1:k} \cap B_{1:k}|}{k} \tag{4.17}$$

Here, φ is a persistence parameter, where we set $\varphi = 0.9$, a typical choice. Since we chose top $K = 20$ key terms for news articles, we compute the RBO value between top 20 terms from each pair of the three methods.

The following results (Table 4.8) are shown for each pair of the methods: PageRank-like vs. linear system, PageRank-like vs. average scores, linear system vs. average scores. In each pair, we computed the average, maximum, minimal and median value for RBO across 66 news articles (dataset as described in subsection 4.6.1.

	Average	Minimal	Maximal	Median
PageRank-like vs. linear system	0.9867	0.8812	0.9999	0.9867
PageRank-like vs. average scores	0.9915	0.8812	0.9999	0.9915
Linear system vs. average scores	0.9939	0.9417	0.9999	0.9999

Table 4.8: Rank Biased Overla between each pair of the re-ranking methods.

Based on comparison results shown in each table, the ranking lists generated from the three re-ranking methods were highly correlated. Thus, we chose one of them (PageRank-like) method as the algorithm for our re-ranking step.

4.6 Evaluation

4.6.1 Data

We evaluated our approach using a collection of news articles taken from March and April 2014, with the resulting succinct queries used to re-query social media one week later. As mentioned above, we worked with a set of pilot news articles to conduct preliminary experiments during the development of our succinct query generation algorithm. These articles were not re-used for the experiments reported in this section.

We developed a fresh test set based on news articles linked from Wikipedia's news pages for March-April 2014. We use Wikipedia as a method for selecting articles to provide breadth and to prevent our personal news preferences from unduly influencing the selection. For simplicity, we restrict the selection to articles from six high-quality mainstream news sources: BBC, CNN, Reuters, the Washington Post, the Guardian, and CBC. Together, these sources provide coverage from a variety of perspectives across much of the English-speaking world. For some major events (e.g., MH370 and Crimea) we removed all but one related article to avoid having these events dominate the test set. A few other articles were removed for technical issues (e.g., parsing problems). This process produced a test set of 66 articles.

4.6.2 Methods and the Baseline

Based on our pilot study and evaluation experiments in each step, the final succinct query generation methods we chose for each step were: 1) pointwise K-L divergence for extracting initial candidate term set; 2) select pairs of terms as probe queries; 3) top-50 tweets returned from our tweet collection as probe ranking for each probe query; 4) $SIM_1 + SIM_2$ as our similarity comparison method; 5) PageRank-like method as our re-ranking method for candidate terms.

We applied our succinct query generation algorithm to each of the 66 articles, taking the top-5 terms from Q as our succinct query. As a baseline for comparison, we ranked the terms from article's headline (H) according to their IDF values in β, again taking the top-5 terms. Tsagkias et al. [151] identify the headline terms as providing a solid baseline for our task, outperformed only by a small number of their methods, which were either based on full articles or on substantial external resources. We executed the queries on Twitter's commercial search service, restricted to English-language tweets. If a query produced less than 25 tweets, we removed the lowest ranking term and re-issued the query, ranking new tweets below existing tweets and repeating until 25 tweets were returned. We evaluated the relevance of these tweets by both in-house assessments and crowdsourcing assessments.

4.6.3 In-house Assessment

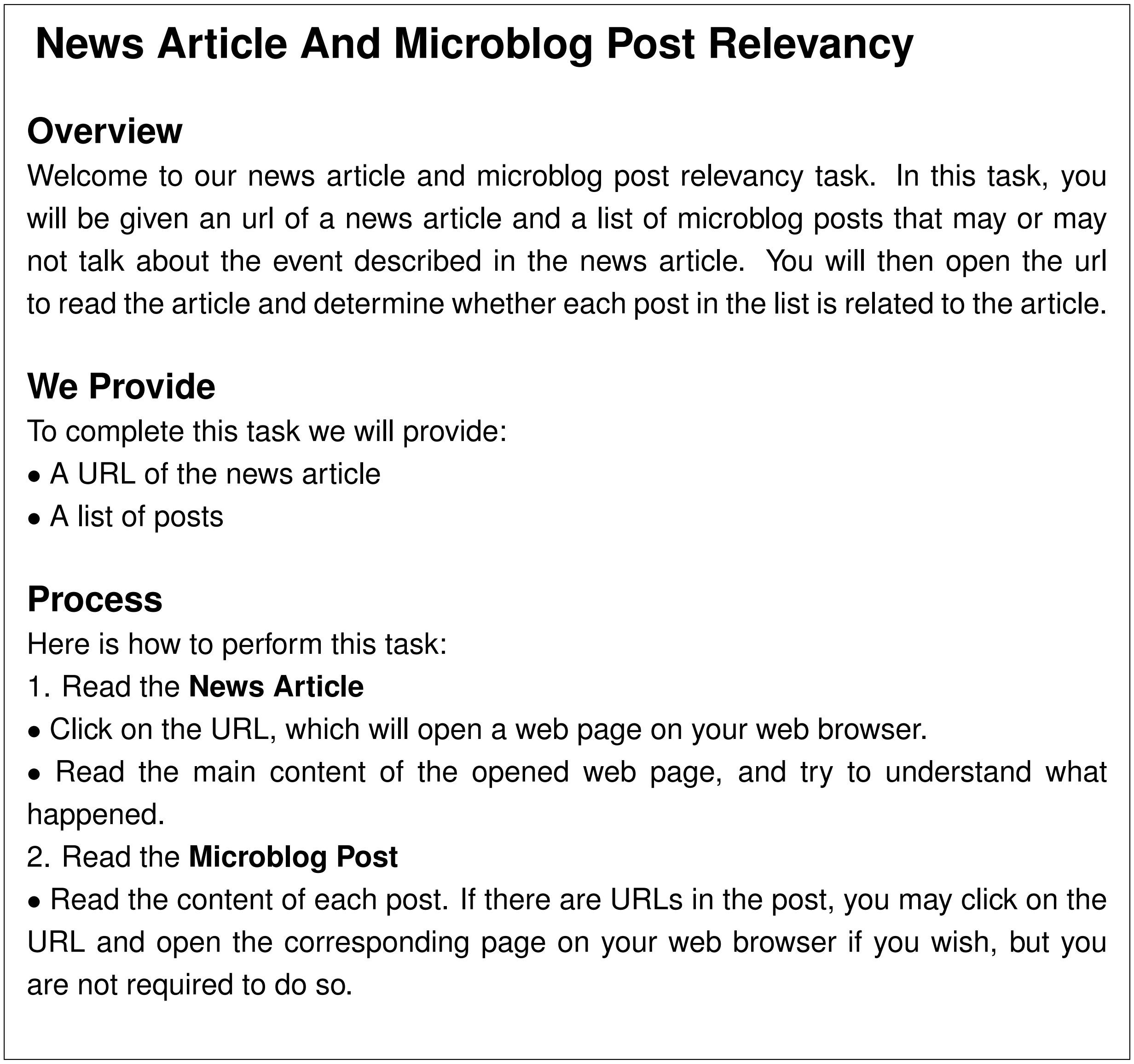
In the in-house assessment experiment, a single member of our research group performed all assessments. For each article, in total 50 tweets returned from both methods were merged and placed in random order on the same page as well as a link to the original news article webpage for relevance assessment.

The assessor first click on the link, open the original news article webpage, read the associated article and formulated a brief statement describing material that could be considered relevant up to one week later. The tweets were then judged one by one in terms of this statement. Judgements were binary, relevant or not. Tweets considered to be borderline were judged as not relevant.

4.6.4 CrowdSourcing Assessment

Crowdsourcing assessment is used more and more frequently for evaluating retrieval methods in IR community [58, 73]. Instead of relying on only well-trained expensive expert assessors, crowdsoucing workers can achieve relevance assessments cheaper and faster. In this step, to validate our single expert assessor judgements, as well as investigate the agreement between the expert assessor and crowdsourcing workers, we designed a set of crowdsourcing jobs and launched them on a crowdsourcing website: CrowdFlower [9].

[9]https://www.crowdflower.com/


News Article And Microblog Post Relevancy

Overview

Welcome to our news article and microblog post relevancy task. In this task, you will be given an url of a news article and a list of microblog posts that may or may not talk about the event described in the news article. You will then open the url to read the article and determine whether each post in the list is related to the article.

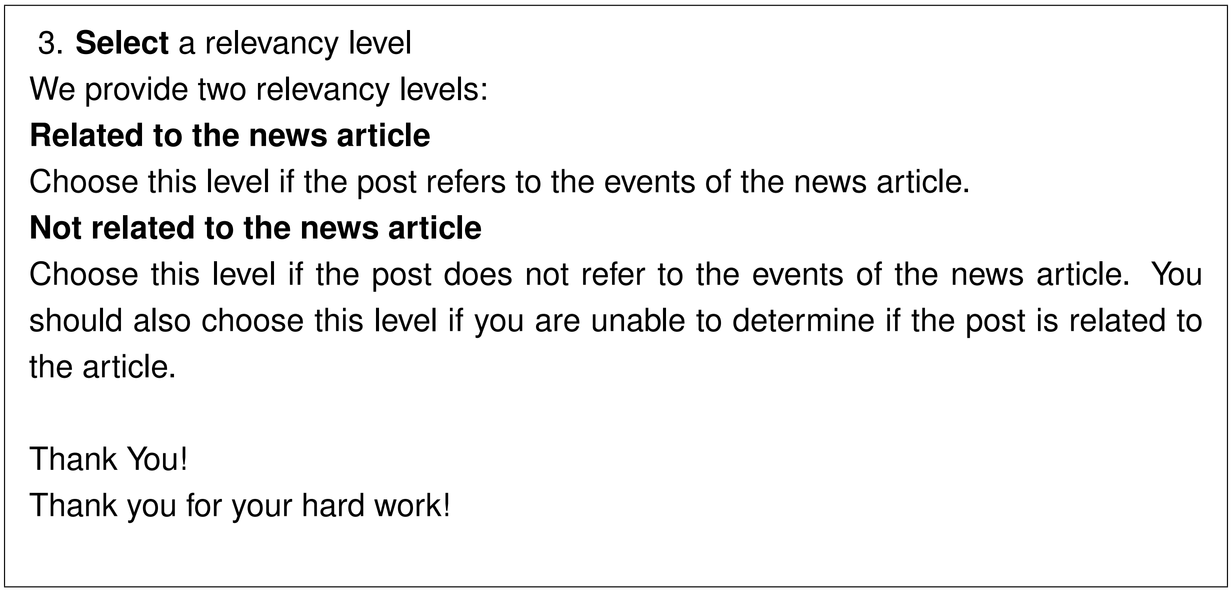
We Provide

To complete this task we will provide:
• A URL of the news article
• A list of posts

Process

Here is how to perform this task:
1. Read the **News Article**
• Click on the URL, which will open a web page on your web browser.
• Read the main content of the opened web page, and try to understand what happened.
2. Read the **Microblog Post**
• Read the content of each post. If there are URLs in the post, you may click on the URL and open the corresponding page on your web browser if you wish, but you are not required to do so.

Figure 4.5: The instruction of CrowdFlower assessment jobs.


3. **Select** a relevancy level

We provide two relevancy levels:

Related to the news article

Choose this level if the post refers to the events of the news article.

Not related to the news article

Choose this level if the post does not refer to the events of the news article. You should also choose this level if you are unable to determine if the post is related to the article.

Thank You!

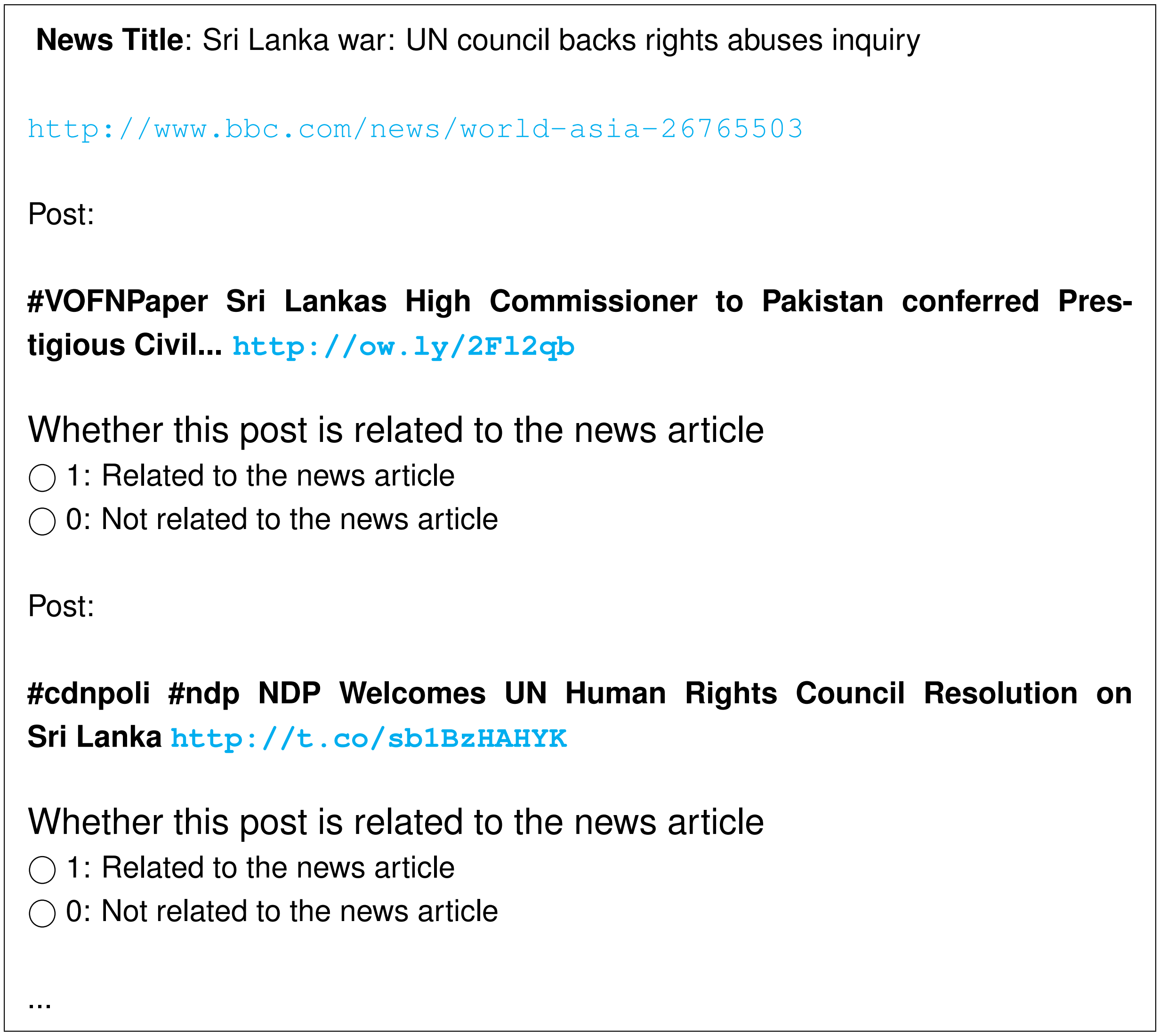
Thank you for your hard work!

Figure 4.6: The instruction of CrowdFlower assessment jobs.(Continued)

CrowdFlower is a crowdsourcing company serving as a platform for both workers and data scientists founded in 2007. Millions of workers, or so-called contributors, all over the world discover online jobs through CrowdFlower's job boards and work on the jobs they are interested in. Data scientists, start-up companies, academic institutions, and others employ the platform to create micro-tasks such as building training data sets for machine learning algorithms, evaluating searching algorithms, and other tasks.

Task Design on CrowdFlower The jobs we designed on CrowdFlower platform were called "News Article and Microblog Post Relevancy". Before starting the task, we provided an instruction for the workers as the webpage shown in Figure 4.5.

After reading the instructions, for each job, 20 tweets were shown to the annotator together with the title of the news article and its hyperlink. We also provided hyperlinks and highlighted the links for the urls in tweets. Annotators were permitted to click on the urls for external information related to the tweets, but they were not required to do so. An example of the judgement task page is shown in Figure 4.7.



News Title: Sri Lanka war: UN council backs rights abuses inquiry

http://www.bbc.com/news/world-asia-26765503

Post:

#VOFNPaper Sri Lankas High Commissioner to Pakistan conferred Prestigious Civil... http://ow.ly/2Fl2qb

Whether this post is related to the news article

◯ 1: Related to the news article

◯ 0: Not related to the news article

Post:

#cdnpoli #ndp NDP Welcomes UN Human Rights Council Resolution on Sri Lanka http://t.co/sb1BzHAHYK

Whether this post is related to the news article

◯ 1: Related to the news article

◯ 0: Not related to the news article

...

Figure 4.7: An example of the task page of CrowdFlower workders.

For each news article, we selected the top 10 tweets from both methods and randomized the order of tweet display. Thus, at most 20 tweets were judged by the CrowdFlower workers. We required multiple judgements from different annotators for each new article.

Jobs Launched To control the quality of these assessments and to avoid random/junk assessments, we used our in-house judgements as a baseline for comparison. Later we also wanted to compute the inter-annotator agreement between in-house assessment and crowdsourcing assessment. Crowdsourcing assessments with 75% of agreement against in-house assessments were labelled as "agreed judgements". At least 3 different "agreed judgements" were obtained for each news article. To achieve this goal, we launched several rounds of the same tasks. Statistical results of each round are showing in Table 4.9.

Round	Paid	# workers	# judgement	# agreed	Average time	Min time	Max time
1	\$0.15	26	198	83	1m18s	0m21s	7m5s
2	\$0.4	56	186	133	3m28s	0m30s	14m6s
3	\$0.4	16	26	14	2m57s	0m18s	14m17s

Table 4.9: Statistical results of CrowdFlower jobs.

Based on major goal, we successfully gathered three round of tasks on CrowdFlower. In the first round, we launched all 66 news articles with 20 tweets of each. 3 different judgements were obtained by paid \$0.15 for each single judgement task. There were 26 different CrowdFlower workers contributed to this job. Among the 198 judgements, 83 of them had at least 75% agreement with our in-house assessment. And 4 out of 66 news articles obtained 3 "agreed judgements". The average finishing time was 78 seconds. The quickest annotator finished in 21 seconds, and the slowest one finished in 425 seconds. Therefore, we launched a second round of the same tasks.

In the second round, 62 news articles with 20 tweets of each were annotated by the workers. Again 3 different judgements were required. Due to the low quality from the first round, we raised our payment to \$0.4 for each judgement task in this round. More contributors were attracted this time, the total number of contributors turned to be 56. Within the total 186 judgements, 133 were "agreed judgements". 49 more news articles obtained at least 3 trusted judgements. The average finishing time was 208 seconds. The quickest annotator finished in 30 seconds, and the slowest one finished in 846 seconds.

In the third and last round, 13 news articles with their 20 tweets were judged twice by different workers. The same amount of money for each judgement task as the second round was paid to the contributors. Overall 16 different workers participated in this round of judgements. 14 of 26 judgements were tested as agreed judgements. The average finishing time was 177 seconds. The quickest annotator finished in 18 seconds, and the slowest one finished in 857 seconds.

Inter-annotator Agreement We computed the inter-annotator agreement both among the crowdsourcing assessors and between the expert in-house assessor and the crowdsourcing annotators. To summarize the overall agreed judgements we obtained for the news articles and tweets, we created Table 4.10. Among the 66 news articles, 13(19.7%) obtained 5 different agreed judgements; 19(28.8%) obtained 4; 26(39.4%) obtained 3 and 5(7.6%) of them obtained 2 agreed judgements; 1 article obtained 1 agreed judgements and 2 articles didn't obtain any agreed judgements.

# agreed judgements obtained	# news articles	# tweets
5	13	260
4	19	380
3	26	520
less than 3	8	160
	total: 66	total: 1320

Table 4.10: Number of agreed judgements obtained among news articles and tweets.

In order to evaluate the agreement among crowdsourcing assessors, we listed the counts of tweets for each agreement case in Table 4.11. Figure 4.8 and Figure 4.9 display the results in graphical representation for both relevance decisions and non-relevance decisions. For example, for a tweet of 5 agreed judgements obtained, we could expect ratio of relevant vs. non-relevant as:

5:0; 4:1; 3:2; 2:3; 1:4; 0:5. Case 5:0 and 0:5 achieved 100% agreement among the 5 annotations. Similarly, 4:1 and 1:4 achieved 80% agreement.

# agreed judgements obtained	percentage agreement	relevance(#/%)	non-rel(#/%)
5 agreed judgements	100% agreement	116/44.6%	76/29.2%
	80% agreement	19/7.3%	33/12.7%
	60% agreement	12/4.6%	4/1.5%
4 agreed judgements	100% agreement	134/35.3%	188/49.5%
	75% agreement	30/7.9%	22/5.8%
	50% agreement	6/1.6%	0/0%
3 agreed judgements	100% agreement	174/33.5%	275/52.9%
	67% agreement	35/6.7%	36/6.9%

Table 4.11: Percentage agreement among the crowdsourcing workers.

We also computed the agreement among workers on relevant and non-relevant judgements. We consider the majority agreement, which means that the percentage agreement is greater than 50%. For the percentage agreement of 50%, we named it as "tie" in the table displayed (Table 4.12. The percentages are the number of majority agreed judgement tweets divided by the total number of agreed judgement tweets.

Relevant	Non-relevant	Tie
44.8%	54.7%	0.5%

Table 4.12: Majority agreement on relevant and non-relevant judgements.

Although we exploited in-house assessor as comparative decision for selecting the agreed judgements of crowdsourcing assessments, we also concern about the real agreement between these two different kind of annotators. Table 4.13 shows the percentages of agreement and disagreement judgements between the majority decision of all crowdsourcing assessors and the expert assessor. 39.6% agreement on relevant judgement and 38.0% agreement on non-relevant judgement, which means in total 77.6% agreement and 8.5% disagreement.

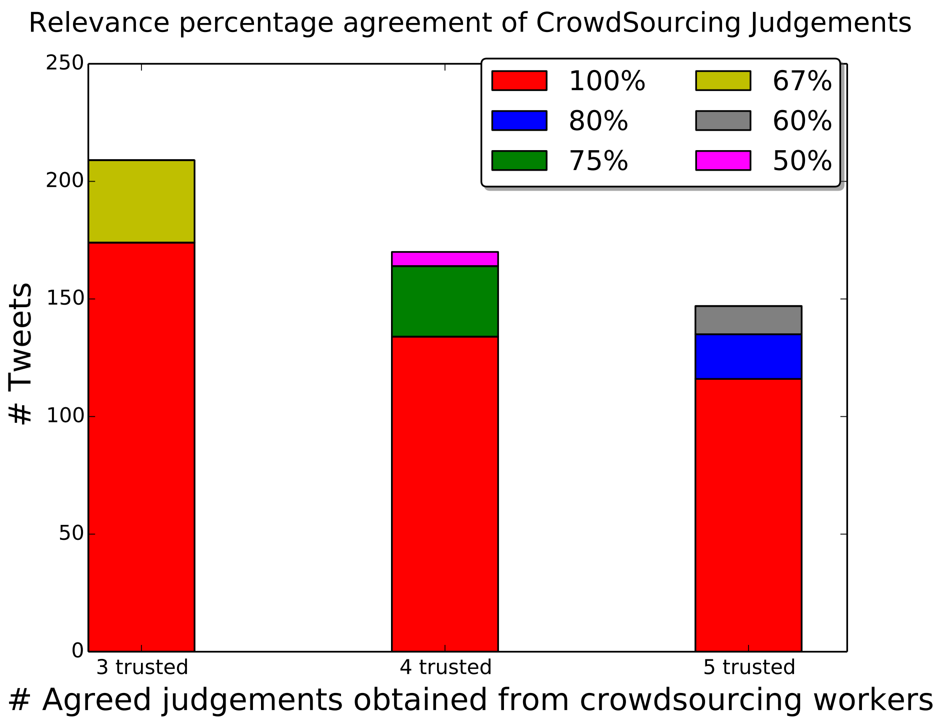


Figure 4.8: Relevance percentage agreement of CrowdSourcing Judgements.

4.6.5 Results and Discussion

Evaluation results are shown in Figure 4.10, which gives average values for several standard effectiveness measures. Although ERR and NDCG are designed for graded relevance values, they adapt naturally to binary values (i.e., two relevance grades). For the in-house assessment, NDCG is computed down to depth 25 and normalized by assuming the collection contains an unlimited number of relevant tweets. Similarly, the results of crowdsourcing assessment are displaying in right side of the figure. The depth computed down of the metrics is 10 for crowdsourcing assessment, since we only asked the workers to judge 20 tweets (top 10 from each method). The results shown in the table are calculated using all the crowdsourcing assessments (including agreed judgements and non-agreed judgements).

All improvements over the baseline are statistically significant ($p < 0.01$) under a two-sided

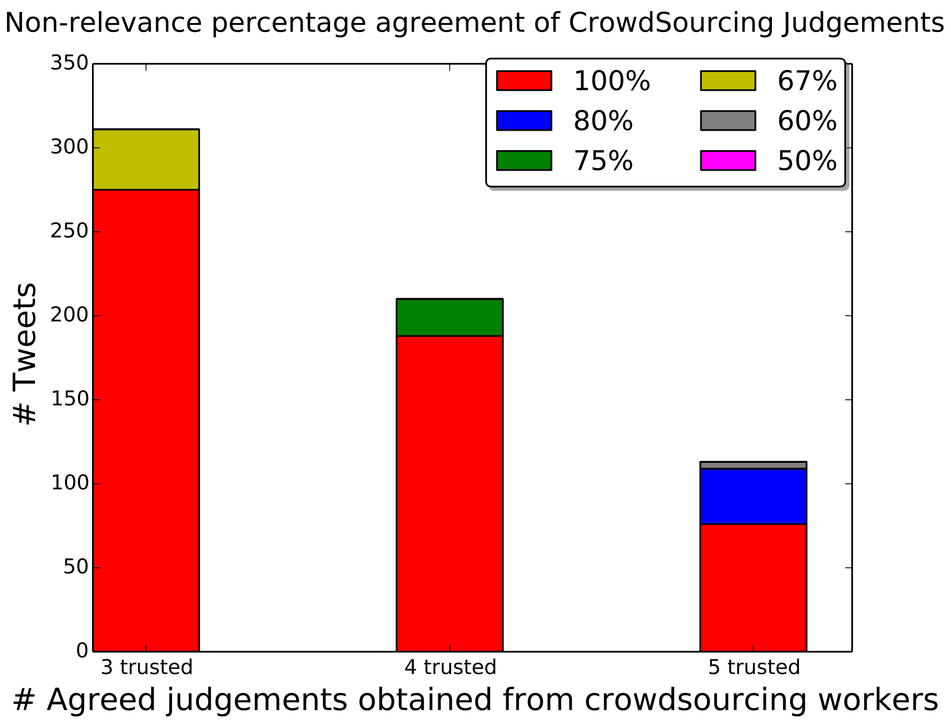


Figure 4.9: Non-relevance percentage agreement of CrowdSourcing Judgements.

paired t-test. Beyond statistical significance, we would expect these improvements, e.g., more than 50% in precision@5, to be practically significant, i.e., noticeable at the user level.

For the news article used as an example in the introduction of this chapter (“More than 100 Congolese refugees killed in boat accident, Uganda says”) the algorithm produced the ranked query: $Q = \{$“albert”,“boat”,“lake”,“accident”,“capsized”$\}$. As with any query re-weighting method, there were misses as well as hits. For an article on President Obama’s new health secretary the query over-emphasizes the departing secretary
$Q = \{$“kathleen”,“sebelius”,“secretary”,“obama”,“obamacare”$\}$, although many tweets were still relevant. In some cases, the generated query merely copied terms from the headline. And, of course, in others the headline outperformed it.

	In-house assessor		
		Relevance	Non-Relevance
Crowdsourcing Workers	Relevance	523(39.6%)	101(7.7%)
	Non-Relevance	11(0.8%)	501(38.0%)
	Tie	37(2.8%)	147(11.1%)

Table 4.13: Agreement and disagreement between the crowdsourcing assessor and in-house assessor.

	In-house assessment					Crowdsourcing assessment			
	P@5	P@10	P@25	NDCG	ERR	P@5	P@10	NDCG	ERR
Headline	0.391	0.323	0.259	0.298	0.353	0.461	0.389	0.431	0.413
Succinct query	0.594	0.542	0.462	0.489	0.481	0.661	0.612	0.640	0.521

Figure 4.10: Experimental results. All improvements over the baseline are significant (two-sided paired t-test, $p < .01$).

4.7 Summary

In this chapter, we explore probe queries as a method for extracting succinct queries from full documents. We apply our algorithm to the problem of linking mainstream news articles to social media. Our evaluation shows statistically and practically significant improvements over baseline queries derived from the headlines of the news articles from both in-house assessment and crowdsourcing assessment.

In candidate term extraction step, we compared two different key word extraction method. We also spent effect to try to improve the similarity function in Equation 4.8, which must recognize related material while avoiding near-duplicate material. We experimented with a learning to rank. We compared several simple methods for re-ranking the candidate terms based on their similarity score with the news articles.

To evaluate our algorithms, both in-house expert assessor and crowdsourcing workers performed judgements. When designing crowdsourcing tasks, we considered quality control strategies. An analysis of inter-annotator agreement was also undertaken.

Chapter 5

Real-time Tracking and Push Notification

In Chapter 4, we discussed the succinct query generation problems for tracking news, which a news article was given as the input. If the news event in the news article is a topical event, the query terms can be used to track this topical event from social media streams, such as Twitter. The tracking provides a means for users to keep up-to-date on topics of interest to them.

Except news articles, user profiles, user interest descriptions or even explicit sets of topic terms are possible contents to express users' information needs. If care is taken, these updates may even be pushed directly to the user through notifications on mobile devices or desktops. However, for push notifications to be successful, the user must be given means to control the frequency and volume of updates, avoiding indiscriminate and unwanted notifications. This frequency and volume depends both on the interests of the user, with topics of greater interest updated in greater volume, and on the topics themselves, with some topics naturally receiving updates more frequently than others.

We might update a user interested in polls for the 2016 U.S. presidential elections many times a day during the election cycle itself, but with updates stopping altogether after November 8. We might update a user interested in California residential water restrictions only when these restrictions change, perhaps a few times a year, but interest in the topic might persist for many years, as long as the user is a resident of the state. For causal sports and entertainment topics (cricket or the Kardashians), a user may not desire more than a few of the most significant updates per day, regardless of events taking place. For topics of great personal importance (a tornado warning or friend's wedding), we might push all updates.

The pushing problem can be compared with the secretary problem [56]. The secretary problem is a famous online decision uncertainty problem. We are interviewing a known number of candidates for hiring one secretary. After each interview, a decision has to be made immediately and if rejected, the candidate will never come back. The goal of this problem is to maximize the probability of getting the best candidate. A most straightforward extension of the classic secretary problem is aiming to hire k secretaries, which is named as k choice secretary problem or multiple-choice secretary problem [78, 175]. Up to k secretaries are allowed to be hired during the interview period. A decision need not be made immediately after each interview. But it's better to be as fast as possible.

The hiring problem was introduced by Broder et al. [20] as another uncertainty problem. A company wants to hire a fixed number of employees, where the number of employees can be from one to up to any finite number. An unknown number of candidates are sequentially interviewed, with immediately decision made by the company and ability no revoke that decision. In solving this problem, we must make a tradeoff between hiring quality and hiring rate.

In this chapter, we view our pushing notification problem as being in the same category as the above two problems, but as a separate problem. While monitoring a social media stream (i.e. the tweet stream), which contains an unknown number of sequentially incoming tweets, a pushing service system will push up to k interesting tweets to the user. The system may also choose to mute itself to avoid annoying the user if nothing interesting is appears. The decision to push a tweet may not have to be made immediately. Instead, systems can make decision within an acceptable delay time interval.

Zhao and Tajima [175] frame a retweet recommendation problem as a multiple-choice secretary problem. They examine Twitter "Portal accounts", which retweet selected tweets for their followers, and consider a number of strategies for tweet selection. They propose and compare a number of online and near-online decision methods, including a history-based threshold algorithm, a stochastic threshold algorithm, a time-interval algorithm and an "every k-tweets" algorithm.

The remainder of the chapter is organized as follows. We introduce the TREC 2015 Mi-

	Secretary Problem	Hiring Problem	Pushing Problem
# candidates	N	unknown	unknown
# hires	k	k [1,)	[0, k]
goal	maximize probability of getting the best k candidates	trade-off between hiring quality and hiring rate	maximize sum of gain (and pain)
decision	immediately; no revocable	immediately; no revocable	within acceptable delay

Table 5.1: Comparisons among Secretary problem, hiring problem and pushing problem

croblog Track in Section 5.1, which provides an experimental and evaluation platform for the real-time tracking and push notification problem, and frame the significant factors that lead to successful solving of the problems. Section 5.2 discusses the understanding of TREC 2015 Microblog Track user interest profiles and tweets, as well as the algorithms for scoring the relevance between tweets and the user profiles. Section 5.3 provide discussion of different pushing notification strategies: without any human involvement; with daily relevance feedback from human assessment. Section 5.4 provide the evaluation results of different pushing strategies and analysis on the performance of pushes. A brief summary and discussion are included in Section 5.5.

5.1 TREC 2015 Microblog Track

The TREC Microblog tracks provide an experimental forum for research groups working in real-time tracking and retrieval of social media stream area. In 2015, the track [98] was a real-time filtering task with the goal of pushing interesting and novel tweets to users with respect to their interest profiles. The track required participating groups to monitor the live “spritzer” stream provided by Twitter over a period of ten days in July (July 20, 2015, 00:00:00 UTC to July 29, 2015, 23:59:59 UTC), selecting tweets relevant to 225 pre-defined interest profiles, each expressed through statements modelled after TREC *ad hoc* topics.

Topics provided to participants in the TREC 2015 Microblog Track include a short query-like description of the information need, called in TREC jargon the “title”. For example, topic MB235 (shown in Figure 5.1) has the title “California residential water restrictions”. Other fields in a topic elaborate on the information need, providing a more complete indication of what is and is not relevant.

```
<top>
<num> Number:  MB235
<title> California residential water restrictions
<desc> Description:  Find descriptions of the effects of
residential water use restrictions due to the drought in
California.
<narr> Narrative:  The user is looking for information on the
effect of residential water use restrictions in California
caused by the on-going drought.  While official reports on the
efficacy of the restrictions are relevant, the user is also
looking for first-hand reports of how residents are complying
and coping with the restrictions.
</top>
```

Figure 5.1: TREC 2015 Microblog Track Topic MB235

.

Participants in the track filtered the live Twitter “sprinkler” stream over ten days, selecting those related to the 225 interest profiles and recording their ids for submission to TREC. In addition to the ids, a push time was recorded for each tweet, indicating the time the system decides to push it to the user. A maximum number of 10 tweets per day, per interest profile can be pushed to the users. Additional tweets beyond ten per day per interest profile are simply ignored by the evaluation metrics. After experimental runs containing these tweets were submitted to TREC, 51 of these profiles were chosen for judging. Tweets were pooled and judged on a three-point scale.

The evaluation measures considered both the relevance of selected tweets and the time at which they were putatively pushed. In addition, the measures accounted for retweets, near-duplicate tweets, and other redundancies, reflecting an expectation that a user would not want to receive notifications about the same thing over and over again. In evaluating a run, a tweet was considered to be redundant, providing no gain, if it did not contain substantial new information not found in previously pushed tweets. Retweets were given special handling, with a push of a retweet treated exactly as if the original tweet were pushed instead.

Along with this mobile notification scenario (called "scenario A") the evaluation supported a daily digest scenario (called "scenario B"). At the end of each of the ten days, participating systems returned a ranked list of tweets from that day, just as if a user was sent a summary of the day's events by email. For this second task, standard evaluation measures for ranked retrieval can be applied, provided that they also appropriately account for redundant content.

The primary evaluation measure for scenario A is expected latency-discounted gain (ELG).

$$ELG = \frac{1}{N}\sum_{i=1}^{N} G_i D_i \tag{5.1}$$

For N pushed tweets, G_i is the gain associated with tweet i, as indicated by the assigned relevance grade, after adjusting for redundancy. The discount applied to tweet i is $D_i = \max(0, (100-L_i)/100)$, where L_i is the latency in minutes between the time the tweet appears on the stream and the time the system decides to push it. ELG is computed on a daily basis, over the tweets pushed by a system that day, with the system's overall score averaged across all topics and all days.

The secondary evaluation measure for scenario A is normalized cumulative gain (nCG).

$$nCG = \frac{1}{Z}\sum_{i=1}^{N} G_i D_i \tag{5.2}$$

Z is the maximum possible gain in the pool, given the 10 tweet per day limit. The gain and discount of each individual tweet is computed as ELG.

ELG and nCG have an interesting discontinuity when a system decides to push nothing. For some topics on some days, when no relevant tweets appear on the stream, this is the correct thing to do. On such days, a system pushing a large number of non-relevant tweets will receive a score of zero. To reward systems that push nothing on such days, the values of ELG and nCG are defined to be 1 (full score). On the other hand, for systems pushing nothing on days when relevant tweets appear on the stream, the values of ELG and nCG are defined to be 0. Since no relevant tweets appeared for many topics on many days, the "null" or "empty" strategy of never pushing anything receives a positive score under ELG and nCG. Indeed, this strategy forms a challenging baseline, which many participating systems failed to beat.

To achieve good results for scenario A, systems must successfully address three requirements implicit in the task:

1. A requirement to score individual tweets with respect to relevance. As a simplification, the evaluation focused on topical relevance. Social signals, such as the prominence of the source or its connection to the user, were not considered, so that the relevance of a tweet is primarily determined by its content.
2. A requirement for novelty, so that the system does not push redundant information. This requirement was operationalized by the assessors considering tweets chronologically: if two textually similar tweets arrive at different times, the later tweet is considered redundant if it does not contain substantive information beyond that found in the earlier tweet [155]. Again, only tweet content is considered, so that a duplicate tweet from a more prominent or authoritative source would still be considered redundant.
3. A requirement to avoid pushing non-relevant information altogether. The evaluation measures for scenario A explicitly rewarded systems that avoided pushing non-relevant information. Thus, appropriate selection of thresholds was critical to success. In some cases, the ideal response for a given day was to push nothing, and the "empty" strategy of never pushing anything formed a challenging baseline that many systems failed to beat.

Since the first two requirements are inherent in any ranking task, we focus on the third requirement. After demonstrating the impact of ignoring the third requirement, we consider various strategies for establishing and maintaining thresholds for pushing tweets. We examine strategies under two assumptions: with and without user feedback. When feedback is not available, thresholds are established from historical information. When feedback is provided, it is limited to once-per-day judgements based on the scenario B output. Of particular importance is the establishment of a global score threshold, applied across all topics in the absence of feedback. Our best technique takes advantage of daily feedback in a simple yet effective manner, achieving the best known result reported in the literature to date.

5.2 Relevance between User Interest Profiles and Tweets

In this section, we focus on the discussion of computing relevance between user interest profiles and the individual tweets. User interest profiles are the expression of user information needs. We apply key word extraction methods to extract key words from descriptions in the interest profiles and pseudo-relevance feedback methods to expand the query terms. Tweets are short sentences with at most 140 characters. Words, hashtags, mis-spellings, URLs, emotions are common components of tweets. Based on the evaluation rules of TREC Microblog Track, we need to pre-filter the tweet streams and control the quality of candidate tweets for pushing to the user. We propose a simple scoring function based on coordinate matching to calculation the relevance score between each user interest profile and each individual tweet.

5.2.1 User Interest Profiles

The user interest profiles adopt the standard TREC topic format. There are three fields: title, description and narrative. Titles contain several key words or key phrases; descriptions are one-sentence statements of the users' information needs; narratives are paragraph-length descriptions of the tweets that the users want to receive. Track participants were permitted to use all these fields for filtering purposes. To implement the filtering tasks, we built a feature (term) vector for each interest profile, and assigned different weights to different types of terms.

Feature Vector

The feature vector representing for each interest profile consists 3 components: title features, narrative and description features, expansion features.

$$< title\ features,\ narr + desc\ features,\ expansion\ features > \tag{5.3}$$

The title features are extracted from title fields. Each title was tokenized by space and punctuation. The stop words were removed from these tokens. Additionally, for noun tokens, we derived both singular and plural forms of these noun tokens. We processed the title of each interest profile, and added processed tokens to the feature vectors of this profile.

To extract important words from descriptions and narratives as narr+desc features, we applied a pointwise K-L divergence method (the same method as described in Section 4.1.1), which was also applied later in the process to generate expansion terms. As we discussed earlier in this thesis, we can exploit the pointwise K-L divergence method using the following equation to rank words:

$$p(t)\log(\frac{p(t)}{q(t)}), \tag{5.4}$$

where $p(t)$ is the relative frequency of term t in the foreground model and $q(t)$ is the relative frequency of term t in the background model collection.

In the case of discovering top-ranked words in narrative and description for each interest profile, we treat the narrative field and the description field in each individual interest profile as the foreground model and all the narrative fields and all the description fields in all 225 interest profiles as the background model. For each user profile, equal weights are assigned to each sentence in both the narrative and description field. The sentences from these two fields are combined together (order doesn't mater) and tokenized as the title field by white space and punctuation. Stop words removal and singular/plural stemming are treated the same as we did for title fields.

By applying pointwise K-L divergence, special terms for each individual narrative and description are ranked higher. The terms that are commonly expressed in many interest profiles are ranked lower. For example, in the interest profile shown in Figure 5.1, terms from narrative and description fields are ranked as the following Table 5.2:

Rank	Term	Rank	Term
1	drought	...	...
2	complying	19	user
3	residents	20	relevant
4	reports	21	information
...	...	22	find

Table 5.2: Term ranking of narrative and description fields of Topic MB235.

In this table, the terms "drought", "complying", "residents" and "reports" are top ranked terms, while “user”, “relevant”, “information” and “find” are low ranking terms. The terms that appear in title fields (both singular and plural format) are removed from the ranking list of narrative and description features.

We take the top-10 ranked terms from this ranking to form a set of important narrative and description terms. In the runs that used narratives and descriptions, these terms are added to the feature vectors. Except the information we can extract from the interest profiles, we also expand terms in the title fields using a pseudo-relevance feedback method.

Pseudo-Relevance Feedback

Since relevant tweets may not contain all, or even any, of the title terms we employed a pseudo-relevance feedback step to expand the title terms. We take a large collection of historical tweets as our background model. This collection was collected through the Twitter Streaming API from November 2013 to March 2015. We restricted the collection to English-language tweets on the basis of the language field associated with each tweet. There are in total approximately 291 million tweets in the collection.

To build a foreground model for each user profile, we take the title terms as query, and search the query in Twitter search engine. The top 1000 tweets are crawled on the search result pages as our foreground model. In the top retrieved tweets, URLs in each tweet are replaced by their web page titles, if the ¡title¿ tag existed in the HTML source of the web pages.

The TREC evaluation lasts for 10 days. For some popular topics during this period, the key words and sub-topics of the Twitter user discussion may change every day. To keep the foreground models up-to-date, we re-built the foreground models at the end of every day during the evaluation period. A ranking of terms for each user profile can be generated by the scores for each term from pointwise K-L divergence equation. The top 5 hashtags and top 10 other terms from the ranking are added to the feature vector of each profile.

5.2.2 Pre-processing Tweets

Tweets are relative short and informally written. As a requirement of the TREC evaluation, only tweets understandable by English readers could be assessed as relevant. Thus, we need to detect English tweets. We applied a simple method to remove non-English tweets and partially kept retweets. To better understand the tweets, a tokenizer that is designed for English Twitter text is used. No stemming or stop word removal was utilized on tweets, but tweet quality was considered.

Language detection and Retweets

Tweets written in a language other than English would be judged as not relevant based on guidelines of Microblog Track. To obtain English-only tweets, we examined the language tag for each tweet feed by the Twitter streaming API, and only keep the ones with ”en” value in their language field. In addition, we remove all NON-ASCII characters from the tweets, which also helps remove non-English tweets. We assume that after the removal, if a tweet can be understood by English readers with only the ASCII characters, it should be considered as an English-language tweet.

A retweet in Twitter slang means a tweet reposted or forwarded from another tweet posted by another user. The assessments of retweets in the TREC 2015 Microblog Track is different from the same tracks in previous years. In previous years, all retweets were automatically considered as non-relevant tweets to all the topics, no matter what contents were included in the retweets. In 2015, the retweets returned were replaced by the original tweets that were reposted. In this case, all additional commentary in the tweets was still ignored, and the created time of each retweet was considered as the creation time of the original tweet. Thus, there was still a high risk associated with returning a retweet, given that the original tweets are older than the retweets. In our submitted runs, we ignored all retweets in Scenario A and keep them in Scenario B. Since the evaluation metric of Scenario A contains a latency penalty component, while no latency penalty applied in metrics of Scenario B.

Tokenizer

Our tokenizer was based on Twokenize, which is a tokenizer designed specificly for English tweet text. It is part of CMU’s Tweet NLP project [1]. Twokenize uses various regular expressions

[1] www.ark.cs.cmu.edu/TweetNLP

to keep emoticons, numbers, emails, URLs, abbreviations, arrows etc. complete.

A hashtag in Twitter slang is a special word or phrase started with a hash symbol (#) to identify tweets for special topics. For the hashtags, Twokenize can recognize the hashtags in tweet contents. We modify Twokenize, to add the base term without the '#' symbol as a token, as well as the hashtag itself.

Since tweets are short, usually each tweet is focuses on only one topic. As a result we place no importance on terms repeated multiple times. Thus, we do not count the term frequencies of the tokens generated from our tokenizer. We treat the tokens as token sets, which means each token was only counted once in each tweet.

Tweet Quality

Each tweet is written by an individual Twitter user. Thus, the quality of tweets varies a lot. Twitter users post tweets for different purposes. A large amount of tweets posted everyday are meaningless, low quality, or junk tweets. With a tweet quality filter, we can filter out low quality tweets and some of junk tweets. Based on previous experimental experience, our quality filter uses a set of arbitrary thresholds to detect low quality tweets. Low quality tweets are ignored in the tweet streams.

Length of tweet Although tweets are at most 140 characters, short tweets are hard to extract topics from. The average length of tweets are 9-10 words. Tweets that are too short are even harder to extract meaningful information from. Thus, we assume, any tweet that has fewer than 5 tokens is a low quality tweet.

Number of Hashtags Hashtags are used to identify topics for tweets. Each tweet is usually talking about one topic. Thus, we believe that tweets with more than 3 hashtags are low quality tweets. Tweets with multiply hashtags are usually advertisement tweets or junk tweets which try to catch Twitter user's attention.

5.2.3 Relevance Scoring

According to above description, user profiles are converted to term vectors and each individual tweet is converted to a token set. In our system, only tweet content is considered for relevance scoring. While this approach proves adequate for TREC evaluations, we recognize that a more realistic system would also consider social signals and other non-content features. A simple vector space model (derived from coordinate matching method and developed through pilot testing) is applied to calculate relevance score between each incoming tweet and each user profile.

Given the short length of tweets, many features typically used in relevance formulae, including term and document frequency features, appear to have limited value for ranking and scoring tweets. For content matching of tweets, query term occurrence appears to be the key feature, with the occurrence of title terms having greater importance than the occurrence of expansion terms.

There are three types of feature terms: title, narrative+description, expansion. We denote types of feature terms by $i = \{t, n, e\}$, where t stands for title, n stands for narrative+description and e stands for expansion.

$$rel = (\sum_{i=\{t,n,e\}} w_i * N_i) * (\frac{N_t}{|T|}) \tag{5.5}$$

In this equation, w_i denotes the weight for type i and N_i stands for the number of times type i feature terms appeared in the tweet. We normalized the score by ratio of title terms appeared in the tweet, where $|T|$ denotes the total number of title terms of the user profile. We assumed that the more title terms appeared in the tweet, the more relevant the tweet would be. Based on pilot experiments, weights were set to $w_t = 3$, $w_n = 2$ and $w_e = 1$.

5.2.4 Novelty

We de-duplicated tweets by computing unigram overlap between each new tweet and the tweets previously pushed for a given topic across all days. Tweets with 60% or more overlap between previously pushed tweets were discarded and not further considered. As with other parameters, we based this overlap limit of 60% on pilot experiments conducted prior to TREC 2015.

5.2.5 Thresholding

Thresholds for pushing tweets were based on the relevance score in Equation 5.5. Each day, a threshold was selected, and only tweets exceeding this threshold were considered for pushing. In addition, the track set a limit of $k = 10$ on the number of tweets that could be pushed each day. Once k tweets are pushed, nothing further is pushed that day.

Because of its simplicity, Equation 5.5 has the interesting property that reasonable relevance thresholds are approximately the same across queries. The rel score has nothing to do with different topics or any single terms. Our simplest thresholding strategy is thus to select a single static threshold (*GT*) across all queries and days. A simple dynamic strategy is to base on the top k from the previous day, selecting as a threshold the score of the kth tweet. However, even under this strategy, we do not lower the threshold below an global minimum, selecting as a threshold *max(GT, top-k yesterday)*.

For our TREC 2015 experiments, we used a global threshold of $GT = 5$. To better understand the impact of this threshold, Figure 5.2 shows the performance of our system for these two strategies across a range of global threshold values. The baseline for this plot is the performance of the null or empty strategy, i.e., never pushing anything, with low threshold values underperforming it. A global threshold of $GT = 6$ slightly outperforms our default threshold of $GT = 5$, which retain for the remainder of this paper.

The oracle run shown in the plot represents the performance achievable if we made an ideal selection of the global threshold for each topic at the beginning of each day. Clearly substantial performance improves could be achieved through improved threshold selection. In the next section we explore a dynamic emission strategy that uses feedback from each day digest (Scenario B) to improve threshold selection for the following day. We imagine a user glancing at this digest once per day, indicating which tweets are relevant, and which are not.

5.3 Pushing Notification Strategies

Since in Scenario A setting, the pushing system has an upper bound of the number of tweets that it can push to each user every day. There may be more relevant and non-redundant tweets than

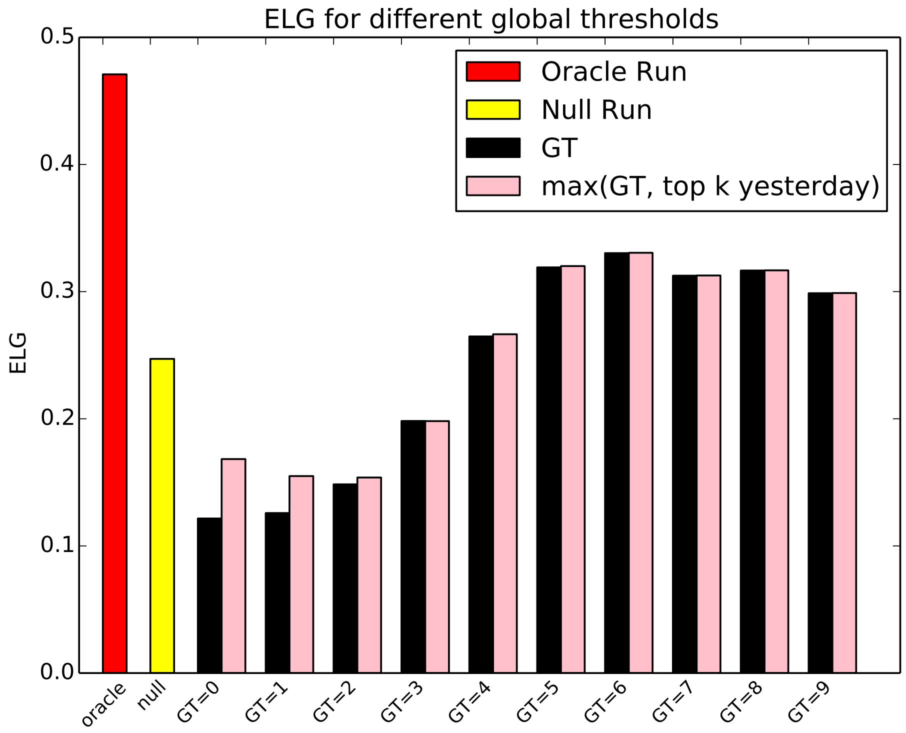


Figure 5.2: ELG for different global thresholds and oracle run.

the upper bound. They can't be all pushed to the user. Thus, we need some strategies to choose which portion of the candidate tweets that should be pushed to users.

During the TREC 2015 implementation period, we considered automatic strategies that did not require any human interruption. Two automatic strategies are: fix time window pushing and dynamic time window pushing. In posterior experiment period, we also simulate daily user feedbacks to improve our automatic strategies.

5.3.1 Automatic Pushing Strategies

Fix Time Window Pushing

The maximum acceptable latency in the Scenario A evaluation metric is 100 minutes. Any tweet that is pushed later than 100 minutes will not receive any credit, even though it might be highly relevant to the user interest profile. Thus, we designed our fix time window pushing strategy based on the length of this latency penalty.

The threshold of this strategy follows our thresholding method. A dynamic threshold which is the score of the top 50th tweet from the previous day with a combination of the global threshold 5. We never lower the threshold below the global minimum, with is annotated as *max(GT=5, top-50 yesterday)*.

Using this strategy, we push tweets periodically. To at least obtain some credit from the latency penalty, we design our time window as 90 minutes. In every 90 minutes, we select the highest score tweet whose score is also higher than the threshold, and the tweet has to be non-redundant to any previously pushed tweets. If there is not any tweet during the 90 minute window that meets our requirement, the slot will be carried to the next 90 minute window. Whenever we have pushed 10 tweets for the user interest profile one day, we will stop pushing for the day.

The choice of 90 minute window size is arbitrary. In the posterior experimental period, we swept the window size to get a best choice of our fix window size. However, surprisingly, the ELG scores of different fix time window sizes almost decrease linearly from one minute to 100 minute. It means that the smaller window size is, the higher ELG score we will obtain. The sweeping result is showing in Figure 5.3. Therefore, in later improvements with simulating user feedback parts, our window size is set to one minute.

Dynamic Time Window Pushing

The dynamic time window pushing strategy starts and ends the time window dynamically. The starting and ending points of each time window depends on previous tweets in the stream.

There are two thresholds for each user profile: in dynamic time window pushing strategy k_0, and k_1, where $k_0 \leqslant k_1$. k_0 was the lower bound of relevant scores for the profile. Here we

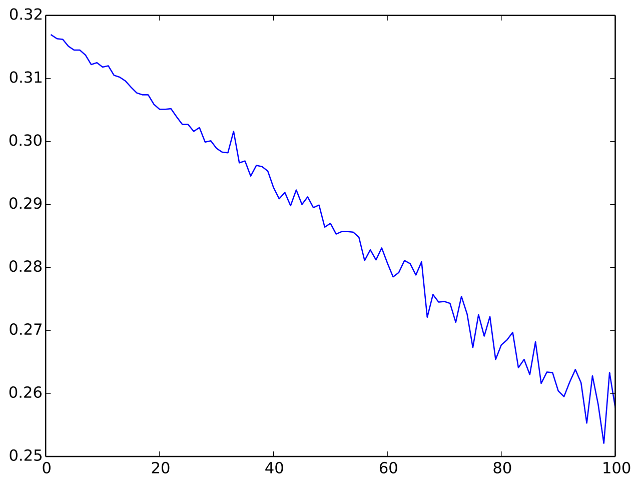


Figure 5.3: Sweep of fix time window size on ELG score.

made several assumptions: any tweet that had score higher than k_1 was relevant to the user's information need; tweets that were scored lower than k_0 were not relevant; tweets with scores between k_0 and k_1 were considered "potentially relevant". The value of k_0 and k_1 were based on the scores of returned tweet in the previous day, and would be updated every night. In our submitted system, k_0 was set to the top 10th tweet score in the previous day; k_1 was set to the top 5th tweet score in the previous day; the maximum waiting time was set to 80 minutes. A dynamic k-minute window was used for this algorithm. Algorithm 21 shows the strategy we applied.

5.3.2 Push with Relevance Feedback

Under Scenario B participants submitted a ranked list of tweets for each topic for each day, providing a daily digest of events for the hypothetical user. In this thesis, we use each day's digest to provide relevance feedback for determining thresholds for the following day. Each day during the evaluation period we saved a ranked list for submission to TREC. While these were not judged until after evaluation period, we are assuming here that they are judged at the end of each day, with relevance information available for immediate use. We imagine a user interacting

Algorithm 1 Dynamic Time Window Algorithm

Data and Result

```
1: Data: Streaming of tweets
2: Result: Relevant tweets to the user profile
```

Variables

```
1: $max\_waiting \longleftarrow k\ mins$
2: $cur\_time \longleftarrow now$
3: $last\_time \longleftarrow cur\_time - max\_waiting$
4: $time\_out \longleftarrow 0$
5: $k_0 \longleftarrow score\ of\ top10\ tweet\ yesterday$
6: $k_1 \longleftarrow score\ of\ top5\ tweet\ yesterday$
7: $candidate \longleftarrow highest\ score\ tweet\ between\ cur\_time\ and\ last\_time\ with\ score \geqslant\ k_0$
```

Steps

```
1: while $True$ do
2:     $curr \longleftarrow highest\ score\ tweet\ between\ now\ and\ last\_time\ with\ score \geqslant\ k_0$
3:     $curr\_time \longleftarrow time\ of\ curr$
4:     if $curr > k_1$ then        ▷ If a tweet has score greater than $k_1$, report it immediately, and restart the waiting window.
5:         Report $curr$
6:         $last\_time \longleftarrow curr\_time$
7:         $time\_out \longleftarrow 0$
```

Algorithm 1 Dynamic Time Window Algorithm (continued)

```
8:      else if $curr \geqslant candidate$ then   ▷ If a tweet is better than the candidate tweet, replace the
        candidate with the current tweet, and restart the waiting window.
9:          $candidate \longleftarrow curr$
10:         $last\_time \longleftarrow curr\_time$
11:         $time\_out \longleftarrow 0$
12:     else                                      ▷ If a tweet is worse than the candidate tweet.
13:         if $time\_out < max\_waiting$ then
14:             $time\_out \longleftarrow now - curr\_time$
15:         else   ▷ Report the candidate, which is the best one during the waiting window. Then
        restart the waiting window.
16:             Report $candidate$
17:             $last\_time \longleftarrow now$
18:             $time\_out \longleftarrow 0$
19:         end if
20:     end if
21: end while
```

with the results once per day, providing feedback as a way of adjusting the filter for the next day. While in practice this daily interaction might not be practical, i.e., not something all users will want to do, the results provide a sense of what gains could be achieved through ongoing feedback.

For our purposes, we consider only the highest ranking 10 tweets from each of these day's digest, as they were included in the judgement pool that TREC used to evaluate Scenario B. To maintain independence from any discrepancies in the replayed implementation, we use the top 10 results from our submission to the track. Furthermore, we only make use of the raw gain assessment of tweets (i.e., not interesting (0), interesting(0.5), and very interesting (1)) and not any cluster-associated redundancy, in order to more accurately model a user providing feedback within reasonable levels of effort.

Tweetid	Gain	Score
623036642367029248	1.0	9.00
623132347995717632	0.5	6.67
623096759351259136	1.0	5.33
623103407314726913	1.0	5.33
623228158506958848	0.5	5.33
623117886035468288	0.5	4.67
623141164447731713	1.0	4.67
623192011995123712	0.5	4.67
623197900768706560	1.0	4.67
623030980060708864	1.0	4.00

Table 5.3: Exemplar tweets from topic MB243 on July 20th for Scenario B for the UWaterlooBTEK system. Note that scores based using the proposed relevance function fall into "blocks".

Equation 5.5 often assigns the same score to several tweets. We use the term "score block" to denote a set of tweets with the same score. Accordingly, we say that for a score s_i, it has a corresponding score block SB_i. To determine a dynamic threshold using relevance feedback, we begin by combining all relevance feedback received into a single list of score blocks. At the end of day 1 for a particular topic, we have 10 tweets worth of feedback, on day 2, 20 tweets worth, and so on.

There are two exceptional cases before the main calculation. In the first case, there are no relevant tweets whatsoever in any score block, and we take the maximum of the global threshold (GT) and the highest score seen so far plus w_t, the weight assigned to a title term match. In the second case, all tweets are relevant and so we take the minimum score seen so far.

If we have a mix of relevant and non-relevant tweets, we first compute the average gain for each score as follows:

$$avg\_gain(s_i) = \frac{\sum\limits_{s_j \geq s_i} \sum\limits_{t \in SB_j} gain(t)}{\sum\limits_{s_j \geq s_i} |SB_j|}$$

Selecting the score that has the maximum average gain would be one such threshold selection strategy. Although, we have found that such a strategy is not particularly good. We then weight score's average gain by the proportion of relevant material provided by that score (i.e., the precision of that score).

$$weight(s_i) = \frac{|\{t | t \in SB_i \wedge gain(t) > 0\}|}{\sum\limits_{s_j} |SB_j|}$$

$$weighted\_avg\_gain(s_i) = avg\_gain(s_i) * weight(s_i)$$

Selecting a threshold based upon this weighted average gain tends to work reasonably well, but tends to introduce too much non-relevant material and thus dampens ELG. To focus on attaining the highest gain from the most relevant tweets, we use the relevance information to adjust this threshold according to the ratio between relevant and non-relevant material.

$$cand\_score(s_i) = \frac{\sum\limits_{s_j \geq s_i} |\{t | t \in SB_j \wedge gain(t) = 0\}|}{\sum\limits_{s_j \geq s_i} |\{t | t \in SB_j \wedge gain(t) > 0\}|}$$

For use as a threshold, a score's *cand_score* must then be less than some cutoff, σ. We can vary σ to be more or less permissive of non-relevant tweets, with $\sigma = 1.75$ achieving substantially improved results on ELG (see Table 5.4). Note that if no score is able to achieve a *cand_score* greater than σ, we set the threshold for the next day to be the maximum of the global threshold (GT) and the highest score seen so far, across all days.

Strategy	ELG
GT=5 (baseline)	0.3191
GT=6	0.3303(p=0.3819)
FB:avg_gain	0.3257(p=0.5664)
FB:weighted_avg_gain	0.3510(p=0.0004)
FB:weighted_avg_gain +cand_score	0.3678(p=0.0000)

Table 5.4: Dynamic emission strategies (p-values are generated from a paired sign test with GT = 5 (baseline)).

5.4 Results

Table 5.5 and Table 5.6 are showing the results of our TREC 2015 submitting systems. The RunID UWaterlooATDK is the best performed automatic run within all 37 submitted runs. Only title features are used in the run and the pushing strategy is the dynamic time window pushing strategy.

Strategy	RunID	ELG	nCG
Title+Dyn	UWaterlooATDK	0.3150 (0.2366 - 0.3933)	0.2679 (0.1864 - 0.3494)
Title+Fix	UWaterlooATEK	0.2654† (0.1892 - 0.3415)	0.2365† (0.1576 - 0.3154)
Title+Narr+Desc+Fix	UWaterlooATNDEK	0.2470† (0.1796 - 0.3144)	0.2170† (0.1474 - 0.2865)

Table 5.5: Results for Scenario A (push notification) for submitted runs with 95% confidence intervals. † denotes $p < 0.01$ in a paired t-test with run UWaterlooATDK.

However, when we preformed posterior experiments, we determined that the UWaterlooATDK run's performance did not benefit from the dynamic time window pushing strategy. In fact, the

Strategy	RunID	nDCG
Title only	UWaterlooBT	0.2200 (0.1684 - 0.2716)
Title+Narrative+Description	UWaterlooBTND	0.2196 (0.1682 - 0.2710)

Table 5.6: Results for Scenarion B (email digest) for submitted runs with 95% Confidence Intervals.

proportion of pushed tweets that waited for the maximum time window size is only 0.2%. This means that most tweets are pushed immediately. The only reason that this run performed well is the k_1 threshold in this strategy is high, i.e., the score of top 5th tweet from the previous day.

5.5 Summary

Simple formulae for content matching and novelty can provide good performance for microblog filtering, provided that care is taken to set appropriate thresholds to avoid pushing non-relevant information. With simple content matching it is even possible to use a fixed, static global threshold across all queries, although the results fall short of what might be achieved if the optimal threshold could be determined for each topic at the beginning of each day. While dynamic thresholds can be set from the tweets of previous days, without feedback, this strategy provides only very limited gains.

On the other hand, dynamic thresholding through feedback has the potential to produce substantial and significant gains. While we have explored only one approach to this idea, through daily relevance judgments, we can imagine more realistic interfaces, which might for example, allow up/down judgments as tweets are pushed. However, to a large extent, our ability to explore further is limited by the assumptions and data of the Microblog Track. Announced plans for TREC 2016 include some level of ongoing judgments, which could provide an vehicle to test other approaches. In addition, we hope to incorporate social signals and other non-content features into the relevance and novelty components of our system, with the goal of retaining our simple approach to thresholding, while improving overall performance.

Chapter 6

Evaluation of Real-time Tracking in Social Media

This chapter explores the problem of evaluating push notification techniques on social media streams in a filtering application. As described in last chapter, we assume a stream of social media posts such as Twitter, against which the user issues an arbitrary number of standing queries representing "interest profiles", analogous to topics in ad hoc retrieval. For example, the user might be interested in poll results for the 2016 U.S. presidential elections and wishes to be notified whenever new results are published. The system's task is to identify relevant tweets from the stream and send these updates directly to the user's mobile phone via a push notification. Since such notifications are often associated with an auditory or visual cue upon arrival, each imposes a non-trivial cognitive burden on the user. Thus, careful control of the volume of notifications is critical to successful push strategies. This chapter explores evaluation metrics for such a task.

At a high level, push notifications should be relevant, timely (i.e., providing updates as soon after the actual event occurrence as possible), and novel (i.e., users should not be pushed multiple notifications that say the same thing). Accordingly, an evaluation metric should reward systems for updates that satisfy these three main criteria. In this chapter, we focus on relevance and timeliness, adopting existing notions of novelty.

We start with an analysis of metrics from the TREC 2015 Microblog track, which operationalizes such a push notification task, and then re-assess submitted runs after making a minor tweak to reflect a different assumption about the user model. We then re-assess the submitted runs using variants of a metric that has previously been applied in the same evaluations. Using score and rank correlations, we compare system effectiveness as measured by each metric. Our results are surprising: We find little correlation between the different metrics. This means that the answer to "which system is better" depends on how you measure it, which is undesirable from an evaluation perspective.

We then examine the effects of latency penalties by considering the impact of different metric variants on runs submitted to TREC 2015. This evaluation forms the basis of the Real-Time Summarization (RTS) track at TREC 2016. This work provides a basis on which the evaluation metrics for the RTS track at TREC 2016 can be developed.

The contribution of this chapter is three-fold: First, we present the novel and surprising finding discussed above: any number of reasonable evaluation metrics give rise to significantly different system rankings. We discuss and analyze why, tracing the issue to the handling of days for which there are no relevant tweets. Second, we argue that the different existing evaluation metrics we applied can be generalized into a framework that uses the same underlying contingency table, but places different weights and penalties. Although we do not propose "one true metric", we believe this framework can guide the future development of an evaluation metric that more accurately models user needs. Third, we know systems should be punished by pushing tweets late. Without an empirical user study of how real users respond to push notifications, we cannot develop a good latency penalty metric.

6.1 TREC 2015 Microblog track Metrics

The application scenario described in Section 5.1 was operationalized in the TREC 2015 Microblog track as the so-called "Scenario A" variant of the real-time filtering task. Over the official evaluation period, which spanned ten days during July 2015, participating systems "listened" to Twitter's live tweet sample stream to identify relevant tweets with respect to 225 topics, 51 of which were later assessed. Each system identified up to ten tweets per day, which were putatively delivered to a hypothetical user. In total, 14 groups submitted 37 runs to the evaluation. Data from this evaluation provides the starting point for our analysis.

The assessment workflow for the track was as follows: first, tweets returned by the systems were assessed for relevance using a traditional pooling process. Relevant documents were then semantically clustered into groups containing tweets sharing substantively similar information. We refer the reader to prior work for more details [155].

The two metrics used to evaluate system runs were expected latency-discounted gain (ELG) and normalized cumulative gain (nCG). These two metrics are computed for each topic for each day in the evaluation period (explained in detail below). The score of the topic is the average of the daily scores in the evaluation period. The score of a run is the average of the scores across all topics.

Expected latency-discounted gain (ELG) was adapted from the TREC Temporal Summarization track [11]:

$$\frac{1}{N}\sum \mathrm{G}(t) \tag{6.1}$$

where N is the number of tweets returned and $\mathrm{G}(t)$ is the gain of each tweet: not relevant tweets receive a gain of 0, relevant tweets receive a gain of 0.5, highly-relevant tweets receive a gain of 1.0.

A key aspect of this metric is its handling of redundancy and timeliness: A system only receives credit for returning one tweet from each cluster. Furthermore, a latency penalty is applied to all tweets, computed as $\mathrm{MAX}(0, (100 - d)/100)$, where the delay d is the time elapsed (in minutes, rounded down) between the tweet creation time and the putative time the tweet was delivered. That is, if the system delivers a relevant tweet within a minute of the tweet being posted, the system receives full credit. Otherwise, credit decays linearly such that after 100 minutes, the system receives no credit even if the tweet was relevant.

The second metric is normalized cumulative gain (nCG):

$$\frac{1}{\mathcal{Z}}\sum \mathrm{G}(t) \tag{6.2}$$

where $\mathcal{Z}$ is the maximum possible gain (given the 10 tweet per day limit). The gain of each individual tweet is computed as above (with the latency penalty). Note that gain is not discounted (as in nDCG) because the notion of document ranks is not meaningful in this context.

Due to the setup of the task and the nature of interest profiles, it is possible (and indeed observed empirically) that for some days, no relevant tweets appear in the judgment pool. In terms of evaluation metrics, a system should be rewarded for correctly identifying these cases and not pushing non-relevant content. If there are *no* relevant tweets for a particular day and the system returns zero tweets, it receives a score of one (i.e., perfect score) for that day; otherwise, the system receives a score of zero for that day. Note that a day may be silent for a system if it has previously pushed tweets in all clusters appearing on that day.

It is worth mentioning that despite superficial similarities, our task is very different from document filtering in the context of topic detection and tracking (TDT) [4]. TDT is concerned with identifying *all* documents related to a particular event—with an intelligence analyst in mind—which requires keeping track of false alarms and missed detections. In contrast, we are focused on identifying a small set of the most relevant updates to push to users, grounded in interactions with mobile devices. Furthermore, in TDT, systems must make online decisions as soon as documents arrive, whereas in our case systems can choose to push older content (subjected to the latency penalty), thus giving rise to the possibility of algorithms operating on bounded buffers. For these various reasons, TDT evaluation tools such as the decision error tradeoff (DET) curve and derivative metrics provide inspiration, but are not directly applicable.

6.2 Analysis of "Silent Days"

In Figure 6.1, we show a scatterplot of the official ELG scores (which we call ELG-1 for reasons that will become clear shortly) vs. nCG. Although there is an overall correlation between ELG-1 and nCG across all submitted runs, we do note that in particular cases ELG-1 and nCG are capturing different aspects of effectiveness: for example, the top three runs (circled in blue) in terms of ELG-1 exhibit relatively large differences in nCG. There are also cases in which systems achieve high nCG relative to their ELG-1 scores (the runs circled in red in the figure).

One interesting aspect of ELG-1 is its handling of days in which there are no relevant documents: for rhetorical convenience, we call days in which there are no relevant tweets for a particular topic (in the pool) "silent days", in contrast with "eventful days" (where there are relevant tweets). In the ELG-1 metric, for a "silent day", the only two possible system scores are one (if the system remained silent) or zero (if the system pushed any tweets). This means that an empty run (a system that never returns anything) may have a non-zero score based on how many

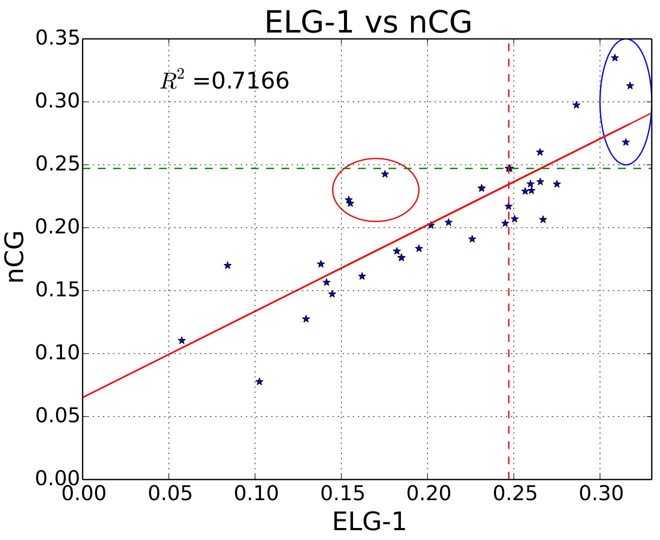


Figure 6.1: ELG-1 (official metric) vs. nCG.

silent days there are in each topic. As it turns out, an empty run will score 0.2471 in ELG-1 and nCG, shown as dotted lines in Figure 6.1. Since this was the first year of the TREC evaluation, systems achieved high scores by simply returning few results, in many cases for totally idiosyncratic reasons—for example, the misconfiguration of a score threshold.

As an alternative, what if we did not reward systems for remaining silent? That is, on a silent day, all systems receive a zero score, no matter what they did. We call this variant metric ELG-0 (in contrast to ELG-1, the official metric). We can justify this from the user perspective in that for a silent day, the user does not obtain any relevant information regardless of system output (since there are no relevant documents). In this case, how would the user know to "reward" a system for remaining silent? That is, properly determining a silent day requires global knowledge (e.g., from pooling), which no individual user has access to.

In Figure 6.2, we show a scatterplot of ELG-1 vs. ELG-0. We see no discernible relationship between the two metrics, which suggests that the handling of silent days *is the most critical part of the metric*, in that different (reasonable) formulations yield dramatically different results and system rankings. In fact, we would go as far as saying that effectiveness under ELG-1 is primarily dominated by a system's ability to identify the silent days. Under ELG-1, systems do

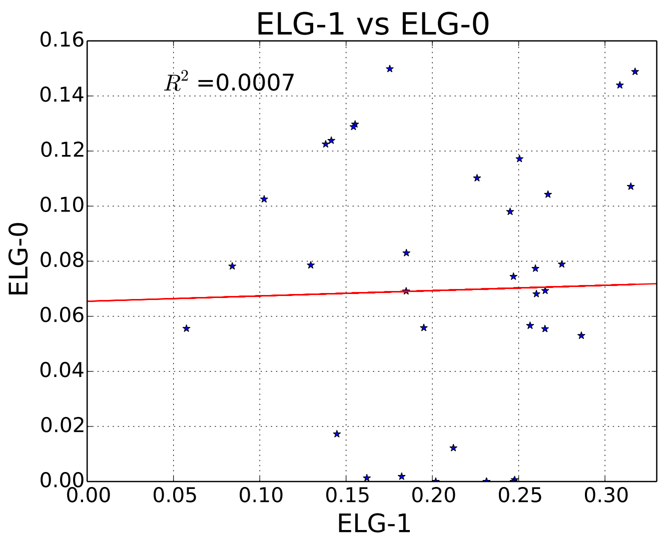


Figure 6.2: ELG-1 vs. ELG-0 for all submitted runs.

well by learning when to "shut up".

This observation is further illustrated by the scatterplots in Figure 6.3, Figure 6.4 and Figure 6.5, where we show silence precision vs. ELG-1, silence recall vs. ELG-1, and silence precision vs. silence recall for all runs.

Silence precision and recall follow the usual definitions of precision and recall, but with respect to identifying the silent days. Since each topic has an equal number of days, there is no distinction between micro- and macro-averaging. Across all the topics, 24.7% of all days are completely silent, while another 6.7% have relevant but redundant material. We see that there is a slightly positive (but very weak) correlation between ELG-1 and silence precision. Figure 6.4, in effect, shows that systems achieve a high ELG-1 by achieving a high silence recall—i.e., getting a good score is dominated by a system knowing when to "shut up". Although systems with comparable silence recall can differ substantially in ELG-1, we were surprised at how much of the variance in the official metric can be explained by silence recall alone.

Figure 6.5 shows the tradeoffs system make with respect to precision and recall. On the right edge of the plot are cases where the systems are almost always quiet, achieving nearly perfect recall; on the left lower corner is a system that never "shuts up", and hence its precision and recall

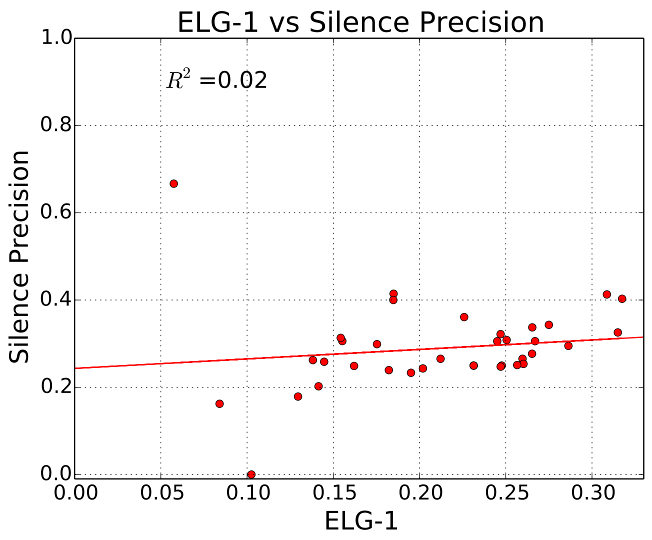


Figure 6.3: Characterizing the effects of “silent days”: Silence precision vs. ELG-1

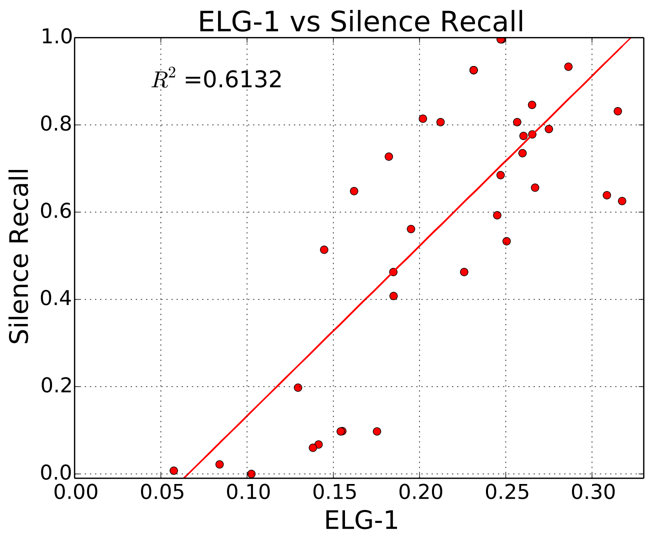


Figure 6.4: Characterizing the effects of “silent days”: Silence recall vs. ELG-1

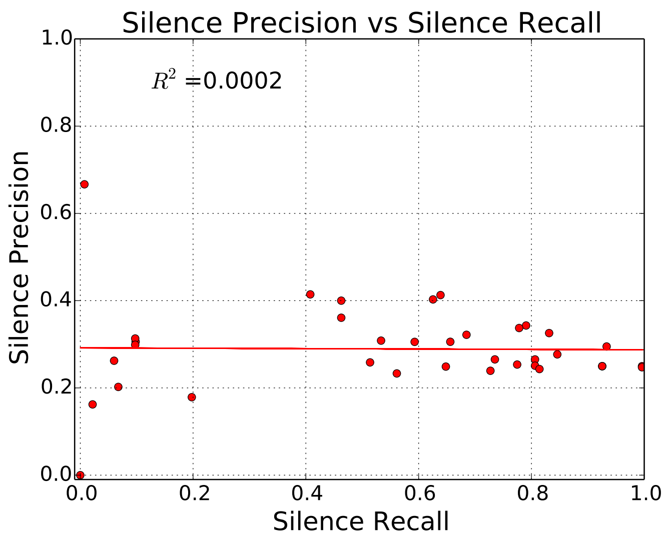


Figure 6.5: Characterizing the effects of "silent days": Silence precision vs. silence recall

are both zero. It is interesting to note that some systems perform poorly in both precision *and* recall—they don't push content when there's relevant content and don't "shut up" when there's no relevant content.

To our knowledge, this is the first time the huge impact of silent vs. eventful days has been understood in the evaluation of push notification. However, we withhold judgment as to whether the current TREC metrics represent the "right" approach: from the user perspective, since push notifications are associated with high cognitive effort, perhaps the metric is correct in forcing systems to focus on learning when to "shut up". On the other hand, having such highly binarized scores on the silent days creates many issues for system tuning, since it creates discontinuities in the objective—the same issue as with trying to optimize a metric such as precision at rank one in a retrieval task.

6.3 Gain and Pain

What are other reasonable ways in which we can evaluate the push notification task? A simple and intuitive utility-based metric would be to reward "gain" based on delivery of relevant information and to deduct "pain" based on delivery of non-relevant information. In fact, the TREC 2012 Microblog track employed exactly such a metric, called T11U [140], itself derived from the linear utility metrics used in the TREC filtering tracks.

We adopt a slightly different but mathematically equivalent formulation as follows:

$$\text{T11U} = \alpha \cdot \mathcal{G} - (1 - \alpha) \cdot N_x \tag{6.3}$$

where $\mathcal{G}$ is total gain, N_x is the number of non-relevant documents pushed, and α controls the relative weight of gain vs. pain. Note that in T11U, the total gain factors in different relevance grades, the latency penalty, and the treatment redundant tweets in exactly the same way as ELG.

In Figure 6.6, we show a scatterplot of ELG-1 vs. T11U with $\alpha = 0.66$, which was the value used in TREC 2012. This value can be understood as setting the gain of a relevant notification (highest relevance grade, no temporal penalty, not redundant) equal to the pain of returning two non-relevant updates. With this setting, we do see reasonable positive correlation between ELG-1 and T11U overall, but this correlation is highly misleading. According to T11U, very few systems achieve positive utility overall—that is, the gain from pushing relevant content is not sufficient to offset the pain from pushing non-relevant content. Furthermore, the relatively large cluster of runs which score around zero in T11U vary widely in ELG, from around 0.2 to over 0.3, with the highest T11U score being somewhere in the middle of this band. In other words, poorly-performing systems score low in both ELG and T11U, but beyond that, T11U and ELG exhibit a weak correlation at best.

Of course, absolute scores and relative system rankings depend on the α parameter that balances gain vs. pain, and the setting of $\alpha = 0.66$ was arbitrary. What would the evaluation results look like for different settings of α? The answer is shown in Figure 6.7, where we sweep the α parameter and compute Kendall's τ with respect to ELG-1 for each setting. The results show that Kendall's τ varies substantially, ranging from moderate correlation to non-existent and even slightly negative correlation. We have shown above that high correlations can be misleading, and this plot shows that $\alpha = 0.66$ is around the highest Kendall's τ we can obtain regardless. These results suggest that T11U and ELG-1 are measuring quite different aspects of effectiveness.

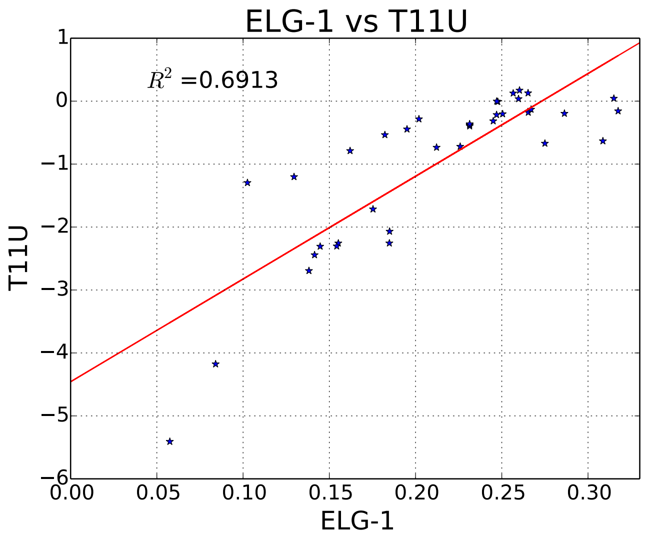


Figure 6.6: T11U vs. ELG-1.

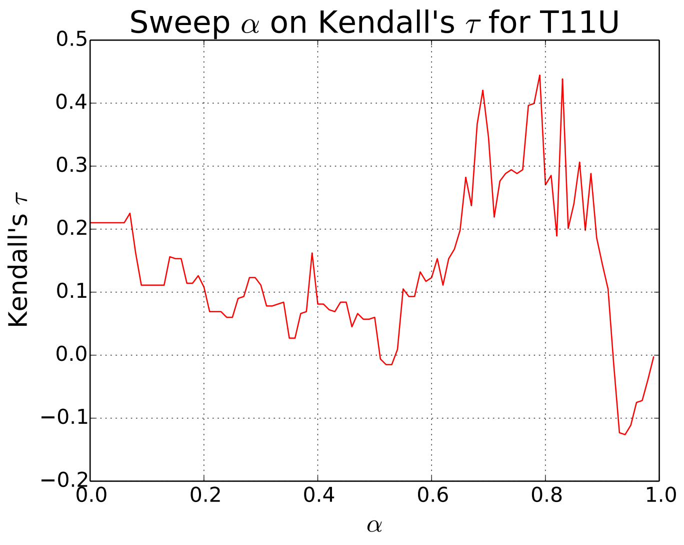


Figure 6.7: Kendall’s τ between T11U and ELG-1 as a function of α.

6.4 Effects of the Delay Penalty

The simplest way to quantify the impact of the latency penalty is to remove it altogether. This analysis is shown in Figure 6.8, where for ELG-1 and ELG-0, we plot the score of each run with and without the latency penalty. In both scatterplots we show the diagonal $y = x$ (note, *not* the best fit line) for reference.

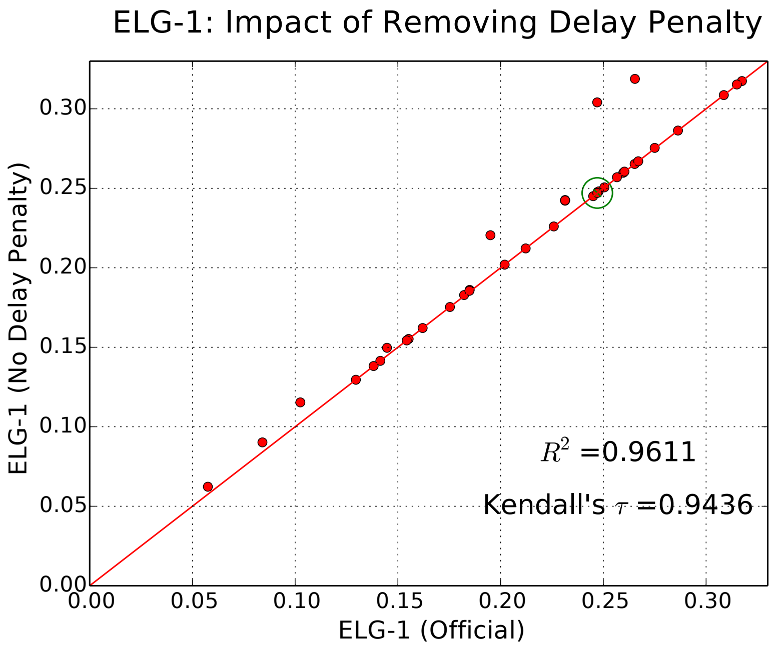
In this and all scatterplots, R^2 values report the results of linear regressions, and rank correlations are shown in terms of Kendall's τ. As expected, all points lie above the diagonal, since without the latency penalty system scores increase. We were surprised, however, that the scores of many systems were exactly the same, which meant that their algorithms made immediate decisions with respect to each incoming tweet. In fact, there were only a few outlier systems whose scores substantially changed, which meant that they pushed tweets posted in the past.

We are quick to caution, however, that past system behavior is not necessarily a good indication of system behavior in future evaluations. In particular, TREC 2015 represented the first evaluation of push notifications, and it is entirely possible that participants focused on simple algorithms that did not attempt to model the tradeoffs involved in pushing past tweets (i.e., accepting the latency penalty for perhaps better relevance scoring).

Another noteworthy aspect of ELG (both the ELG-1 and ELG-0 variants) is that the latency penalty is computed with respect to the pushed tweet, as opposed to the first tweet in the cluster. Recall that in the evaluation protocol, tweets are semantically clustered into "equivalence sets" that contain substantively the same information.

Let's consider the case where tweets A and B belong to the same cluster, but tweet B was posted three hours after tweet A. Suppose system P pushed tweet A two hours after it was posted and system Q pushed tweet B immediately when it was posted. Under the official scoring metric, system P would receive no credit whereas system Q would receive full credit; this doesn't make sense since system P conveyed the relevant information to the user before system Q did.

Recognizing this issue, it seems appropriate to compute the latency penalty with respect to the first tweet in each cluster (which is essentially what the Temporal Summarization track does). The effect of this change on system scores is shown in Figure 6.9, where the scatterplots show each run under the official score definition and the alternate computation of the latency penalty



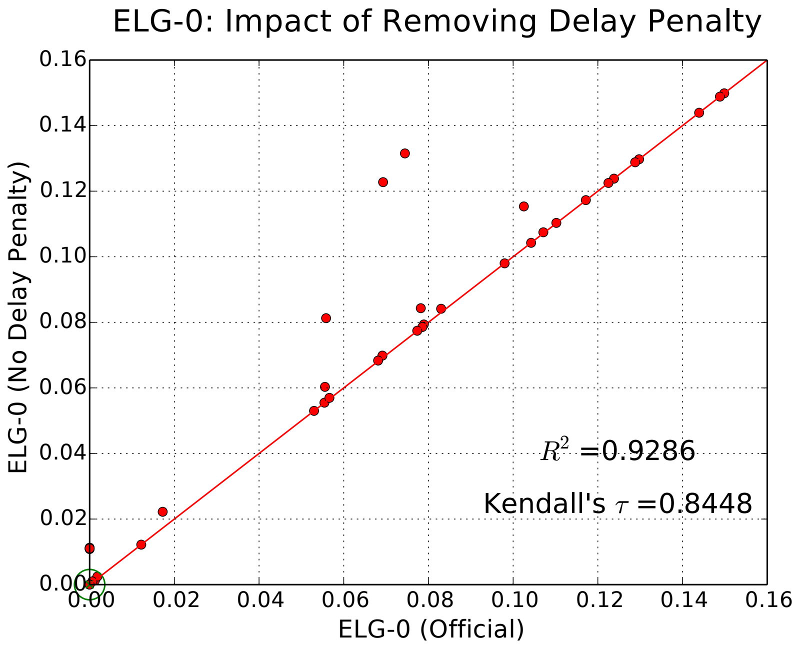


Figure 6.8: ELG-1(top) and ELG-0(bottom) of all runs submitted to TREC 2015, comparing the official latency penalty definition with removing the latency penalty altogether. Green circles indicate empty runs.

with respect to the first tweet in each cluster. In both scatterplots we show the diagonal $y = x$ for reference. As expected, all points lie below the diagonal since scores decrease, but system rankings don't change much.

From another perspective, in Figure 6.10 we show the mean (bars) and median (diamonds) delay in pushing tweets by each system, according to the official metric, and with respect to the first tweet in each cluster in Figure 6.11. In these plots, we only consider tweets that actually contributed to a run's score (i.e., yielded non-zero gain). Note that the y axis is on a logarithmic scale in minutes. The bars are arranged in descending ELG-1 score, from left to right.

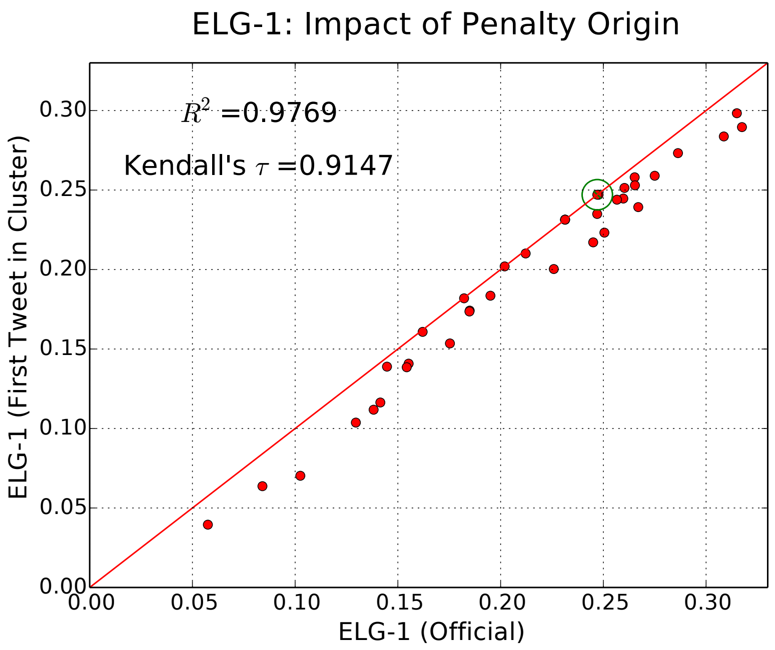
From the bar chart in Figure 6.10, we see that, indeed, most systems always push immediately when a tweet is posted (if the system thinks the tweet is relevant). We also see a few teams that pushed tweets with a large delay—however, these are systems that pushed very few results, and so their ELG-1 scores are fairly close to that of the empty run.

One salient feature of the participating systems is that they vary quite a bit in the volume of relevant tweets that they push. Because of the reward associated with "staying quiet", systems can achieve similar ELG scores with very different push volumes. This is shown in Figure 6.12, where each bar shows the total number of relevant tweets that are pushed by each system. The bars are arranged in decreasing ELG-1 score from left to right. The red portions of the bars represent tweets that contribute non-zero gain, while the tan portions of the bars represent tweets that did not contribute any gain. These are either redundant tweets or tweets pushed beyond the maximum acceptable latency (100 minutes) to receive any credit.

We see that there are many cases where systems that pushed more relevant tweets actually score lower than systems that pushed fewer relevant tweets. Many of these are systems that always push tweets no matter what—in other words, they don't know when to "shut up". In the range of middle-scoring runs, we see a number of systems that barely push any content, and so their ELG-1 scores are very close to that of the empty run (which, recall, was a baseline that actually beats most systems).

This effect is highlighted in Figure 6.13, where the runs are resorted in terms of ELG-0 (but otherwise the bars are exactly the same). Under this metric, systems are not rewarded for staying quiet, and therefore systems that push more relevant tweets tend to score higher.

As a final analysis, in Table 6.1 we tally the number of clusters for each topic, the number



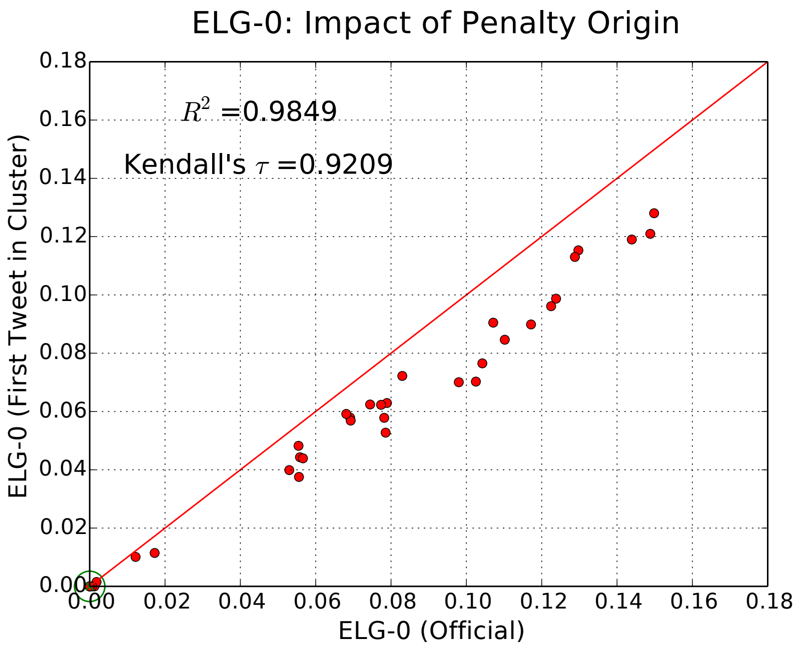


Figure 6.9: ELG-1 (top) and ELG-0 (bottom) of all runs submitted to TREC 2015, comparing the official latency penalty definition with computing the latency penalty with respect to the first tweet in each cluster. Green circles indicate empty runs.

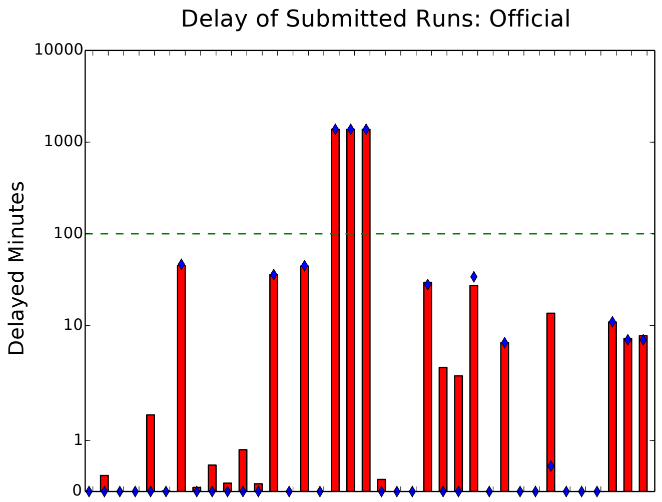


Figure 6.10: Quantifying the delay of each run in pushing tweets, with respect to the posted tweet. Runs are sorted in descending order of ELG-1.

of singleton clusters (with only a single relevant tweet), and singleton clusters expressed as a percentage of all clusters. We see that most of the clusters are singletons, which helps explain the results observed in Figure 6.10: for singleton clusters, the latency penalty is always computed with respect to the same tweet.

Profile	Clusters	Singletons	%
MB228	3	3	100%
MB236	97	40	41%
MB242	73	47	64%
MB243	119	70	59%
MB246	171	127	74%
MB253	3	3	100%
MB254	32	27	84%
MB255	15	12	80%
MB260	1	0	0%
MB262	67	48	72%
MB265	58	44	76%

MB267	39	29	74%
MB278	35	30	86%
MB284	51	42	82%
MB287	71	58	82%
MB298	8	7	88%
MB305	3	2	67%
MB324	3	3	100%
MB326	20	10	50%
MB331	25	8	32%
MB339	10	9	90%
MB344	818	594	73%
MB348	37	22	59%
MB353	10	9	90%
MB354	18	3	17%
MB357	7	7	100%
MB359	8	7	88%
MB362	55	49	89%
MB366	97	79	81%
MB371	103	64	62%
MB377	12	9	75%
MB379	28	15	54%
MB383	16	13	81%
MB384	14	11	79%
MB389	17	14	82%
MB391	75	61	81%
MB392	17	6	35%
MB400	87	81	93%
MB401	311	236	76%
MB405	5	5	100%
MB409	9	5	56%
MB416	8	6	75%

MB419	56	36	64%
MB432	44	41	93%
MB434	276	257	93%
MB439	67	63	94%
MB448	1	0	0%
Average	66	49	74%

Table 6.1: The total number of clusters and singleton clusters for each interest profile.

6.5 Toward A General Framework

Let us take stock of our findings so far: we have evaluated runs from the TREC 2015 Microblog track using official as well as alternative metrics (nCG, ELG-1, ELG-0, and T11U). In comparing the metrics, we observe many inconsistencies in terms of both system scores and relative rankings. In other words, "which system is better" depends on what measure we use. From an evaluation perspective, this is not desirable because researchers lack consistent guidance for algorithm development.

To address this issue, we propose a general evaluation framework built around the contingency table shown in Table 6.2. At the core, our framework is utility-based in that gain is rewarded for pushing relevant content ($+G_E$) and pain is deducted for pushing non-relevant content ($-P_E$ and $-P_0$). However, a key insight is the explicit separation of eventful and silent days, which our ELG-1 vs. ELG-0 experiments have shown to be critical in system evaluations.

System action	"eventful days"	"silent days"
Pushed relevant	$+G_E$	-
Pushed not-relevant	$-P_E$	$-P_0$
Stayed silent	$-S_E$	$+S_0$

Table 6.2: The contingency table for a general evaluation framework for push notifications.

Our evaluation framework is general in that the metrics we have examined in this chapter can be viewed as specific instantiations of the parameters in Table 6.2. For example, T11U sets G_E

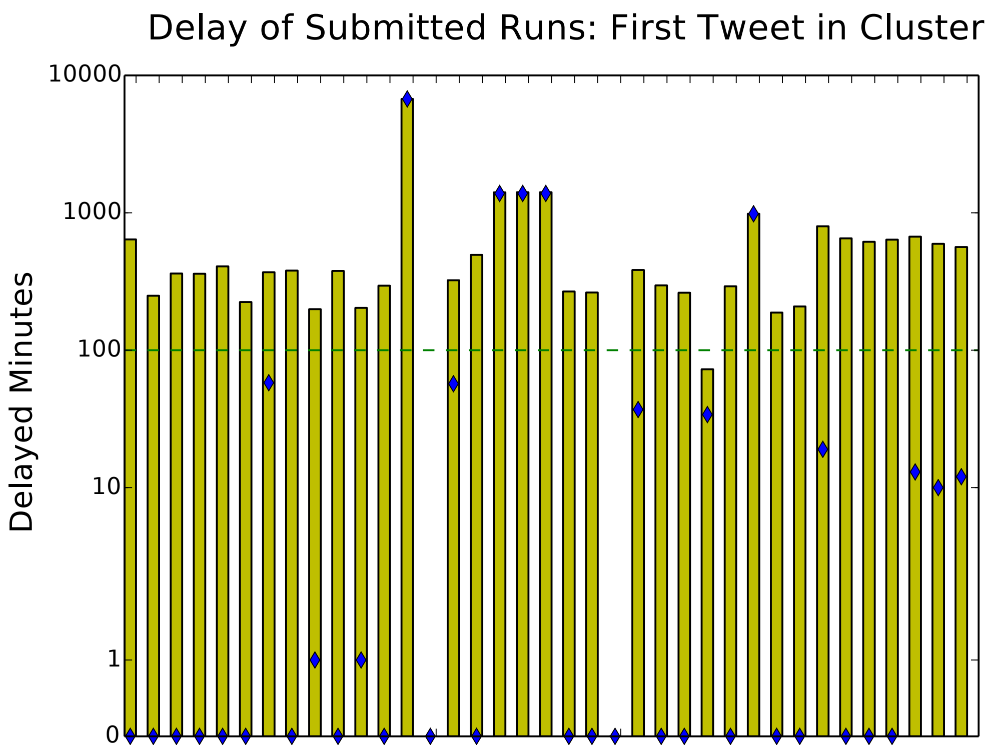


Figure 6.11: Quantifying the delay of each run in pushing tweets, with respect to the first tweet in each cluster. Runs are sorted in descending order of ELG-1.

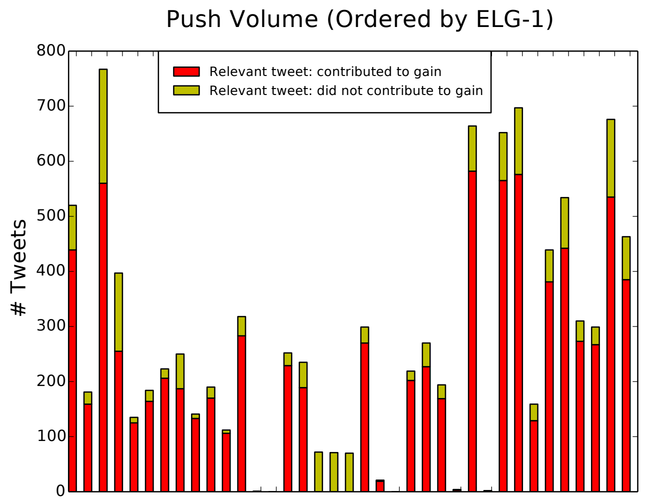


Figure 6.12: The push volume of each system, showing the number of relevant tweets pushed and the fraction that contributed to gain, in descending order of ELG-1.

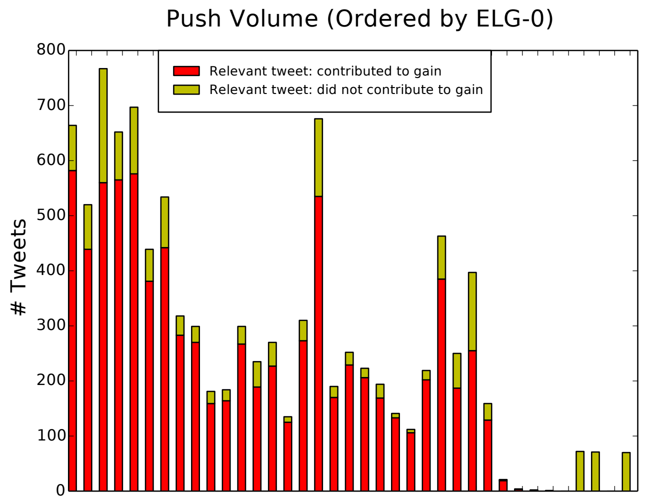


Figure 6.13: The push volume of each system, showing the number of relevant tweets pushed and the fraction that contributed to gain, in descending order of ELG-0.

and P_E (based on α) but ignores the final row, and furthermore does not make the distinction between eventful and silent days. ELG-1 and ELG-0 make different choices on S_0, how systems should be rewarded for staying silent on silent days, but both set P_E and P_0 to zero. That is, no pain is deducted for pushing non-relevant content.

Using the framework presented in Table 6.2 as a guide, we can imagine a family of metrics beyond those already presented. For example, we might augment T11U by creating a distinction between eventful and silent days, thus arriving at a metric that is closer to ELG-1 or ELG-0. We might set P_E differently from P_0 to create more nuanced distinctions in a T11U-like metric. Different ratios between these weights also give rise to emphasis on different aspects of the push notification problem.

The question remains on how to properly set the gain and pain weights in the contingency table—and we presently provide no concrete answer, expect to say that further studies in user modeling are necessary. For example, we have presented two plausible scenario (ELG-1 and ELG-0) on the treatment of silent systems on silent days: a user study is necessary to decide which alternative (or neither) matches user preferences. Our framework contributes to a step toward the goal of the development of the "one true metric".

Push notifications should be relevant, novel, and timely. The focus of this work is the last property. Intuitively, systems should be "punished" for returning tweets late, hence the latency penalty implemented in ELG. There is, however, little empirical characterization of how real users would respond to push notifications with increasing delay. Ultimately, user studies are needed to ensure that metric definition and user needs actually align.

Chapter 7

Ranking Similarity Measurement

Even minor modifications to ranking algorithms, training data, or document collections may cause unpredictable changes to the quality of search engine results. For any given query, documents may move up and down the ranked list. New documents may appear; others may disappear.

Traditionally, changes in search engine results are quantified by way of retrieval effectiveness measures, such as normalized discounted cumulative gain (nDCG), average precision (AP), and expected reciprocal rank (ERR) [30, 66]. These measures are computed over ranked lists before and after a modification. The difference in their values — typically averaged over a number of queries — indicates the magnitude of the change.

Unfortunately, the computation of these measures depends on the existence of explicit relevance judgments. For a given query, we must know whether or not each ranked document is relevant to the query, and for some measures we must also know the degree to which each document is relevant to the query (e.g., definitive, excellent, good, fair, bad, or detrimental [120]). These judgments may be created by direct human assessment of relevance or inferred indirectly from clickthrough logs and other user interaction data [31, 45, 49, 64, 71, 139]. Creation of these judgments involves either substantial effort on the part of assessors or large volumes of interaction data, limiting the number of queries over which the measures may be computed. Moreover, modifications may surface previously unseen and unjudged documents. While we might assess these documents immediately, requiring additional assessment, or treat them as non-relevant, we run the risk of compromising the accuracy of the effectiveness measures.

Alternatively, changes in search results may be quantified by way of rank correlation coefficients and other rank similarity measures, avoiding the need for explicit relevance judgments. For example, the providers of a search service might use such measures to estimate the impact of a proposed change across large numbers of queries — perhaps millions of them — identifying those where the potential impact is greatest. While standard rank correlation coefficients — such as Spearman's ρ or Kendall's τ — might be applied for this purpose, several researchers have proposed specialized rank similarity measures that better reflect the requirements of search.

Some of this prior work focuses primarily on the problem of comparing search result lists [21, 82, 109, 143], a problem which we directly address in this chapter. Most notably, Webber et al. [157] carefully analyze the requirements for comparing ranked lists across a wide range of applications, including search results. Other work focuses primarily on the distinct, but related, problem of comparing system rankings under different effectiveness measures [27, 166], a problem which we do not directly address in this chapter. We note, however, that progress on one problem may translate into progress on the other.

Webber et al. identify three key characteristics that particularly apply to comparisons between search results. First, users tend to focus on the top-ranked results, and more rarely view results deep in the list [65, 71]. Thus, a rank similarity measure for search results must be *top-weighted*, placing greater emphasis on early results, and lesser emphasis on later results. A change in the third result should have greater impact than a change in the 103rd. Second, for a given query, search engines do not typically rank more than a tiny fraction of the collection. Thus, a similarity measure for search results must handle *incomplete* rankings, where a document appearing in one list may not appear in the other.

Finally, since users may stop at any point in a ranked list, either because their information need is satisfied or because their patience is exhausted, a rank similarity measure for search results must appropriately handle *indefinite* rankings. According to Webber et al. the measure "should not arbitrarily assign a cutoff depth, but be consistent for whatever depth is available." Suppose the ranked lists are truncated at an arbitrary depth K and we compute the measure at that depth. The value of the measure at K should allow us to make predictions about the value of the measure if it were computed at depths greater than K, allowing us to compute bounds on the amount it can change as the depth of computation is increased. For example, perhaps the measure would only increase, or stay the same, as the depth increased (i.e., adding more documents could not make the lists less similar).

To satisfy the requirements implied by these three characteristics, Webber et al. proposed a new rank similarity measure, which they call *rank biased overlap* (RBO). Given two ranked lists, A and B, let $A_{1:k}$ denote the top k documents in A, and let $B_{1:k}$ denote the top k documents in B. Define the *overlap* between A and B at depth k as the size of the intersection between these lists at depth k (i.e., $|A_{1:k} \cap B_{1:k}|$) and define the *agreement* between A and B at depth k as the overlap divided by the depth. Webber et al. define RBO as a weighted average of agreement across depths, where the weights decay geometrically with depth, reflecting the requirement for top weighting:

$$\text{RBO} = (1-\psi)\sum_{k=1}^{\infty}\psi^{k-1}\frac{|A_{1:k} \cap B_{1:k}|}{k}. \tag{7.1}$$

In this equation, the parameter $0 < \psi < 1$ represents user persistence, with larger values representing a more patient user. When computing RBO for the comparative experiments reported in this chapter, we set $\psi = 0.9$, a typical choice in Webber et al. The normalization factor $(1-\psi)$ serves to map the value of RBO into the range $[0 : 1]$. In practice, RBO is computed down to some fixed depth K, which replaces ∞ in the equation, reflecting the indefinite and incomplete nature of the ranked lists.

Webber et al. surveyed a large number of rank similarity measures, and argued that only RBO fully satisfies the requirements of ranked search results. In creating RBO, Webber et al. drew inspiration from the user model incorporated into the rank-bias precision (RBP) effectiveness measure proposed by Moffat and Zobel [118]. This model imagines a user scanning a ranked list, starting at the top. After scanning a result, the user proceeds to the next result with probability ψ and stops with probability $1-\psi$. This simple user model provides the basis for the weights appearing in both RBP and RBO.

This close connection with the RBP effectiveness measure helps to justify the application of RBO to search results. In general, given the enormous effort made by the search community to develop and validate effectiveness measures, it seems reasonable to exploit this existing work to guide the creation of rank similarity measures targeted at search results. Taking this observation a step further, we propose a family of distance measures, each directly derivable from an associated effectiveness measure. The core idea is straightforward:

Given two ranked lists, A and B, what is the maximum difference in their effectiveness scores possible under a specified effectiveness measure?

We call this family of distance measures *Maximized Effectiveness Difference* (MED). Since we typically normalize MED into the range $[0 : 1]$, the value $1 - MED$ provides an corresponding family of rank similarity measures.

Like RBO, MED is a metric in the strict mathematical sense of defining a distance between two ranked lists (see Section 7.1). The value of this distance varies with the associated effectiveness measure. Unlike most similarity measures, MED is not a dimensionless quantity, since its value can be interpreted in terms of the associated effectiveness measure.

Like RBO, MED transfers understanding and assumptions about user behavior from an existing effectiveness measure to measure rank similarity. Unlike RBO, MED provides a method for transforming any effectiveness measure into a rank similarity measure. Both MED and RBO derive their approach to top weighting from their associated effectiveness measure, along with an ability to handle incomplete rankings, satisfying the requirements of search. MED is monotonically decreasing with increasing depth of evaluation, satisfying the requirements for indefinite rankings. In addition, MED can take appropriate advantage of known relevance information (see Section 7.1). By the definition of MED, the introduction of known relevance information always decreases the MED distance between two ranked lists. Moreover, if complete relevance information is known, MED reduces to a difference in effectiveness scores.

For an effectiveness measure normalized to the range $[0 : 1]$, the value of MED will also fall into the range $[0 : 1]$. As we discuss later in the chapter, for some effectiveness measures this normalization requires knowledge of collection parameters, e.g., the total number of relevant documents. Since such information cannot be known without complete relevance information, we choose values for these parameters that normalize MED into the range $[0 : 1]$.

To compute MED, we must determine an assignment of relevance values to the documents appearing in the two lists that maximizes the difference in their scores under the specified effectiveness measure, an optimization process that will vary with the effectiveness measure. For some measures, such as RBP, the computation of MED is straightforward (see Section 7.2). For other measures, including AP and ERR, the computation of MED requires the solution of more difficult optimization problems (see Sections 7.3 and 7.4). Either way, MED reflects meaningful differences between ranked lists (see Section 7.5) and provides properties not found in other rank similarity measures on search results (see Section 7.6). Finally, we consider how to extend MED beyond ranked lists, to more other effectiveness measures (see Section 7.7).

7.1 Basic Notation and Properties

We compute MED by maximizing the effectiveness difference between two ranked lists, A and B. If $S(A)$ is the score for list A under some effectiveness measure, and $S(B)$ is the score of list B under the same measure, then we seek to assign relevance values to the documents in A and B to maximize

$$\text{MED}(A, B) = |S(A) - S(B)|\,. \tag{7.2}$$

In the remainder of the chapter, we assume that scores produced by effectiveness measures are always non-negative (and we are not aware of any established effectiveness measure that can produce negative values). Thus, without loss of generality, we present algorithms that maximize

$$S(A) - S(B). \tag{7.3}$$

The overall maximum can be computed by swapping the lists and reapplying the algorithm.

For optimization purposes, we represent A as a vector of variables

$$A = \langle a_1, a_2, ..., a_K \rangle, \tag{7.4}$$

where a_i,$1 \leq i \leq K$, represents the relevance value assigned to the document at rank i in list A. Similarly, we represent B as a vector of variables

$$B = \langle b_1, b_2, ..., b_K \rangle, \tag{7.5}$$

where b_j, $1 \leq j \leq K$, represents the relevance value assigned to the document at rank j in list B. Since a document may appear in both lists, we also have a set of constraints in the form

$$a_n \equiv b_m, \tag{7.6}$$

indicating that the same document appears at rank n in list A and at rank m in list B, so that the same relevance value must be assigned to both variables. Since a document will appear at most once in each list, a given variable will appear in at most one constraint.

We refer to a variable that appears in a constraint as a *bound variable*. If a variable does not appear in any constraint — corresponding to a document that appears in only one list — we refer to it as a *free variable*. In the case that relevance information for a document is known from existing judgments, we assign this known value to its corresponding variable(s), which remain unchanged during maximization. We refer to these variables as *predetermined variables*.

For all effectiveness measures we consider in this chapter, a relevance value is a number in the range $[0 : 1]$. For some measures — such as average precision — relevance is a binary property of a document, and relevance values are either 0 or 1. For graded relevance measures — such as nDCG and ERR — relevance can be one of several values from 0 up to a maximum grade r_G (i.e., the possible relevance grades are $0 = r_0 < r_1 < r_2 < ... < r_G \leq 1$). These grades indicate the level to which a document is judged relevant to the query (e.g., definitive, excellent, good, fair, etc.). For simplicity, we treat binary relevance as a special case of graded relevance, with two grades: $r_0 = 0$ and $r_G = r_1 = 1$. To aid understanding, a relevance value may be interpreted as the probability that a user viewing the corresponding document will consider it to be relevant [30, 34, 37], but this interpretation is not explicitly required in this chapter.

Mathematically, MED is a metric, regardless of the associated effectiveness measure. Non-negativity, identity and symmetry are straightforward. To demonstrate the triangle inequality, consider three ranked lists A, B, and C. Let A' and B' represent the assignment of relevance values that maximizes the effectiveness difference between A and B, i.e., MED$(A, B) = |S(A') - S(B')|$. Let C' be any assignment of relevance values to C that is consistent with A' and B', such that documents are assigned the same relevance values in all three lists. Let A'' and C'' represent the assignment of relevance values that maximizes the effectiveness difference between A and C. Let B''' and C''' represent the assignment of relevance values that maximizes the effectiveness difference between B and C. Now,

$$\begin{aligned} \text{MED}(A,B) &= |S(A') - S(B')| \\ &= |S(A') - S(C') + S(C') - S(B')| \\ &\leq |S(A') - S(C')| + |S(C') - S(B')| \\ &\leq |S(A'') - S(C'')| + |S(C''') - S(B''')| \\ &= \text{MED}(A,C) + \text{MED}(C,B). \end{aligned} \tag{7.7}$$

The second-last step holds by the definition of MED, as maximizing effectiveness difference.

The degree to which MED satisfies our requirements with respect to top-weighting, indefinite rankings, and incomplete rankings, depends upon the associated effectiveness measure. While we are not aware of any established effectiveness measure that cannot handle incomplete rankings, a few measures — such as precision@k —are not top weighted. Moreover, several effectiveness measures — such as precision@k and nDCG@k — are parameterized by a depth k. Changing the value of k effectively creates a new measure, in the sense that precision@5 is a different measure than precision@10. For such effectiveness measures, extending the ranked lists beyond k does not change the value of the measure, or the value of MED. For these measures, MED appropriately supports indefinite rankings only to depths less than or equal to k. When $K < k$, we compute MED by filling ranks $K + 1$ to k with free variables in both lists. Other measures —such as RBP and ERR — are not parameterized by depth and are notionally computed to infinity, providing stronger support for indefinite rankings.

7.2 Simple Dot Product Measures

In this section, we maximize Equation 7.3 for a class of simple but widely used effectiveness measures. These measures may be expressed as a normalized dot product between a vector of relevance values and a vector of rank-based discount values. The class includes RBP and nDCG, along with other measures [117].

Let C be a ranked list (which could be either A or B) represented by a vector $C = \langle c_1, c_2, ...\rangle$ of relevance values. Let $D = \langle d_1, d_2, ...\rangle$ be a vector of discount values, where the value of d_i

depends only on the rank, i. Effectiveness is computed as the normalized dot product of C and D:

$$S(C) = \frac{C \cdot D}{\mathcal{N}}, \tag{7.8}$$

where the normalization factor $\mathcal{N}$ is a constant, which may depend on assumptions about the user or on characteristics of the document collection, such as the number of relevant documents it contains at each relevance grade. Generally, normalization serves to map the value of the measure into the range $[0 : 1]$. To compute MED, we require $\mathcal{N} > 0$ (and we are not aware of any established effectiveness measure where this requirement does not hold).

Each d_i in the discount vector may be interpreted as the probability that a user scanning the search results will reach rank i, and hence view the document at that rank [30, 34, 118], but this interpretation is not explicitly required in this chapter. To compute MED, we require discount values to be non-negative and to decrease monotonically with increasing rank, i.e., $d_i \geq d_j$ if $i < j$ (and we are not aware of any established effectiveness measures where this requirement does not hold).

Maximizing Equation 7.3 for simple dot product measures is straightforward. First, we set all predetermined variables in both lists to their known values. Second, we set all free variables in A to r_G and all free variables in B to $r_0 = 0$. If $S(A) - S(B)$ is maximized, then all free variables in A must have value r_G, for otherwise we could increase $S(A) - S(B)$ by increasing the value of these free variables. Similarly, if $S(A) - S(B)$ is maximized, then all free variables in B must have value 0.

Finally, given a constraint $a_n \equiv b_m$, the contribution of these variables to $S(A) - S(B)$ is

$$\frac{a_n d_n - b_m d_m}{\mathcal{N}}. \tag{7.9}$$

Since discount decreases monotonically with increasing rank, $S(A) - S(B)$ is maximized by setting $a_n = b_m = r_G$ if $n < m$, or $a_n = b_m = 0$ if $n > m$. If $n = m$ then these variables contribute nothing to the value of $S(A) - S(B)$ and can be set to any value.

For example, we may express precision@k as a simple dot product measure using a vector of binary relevance values, a normalization factor of k, and a discount with the first k values set to one and the remaining values set to 0:

$$D = \langle d_1 = 1, d_2 = 1, ..., d_k = 1, d_{k+1} = 0, ...\rangle. \tag{7.10}$$

In the case that the ranked lists are not fully specified to depth k, we complete them with arbitrarily chosen free variables. Applying the relevance assignment process above, we see that MED for the precision@k effectiveness measure (which we call MED-precision@k) is just one minus the overlap between A and B at depth k:

$$1 - \frac{|A_{1:k} \cap B_{1:k}|}{k}. \tag{7.11}$$

7.2.1 Computing MED-RBP

Moffat and Zobel [118] define the formula for computing rank biased precision (RBP) as:

$$S(C) = (1 - \psi) \sum_{i=1}^{\infty} c_i \psi^{i-1}, \tag{7.12}$$

where $C = \langle c_1, c_2, ... \rangle$ is a vector of graded relevance values, although the definition works equally well for binary values. For the experiments reported in this chapter, we assume $r_G = 1$. As it does in RBO, the parameter $0 < \psi < 1$ represents user persistence, with larger values representing a more patient user. We may easily express this formula in the form of Equation 7.8, making it straightforward to compute MED for the RBP measure (MED-RBP). When computing MED-RBP for the experiments reported in this chapter, we set $\psi = 0.9$, a typical choice [157]

MED-RBP provides strong properties in support of indefinite rankings. For ranked lists specified only to depth K, we compute MED-RBP down to infinite depth by assuming arbitrary free variables in both lists below K. To maximize $S(A) - S(B)$, in list A we set these free variables to 1, and in list B we set these free variable to 0, so that we maximize:

$$\begin{aligned} S(A) - S(B) &= (1 - \psi) \left(\sum_{i=1}^{K} (a_i - b_i) \psi^{i-1} + \sum_{i=K+1}^{\infty} \psi^{i-1} \right) \\ &= (1 - \psi) \left(\sum_{i=1}^{K} (a_i - b_i) \psi^{i-1} \right) + \psi^K. \end{aligned} \tag{7.13}$$

If the ranked lists are later specified to greater depth, increasing K and potentially introducing new bound and predetermined variables, MED cannot increase. Moreover, the MED-RBP distance cannot decrease by more than $2\psi^K$ as the depth goes to infinity.

7.2.2 Computing MED-DCG and MED-nDCG

At the time Järvelin and Kekäläinen [66] created normalized discounted cumulative gain (nDCG) no other established effectiveness measure accommodated graded relevance values. Since then, nDCG has become widely used for Web-related research. In this chapter, we work with a version of nDCG that has become standard in the research literature and through industry practice [30, 120]:

$$S(C) = \left(\frac{1}{\textit{ideal DCG}}\right) \sum_{i=1}^{k} \frac{c_i}{\log(i+1)}. \tag{7.14}$$

$C = \langle c_1, c_2, ... \rangle$ is a vector of relevance values, where each c_i is one of r_0 ... r_G. When computing nDCG for our experiments, we set $k = 20$.

For nDCG@k, relevance values are computed using the formula [30]:

$$r_j = \frac{2^j - 1}{2^G}, \;\; j = 0, ..., G. \tag{7.15}$$

For our experiments, the test collection employs three relevance grades, so that $r_0 = 0$, $r_1 = 1/4$, and $r_2 = r_G = 3/4$. These grades indicate documents that are judged to be non-relevant, relevant, and highly relevant, respectively.

We may express Equation 7.14 in the form of Equation 7.8, with a discount vector of

$$D = \langle 1, ..., 1/\log(i+1), ... \rangle. \tag{7.16}$$

Järvelin and Kekäläinen call the dot product $C \cdot D$ *discounted cumulative gain* (DCG). DCG is then normalized by ideal DCG to give nDCG.

DCG is a useful measure in and of itself [1], and we can apply the methods of this sections to compute MED-DCG. On the other hand, the computation of MED-nDCG requires an value for ideal DCG, which in turn requires knowledge about characteristics of the document collection, specifically the number of relevant documents at each relevance grade. Since complete relevance information will not be available (or else we would just compute actual differences) we must make assumptions about these characteristics for normalization purposes.

7.2.3 Normalization for MED-nDCG

Ideal DCG is a constant, defined as the maximum DCG achievable over the collection, which can be determined by ranking all the most relevant documents first, followed by the next most relevant, and so on. In theory, determining ideal DCG may require exhaustive judging, which is rarely feasible on realistically sized collections. In practice, ideal DCG is usually estimated from the known relevant documents surfaced during an retrieval experiment. If changes to ranking algorithms surface new relevant documents, the estimate of ideal DCG may grow and the value of nDCG may drop, even when the changes improve the algorithm. When applying nDCG as an effectiveness measure, care must be taken to account for this potential growth in ideal DCG.

When computing MED for the nDCG measure (MED-nDCG), we may have no judgments at all. For MED-nDCG and other measures requiring a normalization computed from collection characteristics, we adopt the convention of normalizing MED into the range $[0 : 1]$, making the assumption that the collection contains the necessary number of maximumly relevant documents for this purpose. Following this convention, we define

$$\textit{ideal DCG} = \mathcal{N} = \sum_{i=1}^{k} \frac{r_G}{\log{(i+1)}}. \tag{7.17}$$

Using this normalization factor, the value of MED-nDCG will fall in the range $[0 : 1]$, producing a MED-nDCG value of 0 for identical lists and 1 for completely different lists. This convention provides a standard method for extending MED to measures requiring knowledge of collection characteristics, which we apply to MED-MAP in section 7.3.

Admittedly, we do lose one property of MED by adopting this convention. The calculation of nDCG requires the estimation of a constant that depends solely on the collection, not on the ranked lists being measured. With no relevance judgments available, we have no estimate for the value of this constant. By normalizing it away, as we do above, we lose the property that MED can be interpreted as an actual maximum difference in effectiveness values. MED-nDCG continues to be a metric in the mathematical sense, able to measure distances between rankings, but these distances are scaled by an unknown constant. In this regard, MED-nDCG is no different than existing correlation coefficients, including RBO, which are also dimensionless quantities.

Alternatively, for some queries we may know, or be able to reasonably guess, properties of the collection that would impact normalization. For example, we may known through the application of a query type classifier that a given query is navigational, and we may know from experience that a navigational query typically has one highly relevant document and a small number — no more than a dozen, say — of marginally relevant documents. Under these assumptions the computation of MED-nDCG would require a different normalization constant and additional constraints on the optimization, limiting the values of variables. We leave the exploration of this idea for future work.

7.3 Computing MED-AP

Average precision (AP) is defined over a vector $C = \langle c_1, c_2, ... \rangle$ of binary relevance values as:

$$S(C) \ = \ \frac{1}{R}\sum_{i=1}^{k}(c_i \cdot \ \text{precision@}i) \ = \ \frac{1}{R}\sum_{i=1}^{k}\frac{c_i}{i}\sum_{j=1}^{i}c_j, \tag{7.18}$$

where R indicates the number of relevant documents in the collection, and k an arbitrary maximum depth for computation. Although it is rarely made explicit in the research literature as AP@k, this maximum depth is as important for AP as it is for precision@k and nDCG@k. Changing k effectively creates a different measure. When computing AP for our experiments, we set $k = 100$.

Like ideal DCG, determining R theoretically requires exhaustive judging. However, as we did for ideal DCG, when computing MED for the AP measure (MED-AP) we adopt the convention of normalizing MED into the range $[0 : 1]$. We replace R with k in Equation 7.18, producing a MED-MAP value of 0 for identical lists and 1 for completely different lists.

$$S(C) = \frac{1}{k}\sum_{i=1}^{k}\frac{c_i}{i}\sum_{j=1}^{i}c_j. \tag{7.19}$$

Unfortunately, Equation 7.19 is quadratic in its relevance values and does not fit the simple dot product form assumed in Section 7.2. Maximizing $S(A) - S(B)$ requires a little more effort than it does for those measures. Fortunately, this maximization problem can be re-expressed as a quadratic 0-1 optimization problem, a heavily researched and well understood class of problems [128, 16]. Our goal is to assign relevance values to the documents in lists A and B that maximizes

$$S(A) - S(B) = \frac{1}{k}\left(\sum_{i=1}^{k}\frac{a_i}{i}\sum_{j=1}^{i}a_j - \sum_{i=1}^{k}\frac{b_i}{i}\sum_{j=1}^{i}b_j\right). \tag{7.20}$$

If $S(A) - S(B)$ is maximized, the free variables in A must be set to one, and the free variables in B must be set to zero, since the value of AP increases with more relevant documents. Of course, predetermined variables must be set to their known values.

After setting the values for free and predetermined variables, our next step is to replace each pair of variables appearing in a constraint with a single variable. To do this, we create a combined variable vector $Z = \langle z_1, z_2, ..., z_{k'}\rangle$, where $k' \leq k$ is the number of constraints. Each document appearing in both in A and B corresponds to a single variable in Z. The ordering of Z is arbitrary. Equation 7.20 can now be re-written in the form

$$Z^T QZ + L^T Z + F, \tag{7.21}$$

where Q is a matrix of order k', L is a vector of dimension k', and F is a constant. Equation 7.21 is the standard form for quadratic 0-1 optimization, a heavily studied NP-complete problem. Quadratic 0-1 optimization is equivalent to the weighted max-cut problem, one of Karp's original 21 NP-complete problems.

To approximate MED-AP, we implemented a version of *tabu search* [16], a standard local search method that creates and maintains a set of disallowed (or *tabu*) moves to avoid repeated visits to suboptimal solutions. Unfortunately — while the computation of MED for dot product measures is essentially instantaneous — the computation of MED-AP may require a second or so on a typical desktop machine.

7.4 Computing MED-ERR

Expected reciprocal rank (ERR) is based on the *cascade model* of user browsing behavior over search results [30, 37]. The model implicitly assumes that the user is seeking a single relevant document. After entering a search query and receiving a result list, the user scans the list, starting at the first result. With probability c_1 the user finds the information she seeks and stops browsing. Otherwise, with probability $1 - c_1$, she continues to the second result, and so on. In general, if the user reaches the result at rank i, she finds the information she seeks with probability c_i, or continues on to the result at rank $i + 1$ with probability $1 - c_i$. Thus, the probability that the user reaches rank i is

$$\prod_{j=1}^{i-1}(1 - c_j). \tag{7.22}$$

ERR is then defined as the expected reciprocal rank where the user's information need is satisfied:

$$S(C) = \sum_{i=1}^{\infty} \frac{c_i}{i} \prod_{j=1}^{i-1}(1 - c_j). \tag{7.23}$$

ERR uses the same relevance grades as nDCG, as defined by Equation 7.15. ERR is not normalized; its value naturally falls into the range $[0 : 1]$ (with a maximum value $<$ 1.0).

The cascade model is closely related to the user model incorporated into RBO and RBP, as described in the introduction. For RBO and RBP, the probability that the user will move from rank i to $i + 1$ is constant (ψ). Under the cascade model, this inter-rank transition probability depends on the relevance of the document at rank i, and the probability that the user will reach rank i depends on the relevance of all the documents appearing before it. If a few highly relevant documents appear above rank i, the probability of reaching that rank becomes relatively small. For example, if $r_G = 3/4$, it takes only four such documents for the probability in Equation 7.22 to drop below 1%. After viewing five such documents, less than one in a thousand users will continue.

Under ERR, if $S(A) - S(B)$ is maximized, then free variables in A must be set to r_G and free variables in B must be set to $r_0 = 0$. To demonstrate this claim, first consider the variable

a_n from list A, and then let

$$\begin{aligned}
\alpha_0 &= \sum_{i=1}^{n-1} \frac{a_i}{i} \prod_{j=1}^{i-1} (1 - a_j), \qquad (7.24)\\
\alpha_1 &= \prod_{j=1}^{n-1} (1 - a_j), \text{ and}\\
\alpha_2 &= \sum_{i=n+1}^{\infty} \frac{a_i}{i} \prod_{j=n+1}^{i-1} (1 - a_j),
\end{aligned}$$

which are constants with respect to a_n, with $0 \leq \alpha_0 \leq 1$, $0 \leq \alpha_1 \leq 1$, and $0 \leq \alpha_2 \leq 1/(n+1)$. We may now express $S(A)$ as a linear function of a_n:

$$S(A) = (\alpha_0 + \alpha_1\alpha_2) + a_n\alpha_1(1/n - \alpha_2), \qquad (7.25)$$

where $(1/n - \alpha_2) > 0$. Thus, if a_n is a free variable, its value must be r_G for $S(A) - S(B)$ to be maximized. Similarly we may express $S(B)$ as a linear function of a variable b_m from list B:

$$S(B) = (\beta_0 + \beta_1\beta_2) + b_m\beta_1(1/m - \beta_2), \qquad (7.26)$$

where β_0, β_1, and β_2 are constants with respect to b_m. Thus, if b_n is a free variable, its value must be 0 for $S(A) - S(B)$ to be maximized.

Moreover, if $S(A) - S(B)$ is maximized, the value of bound variables may be set to either r_G or 0. To demonstrate this claim, we combine Equations 7.25 and 7.26 under the assumption that $a_n \equiv b_m$ are bound variables, with $a_n = b_m = x$, giving

$$S(A) - S(B) = (\alpha_0 + \alpha_1\alpha_2) - (\beta_0 + \beta_1\beta_2) + x\gamma, \qquad (7.27)$$

where $\gamma = \alpha_1(1/n - \alpha_2) - \beta_1(1/m - \beta_2)$. Depending on the sign of γ, the value of x must be either r_G or 0 for $S(A) - S(B)$ to be maximized. If $\gamma = 0$, x may be set to either value.

Thus, in maximizing $S(A) - S(B)$ we set free and bound variables to either r_G or 0, allowing us to ignore intermediate relevance grades. Unfortunately, maximizing $S(A) - S(B)$ still requires us to solve a highly non-linear optimization problem, with what are effectively 0-1 constraints. Fortunately, we can take advantage of properties of ERR to efficiently approximate the solution.

In particular, as we noted previously, the value of ERR largely depends on the position of the first few relevant documents. As the user views more and more relevant documents, the probability she will continue to lower ranks drops exponentially.

Consider the calculation of $S(A)$ for some assignment to the variables in A. We start calculating the summation at rank $i = 1$. Suppose at rank k we have encountered p variables with value r_G. The sum over the remaining ranks is bounded by

$$\begin{aligned}
\epsilon &= \sum_{i=k+1}^{\infty} \frac{c_i}{i} \prod_{j=1}^{i-1} (1 - c_j) && (7.28) \\
&\leq \sum_{i=p+1}^{\infty} \frac{c_i}{i} \prod_{j=1}^{i-1} (1 - c_j) \\
&\leq \sum_{i=p+1}^{\infty} \frac{r_G}{i} \prod_{j=1}^{i-1} (1 - r_G) \\
&= (1 - r_G)^p \sum_{i=p+1}^{\infty} \frac{r_G}{i} \prod_{j=p+1}^{i-1} (1 - r_G) \\
&< \frac{r_G (1 - r_G)^p}{p+1} \sum_{i=0}^{\infty} (1 - r_G)^i \\
&= \frac{(1 - r_G)^p}{p+1}.
\end{aligned}$$

If $p = 5$ and $r_G = 3/4$ then $\epsilon < 0.0002$. This bound on $S(A)$ is also a bound on $S(A) - S(B)$, since setting variables in B to r_G increases $S(B)$ and decreases the difference.

Thus, to approximate MED-ERR, we adopt a brute force approach, trying all combinations of up to $p = 5$ bound variables in A down to depth 30. We set each combination to the value r_G, compute the value of ERR, and take the maximum across the combinations. This brute force approximation requires a dozen milliseconds or so on a typical desktop machine.

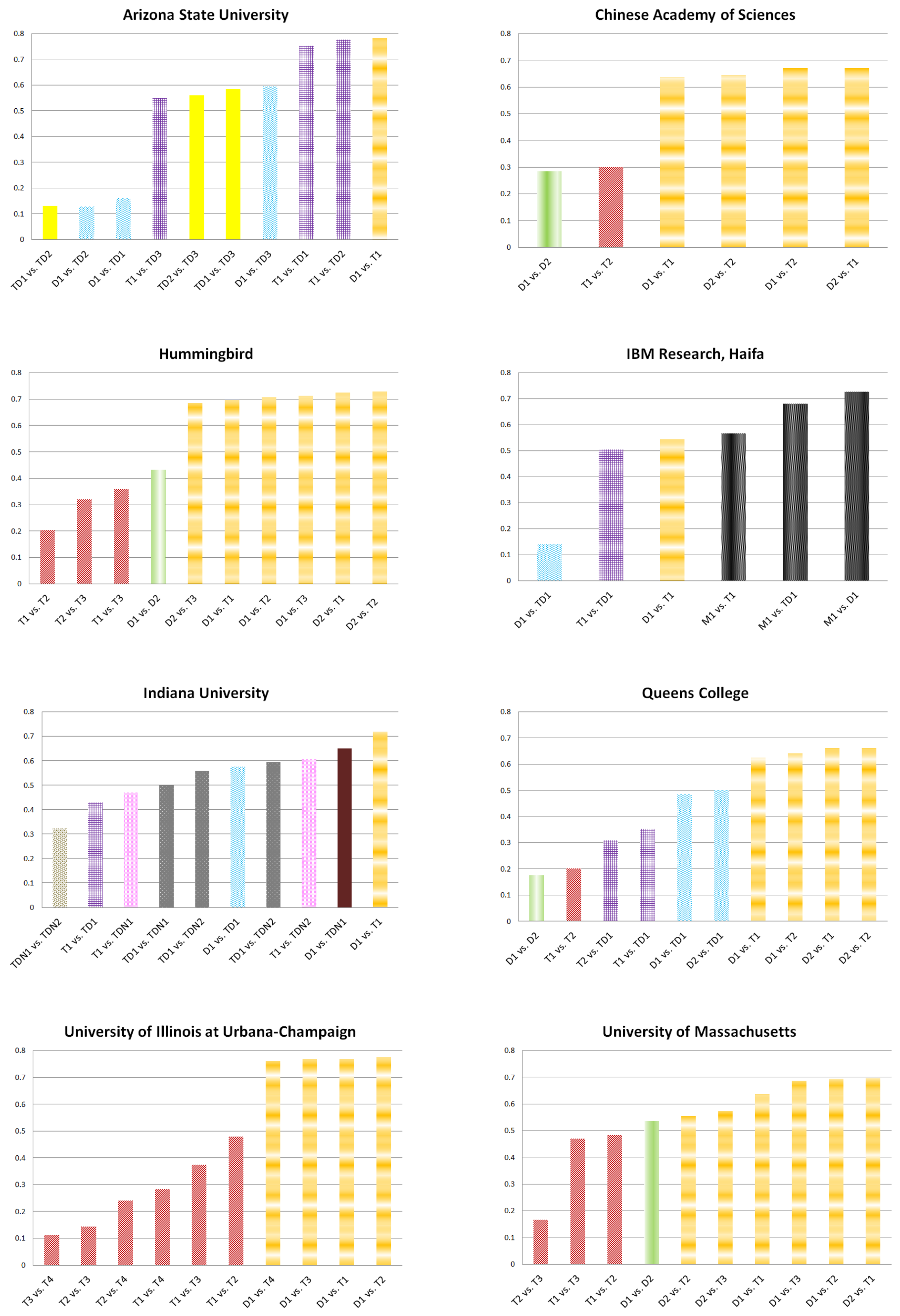


Figure 7.1: Intra-group MED-nDCG@20 values for selected TREC 2005 Robust Retrieval Track participants.

7.5 Validation

To validate MED, we employ a set of experimental runs submitted to the TREC 2005 Robust Retrieval Track [154]. As our primary goal, we hope to demonstrate that MED reflects meaningful differences between retrieval runs. The TREC Robust Track provides a particularly appropriate dataset for this purpose because its retrieval topics were chosen for their anticipated difficulty, with many standard retrieval algorithms performing poorly on them. As a result, track participants applied an unusually wide variety of retrieval methods, including some entirely novel methods, particularly in the area of query expansion.

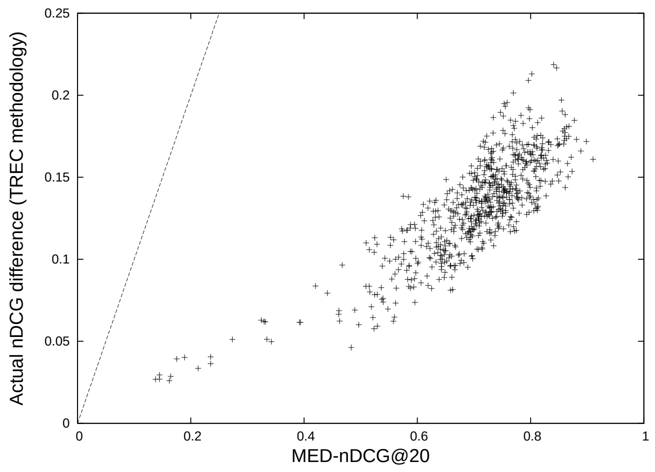
A total of 17 groups participated in the track submitting a total of 74 runs. We start by examining differences between runs submitted by the same group. If we assume that each group used a core retrieval approach across all its runs, we hope to interpret MED values in terms of meaningful changes between these runs. Among the groups who submitted multiple runs, we selected the eight groups with the best overall performance. While we now focus our attention on these eight groups, we note that results for the excluded groups exhibit consistent behavior, with the general observations we make below applying to these groups as well.

We computed MED-nDCG between all pairs of runs within each group, averaging across the 50 topics used in the track. Figure 7.1 shows the results, with one chart for each group, appearing in alphabetical order. Within each chart, pairs are ordered by increasing MED distance. Each bar corresponds to a single pair of runs, with the value of MED-nDCG given on the y-axis.

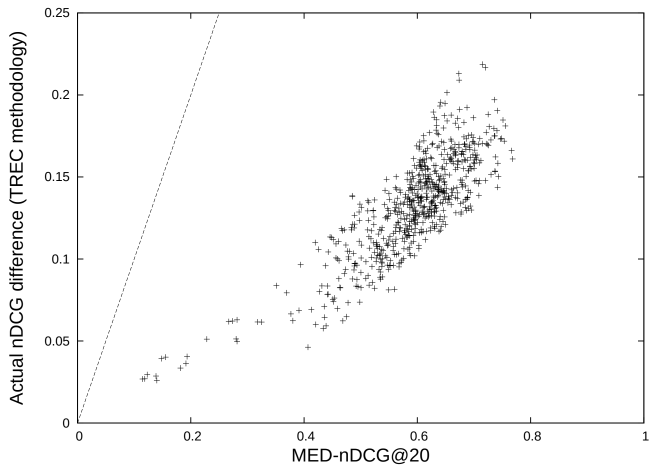
Each run is identified by a code indicating the type of information used to automatically formulate the query. Like many TREC tasks, retrieval topics for the Robust Track provide multiple expressions of the associated information need, with each topic including:

- a title field (T), providing one to three keywords expressing the information need,
- a description field (D), providing a single sentence expressing the information need, and
- a narrative (N), providing a longer expression of the information need.

Run codes indicate the set of fields used for query formulation. For example, the codes TD1 and TD2 in the Arizona State University chart both indicate runs where the title and description

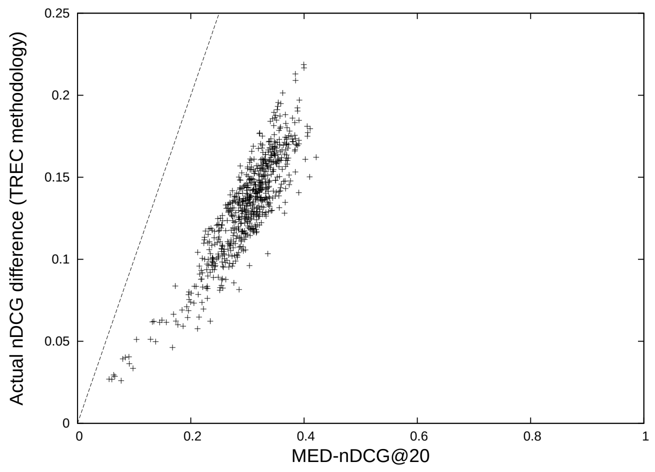


(a) No predetermined variables

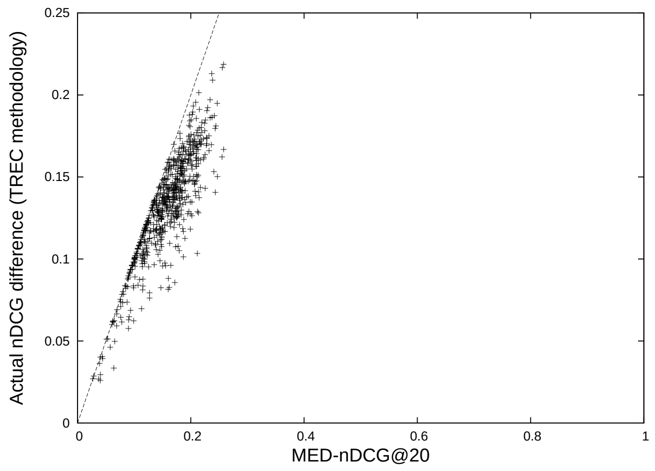


(b) 25% of available TREC qrels

Figure 7.2: Actual nDCG differences vs. MED-nDCG@20 across all pairs of runs from the TREC 2005 Robust Retrieval Track.



(a) 75% of available TREC qrels



(b) 100% of available TREC qrels

Figure 7.3: Actual nDCG differences vs. MED-nDCG@20 across all pairs of runs from the TREC 2005 Robust Retrieval Track.(Continued)

were used for query formulation. Within each set of codes, numbers are assigned arbitrarily. In addition, IBM submitted a single manual run (M1) in which the query formulation process included human assistance.

We indicate the types of queries in each pair using unique colors and shading. For example, title vs. title runs are colored in strawberry, description vs. description runs are colored in lime, and title vs. description runs are colored in tangerine. The charts for the Chinese Academy of Science, Hummingbird, Illinois, and Massachusetts contain only pairs of these types. An interesting trend is apparent in these four charts, with title vs. title pairs and description vs. description pairs being noticeably closer than title vs. description pairs.

Digging deeper, we consider the runs from Arizona State University. From the chart, we see that TD1, TD2, and D1 are all several times closer to each other than they are to the other two runs. These two runs, T1 and TD3, are closer to each other than they are to any of the other runs. The group's workshop report provides an explanation [132]. Reading their report, we learn that T1 and TD3 used the same query expansion model, which differed from the expansion model used by the other runs. Digging to the other groups reveals similar relationships reflected in MED-nDCG. For example, as shown in the chart from IBM, manual runs tend to be farther from automatic runs than automatic runs are from each other.

The computation of MED does not depend on relevance judgments. However, we would hope that MED distance provides some indication about actual effectiveness differences. Figure 7.2(a) plots (the absolute value of) actual nDCG@20 differences against MED-nDCG@20 distances across all pairs for the groups listed in Figure 7.1. The plot includes both intra-group pairs and inter-group pairs, 703 pairs in total. The plot shows a clear correlation, although the MED-nDCG distances have much higher values.

As relevance judgments become available, creating predetermined variables, MED-nDCG distances become increasingly correlated with actual nDCG differences, moving closer and closer to the actual differences. Figure 7.2(b) plots actual nDCG differences against MED-nDCG after we set variables to predetermined values from the using 25% of available TREC relevance judgments, randomly selected. Figure 7.3(a) provides an equivalent plot for 75% of available TREC judgments, and Figure 7.3(b) provides an equivalent plot for 100% of available TREC judgments. In this last plot, all of the MED-nDCG distances do not match actual differences

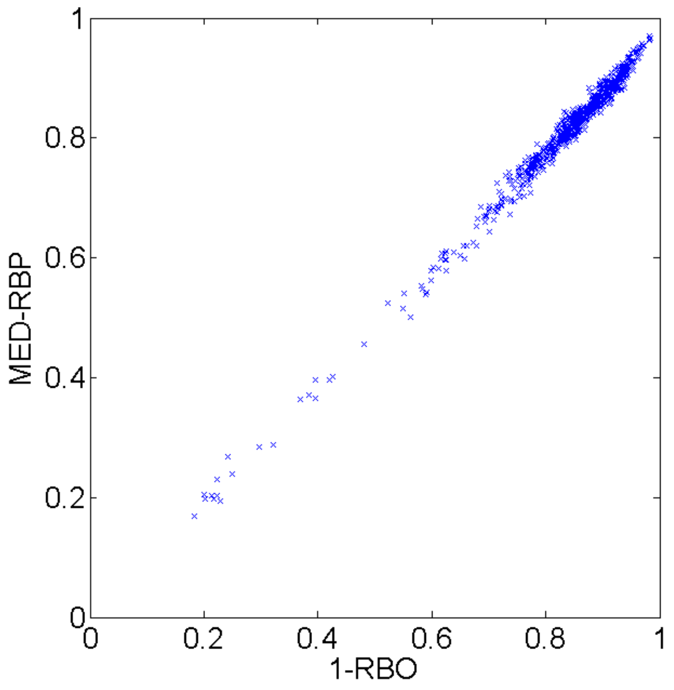


Figure 7.4: RBO vs. MED-RBP across all pairs of runs.

because TREC 2005 Robust Track runs were not fully judged down to depth 20, and TREC assumes unjudged documents to be non-relevant.

Figure 7.5 provides comparisons between RBO, MED-nDCG, MED-RBP, MED-AP and MED-ERR. In all plots, most of the points fall toward the upper right, indicating larger differences in the pairs. The pairs appearing towards the lower left are generally from the same groups, using the similar retrieval techniques.

Given that MED-RBP and RBO share a common user model, we would expect a high correlation between them. Figure 7.5(a) shows the correlation. The strength of this correlation suggests that other variants of MED may similarly reflect the user models underlying those measures. Interestingly, MED-RBP is even more highly correlated with MED-nDCG@20, as shown in Figure 7.5(b). This correlation is related to the choice of persistence parameter ($\psi = 0.9$). Lower and higher values for ψ produce weaker correlations. Figures 7.5(e) and 7.5(f) show how values of MED-nDCG change for different values of k (5, 20, and 100).

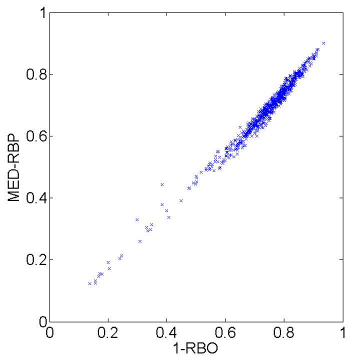
7.6 Comparison with Prior Work

As discussed in the introduction, the RBO similarity measure of Webber et al. [157] directly inspired our efforts. As part of that paper, Webber et al. provide a substantial review of related research up to early 2010, when the final version of their paper was submitted and accepted. We encourage readers who are interested in a thorough analysis of this prior work to consult that paper. In this section, we touch only on the prior work that is most closely connected with our efforts, including papers appearing since early 2010.

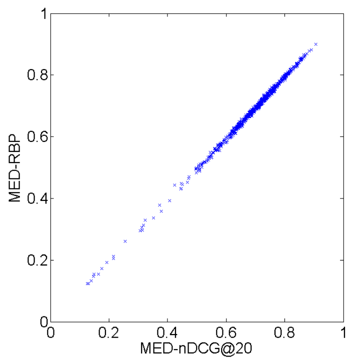
Working independently of Webber et al., and publishing at roughly the same time, Sun et al. [143] identified a similar set of desiderata for rank similarity in the context of search. These desiderata essentially include a requirement for top-weighting, and an ability to handle indefinite and incomplete results. In addition, they suggest that measures should be symmetric, should be computationally efficient, and should allow meaningful aggregation over multiple queries. To address these desiderata, Sun et al. defined a similarity measure based on a weighted version of the Hoeffding distance. They then applied this measure to visualize differences in search engines through multidimensional scaling.

MED satisfies the additional desiderata of Sun et al. Symmetry is straightforward from the definition of MED. The versions of MED detailed in this chapter satisfy the efficiency requirement to varying degrees, and all are reasonably efficient. Computation of MED for the dot product measures is essentially instantaneous, with run times dominated by I/O and data conversion. For members of the MED family not covered in this chapter, computational efficiency will depend on the details of the associated effectiveness measure. MED also derives its approach to aggregation from the associated effectiveness measure, where an arithmetic mean is typical, but other approaches are possible [130].

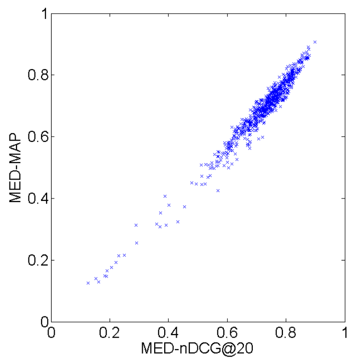
Kumar and Vassilvitskii [82] proposed various extensions to the Kendall's τ and Spearman's footrule correlation coefficients intended to address the requirements of search. These extensions are intended to handle top weighting, known document relevance, and the similarity between



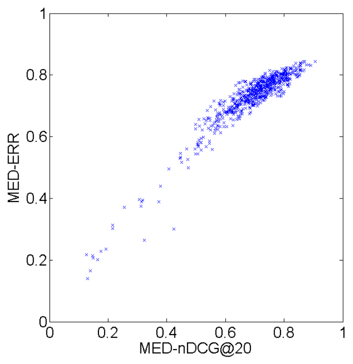
(a) MED-RBP vs. RBO



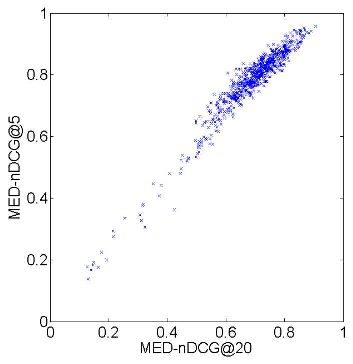
(b) MED-RBP vs. MED-nDCG@20



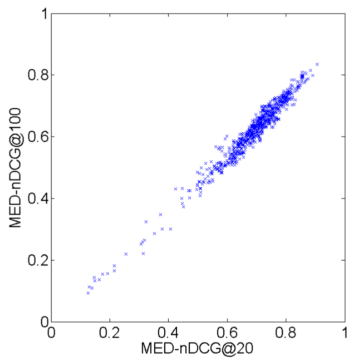
(c) MED-AP vs. MED-nDCG@20



(d) MED-ERR vs. MED-nDCG@20



(e) MED-nDCG@5 vs. MED-nDCG@20



(f) MED-nDCG@100 vs. MED-nDCG@20

Figure 7.5: Comparisons between RBO, MED-nDCG, MED-RBP, MED-AP and MED-ERR for all runs from the TREC 2005 Robust Retrieval Track.

documents. They demonstrated that these extensions maintain the Diaconis-Graham inequality, which guarantees that Kendall's τ and Spearman's footrule differ by at most a constant factor.

Like Webber et al. [157] and Sun et al. [143], Kumar and Vassilvitskii suggest a list of desiderata for rank similarity in the context of search. These desiderata include support for top weighting and the triangle inequality. While all their requirements are not precisely defined, MED appears to satisfy them with one interesting exception. The family members of MED defined in this chapter do not provide support for inter-document similarity. When comparing two rankings, replacements or swaps of documents with similar content may not greatly impact the user, and a similarity measure might reasonably reflect this consideration. From the perspective of MED, we can trace this requirement back to the associated effectiveness measures, which also do not appropriately handle documents with similar or duplicate content. As researchers begin to consider these issues in the design of effectiveness measures [138], solutions will transfer naturally to MED.

Several other researchers have also adapted existing rank similarity measures to the requirements of search, particularly the Kendall's τ correlation coefficient [166, 27, 109]. These efforts primarily address the issue of top weighting, paying less attention to incomplete and indefinite rankings. Rather than measuring differences between result lists, much of this work focuses on the problem of comparing system rankings for the purpose of validating newly proposed effectiveness measures, and other evaluation methodologies. In our work, rather than adapting existing rank similarity measures to the requirements of search, we adapt existing search effectiveness measures to the computation of rank similarity, inheriting properties of these effectiveness measures and cleanly accommodating known relevance information.

Rank similarity measures provide a method for comparing search results without the need for relevance information. Numerous researchers have examined the related problem of estimating effectiveness measures and comparing systems using limited relevance information. Most notably, Carterette et al. [28, 29, 26] define algorithms for selecting minimal sets of documents for judging, with the aim of ordering a group of retrieval systems at a given confidence level. Similar in spirit to MED, and using equivalent methods, documents are selected to maximize differences in retrieval effectiveness scores.

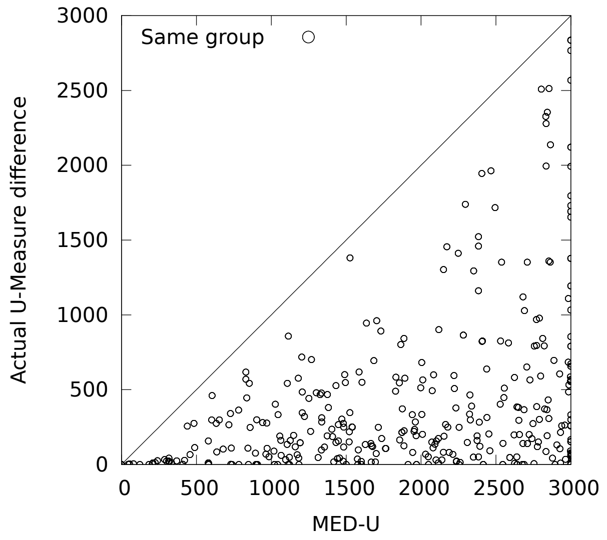
Interestingly, in his doctoral thesis Carterette [26, pages 156–157] notes the need for rank similarity measures that incorporate top-weighting and an ability to handle incomplete rankings. To address these shortcomings of traditional correlation coefficients, such as Kendall's τ and Spearman's ρ, he proposes his own rank similarity measure based on differences in reciprocal rank. He employs this measure to study properties of his judging algorithms and TREC runs.

We take his work a step further, recognizing and demonstrating that maximized effectiveness difference itself can form the basis for measuring rank similarity. In addition, we extend the idea of maximized effectiveness difference to effectiveness measures that did not exist at the time of his thesis, including ERR and RBP. Some of our ideas could also be applied back to his work, including our generalization for the dot-product measures, our formulation for AP, and our formulation for ERR.

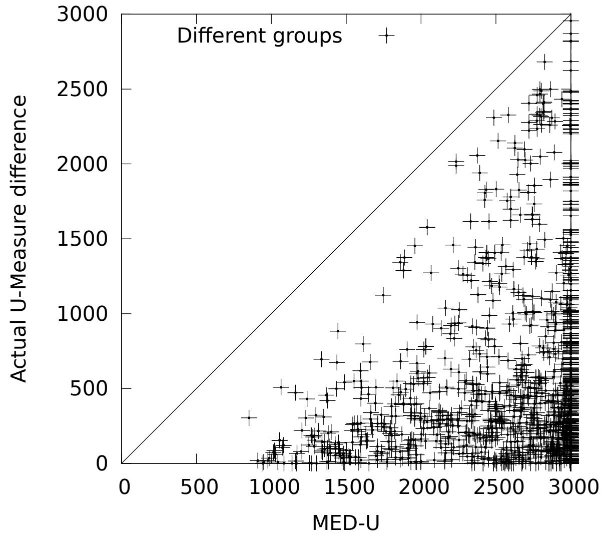
Apart from Webber et al. [157] and Carterette et al. [28, 29, 26], the work closest to ours is the AnchorMap measure proposed by Buckley [21]. AnchorMap compares two ranked lists by assuming that the top k documents from one list are relevant, and then computing AP for the other list using this relevance information. While AnchorMap is not symmetric, and has other limitations [157], the idea captures the essence of our proposal.

Several researchers have used rank similarity measures to monitor and compare commercial search engines [157, 82, 25, 143]. Cardoso and Magalhães [25] applied RBO to compare the behavior of two of the most heavily used commercial search engines. In addition, they applied Jensen-Shannon divergence to measure the similarity between search results based on the contents of the top documents returned. They argue that this second approach provides deeper insights into the differences between the search engines, a view that reflects some concerns of Kumar and Vassilvitskii [82].

In other related work, Büttcher et al [22] trained a classifier to predict the relevance of unjudged documents. Jensen et al. [67] combined limited manual judgments with automatically generated pseudo-judgments to evaluate search results in dynamic environments. Yilmaz et al. [167] developed sampling methods for estimating standard effectiveness measures. Sakai and Kando [134] explored the impact of missing relevance judgments on standard effectiveness measures across four large test collections.



(a) Pairs of retrieval results from the same group



(b) Pairs of retrieval results from different groups

Figure 7.6: Actual U-measure differences vs. MED-U@12000 across pairs of passage-oriented runs on passage-oriented topics from the TREC 2004 HARD Track. We plot a 20% sample, randomly selected, for visual clarity.

7.7 Beyond the Ranked List

MED may be extended to measure distances between search results beyond the ranked list. As a simple example, we consider the *U-measure* proposed by Sakai et al.[133]. This measure assumes that a retrieval result is not expressed as a ranked list of documents, but rather as a *trailtext*, a concatenation of all text presented to (or seen by) the user. Given a trailtext of length l, the U-measure is computed as

$$\frac{1}{\mathcal{N}} \sum_{i=1}^{l} c_i d_i. \tag{7.29}$$

Just as Equation 7.1 sums over documents in the ranked list, the U-measure sums over character offsets in the trailtext. The value c_i represents the graded relevance value of the character at offset k, and the value d_i represents a position-oriented discount. For their experiments, Sakai and Dou use no normalization ($\mathcal{N} = 1$) and a linear discount ($d_i = 1 - i/l$) but clearly the measure can be generalized to other discounts and normalization values.

The U-measure provides one method for evaluating retrieval systems that attempt to return sub-document components, such as text passages, to the user. For example, a system might respond to a query by extracting relevant passages from books, and other longer documents, placing them in order for reading by the user. Essentially this problem was addressed by a task included as a option to the TREC 2004 HARD Track [3].

The 2004 HARD Track included 25 topics for which relevance judging was performed at the sub-document level, by identifying offsets and lengths of relevant passages within documents. The corpus for the track comprised more than a half-a-million news articles from the year 2003, drawn from a variety of sources, including the Associated Press, the New York Times, and the Washington Post. Participating systems returned a ranked list of passages for each of the passage-level topics, where each passage consisted of a starting byte offset within a document and a length in bytes.

Since characters within passages cannot be arbitrarily re-ordered, standard rank correlation coefficients are not appropriate to compute similarities between these passage-oriented results. However, we may apply apply MED for the U-measure (MED-U) for this purpose. Although we have switched from documents to characters, the properties of MED from Section 7.1 continue to hold; nothing in that section requires variables to represent documents.

The U-measure may be viewed as a type of dot-product measure, as discussed in Section 7.2. We apply the methods of that section to compute MED-U over passage-oriented retrieval runs submitted to the track. We compute the measure down to depth $l = 12,000$, consistent with HARD Track practice, and use binary values for c_i, i.e, a character is either relevant or not. Results are shown in Figure 7.6.

Figure 7.6 plots actual U-measure differences vs. U-MED across pairs of passage-oriented topics taken from passage-oriented runs. For visual clarity, we plot a 20% sample, selected randomly. Figure 7.6(a) plots intra-group pairs; Figure 7.6(b) plots inter-group pairs. In contrast to Figure 7.2(a), retrieval results tend to be very different from one another, although runs from the same group tend to be closer than runs from different groups. Many pairs from different groups are completely different (with the maximum possible MED-U@12000 value of 2999.75). These large differences may reflect the unusual nature of the task, as well as unfamiliarity with passage retrieval. If passage retrieval was a better understood task, we might much expect closer results, particularly for intra-group runs.

In other work, Smucker and Clarke [138, 137, 35] present a general effectiveness measure, called *time-biased gain*, which may be applied to measure search results in many forms. For example, retrieval results might be specified as summaries, snippets, or video clips. As the user interacts with the results, the total benefit to the user is expressed as a function $G(t)$, the *cumulative gain* at time $t \geq 0$. This gain is realized as relevant material is encountered by the user, perhaps by reading relevant text or viewing relevant video. For a retrieval result X, with associated cumulative gain function $G_X(t)$, time-biased gain is

$$S(C) = \int_0^\infty \frac{dG_X}{dt} D(t) dt. \tag{7.30}$$

$D(t)$ is a *decay* function, typically exponential, indicating the probability that the user interacts with the material until time t. Given two retrieval results, A and B, we treat the material shared by these results as *bound material*, material appearing in only one result as *free material*, and material with known relevance as *predetermined material*. To compute MED(A,B), we must assign relevance to bound and free material to maximize

$$S(A) - S(B) = \int_0^\infty \frac{d(G_A - G_B)}{dt} D(t) dt \ . \tag{7.31}$$

MED-RBP, MED-nDCG, MED-ERR, and even MED-AP are all specific examples of this general equation, under appropriate list-oriented definitions for gain and decay [138, 35]. We leave to future work the application of this equation to measure search-result distances beyond the ranked list.

Chapter 8

Conclusion and Future Directions

In this thesis, we explore problems in tracking news events in social media streams, especially Twitter streams. We examine several different tasks involving tracking. We learn an improved understanding of word usage in social media, comparing it with word usage in other source platforms. Given an interest source document, we propose a succinct query generation approach by executing a series of probe queries selected from candidate extracted key words over a sample of target resource corpus. Given an explicit user interest profile, we develop an automatic system for monitoring live social media streams and pushing relevant updates to users directly on their mobile devices or desktops. Different pushing strategies and threshold setting methods are examined through this system. Furthermore, multiple evaluation metrics are analyzed on various similar real-time tracking and pushing notification systems. A framework with different user model assumption is proposed towards the best evaluation measurement for the real-time tracking and pushing notification scenario. Throughout the entire thesis, ranking similarity measurement is a crucial tool for comparing intermediate results. In the last chapter, we propose a family of distance measures, each directly derived from an associate information retrieval effectiveness measure, for computing the maximized effectiveness difference of two ranked lists under the specified effectiveness measure

8.1 Improving Understanding of Word Usage in Social Media

We adapt machine translation methods across languages to explore differences between corpora written in a single language. In prior research, it was found that the positions of semantically related words in different language vector spaces are relatively similar. Based on this background, we derive a linear transformation relationship between vectors for the most frequent words from both the highly informal language used in English-language Twitter corpus and the more formal language used in Wikipedia corpus. We learn a liner projection matrix that maps one corpus to the other one. We prove our hypothesis that usage of those words appearing far apart after this transformation differs substantially between the two corpora.

After describing experiments, we list some common words with large different usage in the two corpora as examples. These examples include words that are frequently used as abbreviations in Twitter, but more commonly used as normal words with different meanings in Wikipedia; words that are often misspelling of another word in Twitter but are common words in Wikipedia; popular culture, jargon or slang words in Twitter, but different meanings in Wikipedia.

The results and findings of this work might be applied to methods that normalize the nonstandard language usage in social media. If we can distinguish misspelling and abbreviations from the social media vocabulary and translate different spellings of same meaning words into the same spelling, the performance of using both natural language processing tool and information retrieval techniques will be improved.

8.2 Succinct Query for Linking Different Resources

From a given source document, such as a news article, our approach generates succinct queries from a candidate set of extracted terms, comprising perhaps four or five terms in total. Starting from the candidate terms, we execute a series of probe queries over a sample of contemporaneous social media collection. By analyzing the results of the probes, we build a formula to compare the results of probes to the language of the source document, we rank the terms according to their ability to retrieve related material.

For each step of this process, we examine different approaches and compare the intermediate results. To extract the initial candidate term set, we compare pointwise kullback-leibler divergence method with the TextRank model for extracting key words, which has been proved to be an efficient key word extraction method in previous literatures. We have demonstrated that K-L divergence method perform better than TextRank in both effective and efficient viewpoints. Several other key phrase extraction methods are also utilized, however, they are more complex to be ranked as well as longer average length which does not satisfy our succinct requirement.

We analyze the probe queries by computing the similarity between the source document and the results of probe queries. An unsupervised method is proposed based on prior experiment results and a learning to rank method considering the similarity scores as measurement for ranking the probe queries. When evaluating our experiments, we apply both in-house assessment and crowdsourcing assessment. The two different source of assessments verify our results. We also compare the agreement of both inter-annotator and among different assessment workers.

8.3 Real-Time Tracking in Social Media and Push Notification

A push notification service provides a direct way to deliver relevant information generated from real-time tracking systems to users. In our experiments, to filter and track social media posts, simple formulae for content matching and novelty can achieve good performance. Appropriate thresholds should be set to avoid pushing non-relevant information to annoy users. Together with the simple content matching method, it is even possible to user a fixed, static global threshold across all topics, all queries. Although better performance can be achieved by carefully select dynamic thresholds based on previous days pushing performance for different topics different days. With user feedback or daily relevance judgements of pushed tweets available, systems are able to set more accurate thresholds for each topic and each day.

Push notifications should be relevant, novel, and timely. We analyze the official evaluation metrics of TREC 2015 Microblog track which provided a platform of topics and evaluations for real-time tracking of Twitter stream and push notification systems. There are some assumptions in the official evaluation metric that are arbitrary. We modify the assumptions or remove the assumptions. The performance ranking of systems are significantly different. Thus, we present the novel and surprising find: any number of reasonable evaluation metrics give rise to significantly different system rankings.

In TREC 2016 Real-time Summarization track, the organizers develop a user judgement interface for the mobile users, allowing them to judge whether each tweet pushed to them is relevant or non-relevant to the topics. This is helpful to provide real user judgements. However, the feedbacks do not go back to the pushing systems directly. In the future, if real-time judgement feedback can be made immediately available to the systems, such that they can use the feedback to learn on the performance of their pushes to the users and improve their thresholds or even algorithms. Moreover, a volume button in the user interface which indicates the users need more or fewer updates about a particular topic will also help improve the push notification service.

In the future, we hope to incorporate social signals and other non-content features into the relevance and novelty components of our system, with the goal of retaining our simple approach to thresholding, while improving overall performance. For now, only the content of each post is analyzed and compared with our topical events. More social signals, such as Twitter user information, Twitter user friends information, the posts and counts of re-posting can be added into our algorithm.

There is little empirical characterization of how real users would respond to push notifications. Ultimately, user studies are needed to help understand the push notification service better as well as ensure that metric definition and user needs actually align. We leave this as a future work.

8.4 Ranking Similarity Measurement

Rank similarity measures cannot replace traditional effectiveness measures for determining the absolute performance of search engines. However, they are more easily applied over larger numbers of queries, without the need for relevance judgements, providing an additional tool for assessing the scope of a search engine change. The MED family of rank similarity measures, satisfies various desiderata suggested in prior work for rank similarity measures in the context of search. Unlike the rank similarity measures presented in this prior work, the MED family allows us to translate our understanding and assumptions about user behavior from an existing effectiveness measure to create a rank similarity measure. In addition, MED cleanly accommodates partial relevance judgments, when this information is available, where adding relevance information can only reduce the difference between lists. If complete relevance information is available, MED reduces to a simple difference between effectiveness values. Software to compute MED for various effectiveness measures is available at plg.uwaterloo.ca/~claclark/med.

Differences between retrieval results might also be studied under conditions other than the "worst case" assumption of MED. For example, we might assume that an unjudged document is equally likely to be relevant or non-relevant. Under this assumption, we can compute a distribution of differences, perhaps treating the mean difference as a similarity measure, although it may not be a metric in the mathematical sense. The probability of relevance for an unjudged document could also be conditioned on rank, or computed by a classifier, with the goal of estimating actual differences. Some of these ideas have been partially explored in related work, as discussed. We leave additional exploration as future work.

Some newer proposals for effectiveness measures incorporate more complex user models, creating interesting and challenging optimization problems, which we hope to examine in the future. Several research groups have suggested measures that reward novelty and diversity in search results [135, 30, 34]. Computing MED for these effectiveness measures may require the explicit assignment of query interpretations to documents in order to maximize effectiveness difference. Other proposals use time as the primary indicator of user effort, creating measures that reflect the impact of snippets and other user interface features [15, 138]. Some of these proposals employed simulation as a method for determining effectiveness [15, 137, 35], and computing MED for these effectiveness measures may also require extensive simulation.

References

[1] Azzah Al-Maskari, Mark Sanderson, and Paul Clough. The relationship between IR effectiveness measures and user satisfaction. In *Proceedings of 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 773–774, 2007.

[2] M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. On sparsity and drift for effective real-time filtering in microblogs. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 419–428. ACM, 2013.

[3] James Allan. HARD track overview in TREC 2004: High accuracy retrieval from documents. In *Proceedings of 13th Text REtrieval Conference*, 2004.

[4] James Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science and Business Media, 2012.

[5] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. In *Proceedings of the Broadcast News Transcription and Understanding Workshop*, 1998.

[6] James Allan, Victor Lavrenko, David Frey, and Vikas Khandelwal. Umass at tdt 2000. In *Proceedings of Topic Detection and Tracking Workshop*, pages 109–115. Citeseer, 2000.

[7] James Allan, Victor Lavrenko, and Ramesh Nallapati. Umass at tdt 2002. In *Proceedings of Topic Detection and Tracking Workshop*. Citeseer, 2002.

[8] James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*, pages 37–45. ACM, 1998.

[9] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 3–12. IEEE, 2008.

[10] Gianni Amati, Giuseppe Amodeo, Marco Bianchi, Giuseppe Marcone, Fondazione Ugo Bordoni, Carlo Gaibisso, Giorgio Gambosi, Alessandro Celi, Cesidio Di Nicola, and Michele Flammini. Fub, iasi-cnr, univaq at trec 2011 microblog track. In *Proceedings of the 20th Text REtrieval Conference*, 2011.

[11] Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. Trec 2014 temporal summarization track overview. Technical report, DTIC Document, 2014.

[12] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING-ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics, 2006.

[13] Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R Carvalho. Exploring reductions for long web queries. In *Proceedings of the 33rd International ACM Conference on Research and Development in Information Retrieval*, pages 571–578. ACM, 2010.

[14] Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. How noisy social media text, how diffrnt social media sources. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 356–364, 2013.

[15] Feza Baskaya, Heikki Keskustalo, and Kalervo Järvelin. Time drives interaction: Simulating sessions in diverse searching environments. In *Proceedings of the 35th International ACM Conference on Research and Development in Information Retrieval*, pages 105–114, 2012.

[16] J. E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, The Management School, Imperial College, London, December 1998.

[17] Hila Becker, Mor Naaman, and Luis Gravano. Learning similarity metrics for event identification in social media. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 291–300, New York, NY, USA, 2010. ACM.

[18] Michael Bendersky and W Bruce Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval*, pages 491–498. ACM, 2008.

[19] Michael Bendersky, Donald Metzler, and W Bruce Croft. Parameterized concept weighting in verbose queries. In *Proceedings of the 34th International ACM Conference on Research and Development in Information Retrieval*, pages 605–614. ACM, 2011.

[20] Andrei Z Broder, Adam Kirsch, Ravi Kumar, Michael Mitzenmacher, Eli Upfal, and Sergei Vassilvitskii. The hiring problem and lake wobegon strategies. *SIAM Journal on Computing*, 39(4):1233–1255, 2009.

[21] Chris Buckley. Topic prediction based on comparative retrieval rankings. In *Proceedings of the 27th Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 506–507, 2004.

[22] 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 Conference on Research and Development in Information Retrieval*, pages 63–70, 2007.

[23] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval*, pages 243–250. ACM, 2008.

[24] Jaime Carbonell, Yiming Yang, John Lafferty, Ralf D Brown, Tom Pierce, and Xin Liu. Cmu report on tdt-2: Segmentation, detection and tracking. In *Proceedings of the DARPA Broadcast News Workshop*, pages 117–120, 1999.

[25] Bruno Cardoso and João Magalhães. Google, Bing and a new perspective on ranking similarity. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 1933–1936, 2011.

[26] Ben Carterette. *Low-Cost and Robust Evaluation of Information Retrieval Systems*. PhD thesis, University of Massachusetts Amherst, 2008.

[27] Ben Carterette. On rank correlation and the distance between rankings. In *Proceedings of the 32nd International ACM Conference on Research and Development in Information Retrieval*, pages 436–443, 2009.

[28] Ben Carterette and James Allan. Incremental test collections. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 680–687, 2005.

[29] Ben Carterette, James Allan, and Ramesh Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of the 29th Annual International ACM Conference on Research and Development in Information Retrieval*, pages 268–275, 2006.

[30] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 621–630, 2009.

[31] Olivier Chapelle and Ya Zhang. A dynamic Bayesian network click model for web search ranking. In *Proceedings of the 18th International World Wide Web Conference*, pages 1–10, 2009.

[32] Jilin Chen, Allen Cypher, Clemens Drews, and Jeffrey Nichols. Crowde: Filtering tweets for direct customer engagements. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*. Citeseer, 2013.

[33] Jaeho Choi and W Bruce Croft. Temporal models for microblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2491–2494. ACM, 2012.

[34] Charles L.A. Clarke, Nick Craswell, Ian Soboroff, and Azin Ashkan. A comparative analysis of cascade measures for novelty and diversity. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 75–84, 2011.

[35] Charles L.A. Clarke and Mark D. Smucker. Time well spent. In *Proceedings of the Information Interaction in Context Conference*, 2014.

[36] Margaret Connell, Ao Feng, Giridhar Kumaran, Hema Raghavan, Chirag Shah, and James Allan. Umass at tdt 2004. In *Proceedings of Topic Detection and Tracking Workshop*, 2004.

[37] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 1st International Conference on Web Search and Data Mining*, pages 87–94, 2008.

[38] Gabor Cselle, Keno Albrecht, and Rogert Wattenhofer. Buzztrack: Topic detection and tracking in email. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, pages 190–197, New York, NY, USA, 2007. ACM.

[39] Ovidiu Dan, Junlan Feng, and Brian D Davison. A bootstrapping approach to identifying relevant tweets for social tv. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, 2011.

[40] Ali Dasdan, Paolo D'Alberto, Santanu Kolay, and Chris Drome. Automatic retrieval of similar content using search engine query interface. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 701–710. ACM, 2009.

[41] Sudip Datta and Vasudeva Varma. Tossing coins to trim long queries. In *Proceedings of the 34th International ACM Conference on Research and Development in Information Retrieval*, pages 1255–1256. ACM, 2011.

[42] Jeffrey Dean and Monika R Henzinger. Finding related pages in the world wide web. *Computer Networks*, 31(11):1467–1479, 1999.

[43] Patrick Drouin. Detection of domain specific terminology using corpora comparison. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004.

[44] Yajuan Duan, Furu Wei, Ming Zhou, and Heung-Yeung Shum. Graph-based collective classification for tweets. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2323–2326. ACM, 2012.

[45] Georges Dupret and Ciya Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 181–190, 2010.

[46] Miles Efron. Hashtag retrieval in a microblogging environment. In *Proceedings of the 33rd International ACM Conference on Research and Development in Information Retrieval*, pages 787–788. ACM, 2010.

[47] Miles Efron. Information search and retrieval in microblogs. *Journal of the American Society for Information Science and Technology*, 62(6):996–1008, 2011.

[48] Miles Efron, Jimmy Lin, Jiyin He, and Arjen de Vries. Temporal feedback for tweet search with non-parametric density estimation. In *Proceedings of the 37th international ACM Conference on Research and Development in Information Retrieval*, pages 33–42. ACM, 2014.

[49] Carsten Eickhoff, Christopher G. Harris, Arjen P. de Vries, and Padmini Srinivasan. Quality through flow and immersion: Gamifying crowdsourced relevance assessments. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 871–880, 2012.

[50] Jacob Eisenstein. What to do about bad language on the internet. In *Proceedings of the 12th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, 2013.

[51] Yue Fei, Yihong Hong, and Jianwu Yang. Handling topic drift for topic tracking in microblogs. In *Advances in Information Retrieval*, pages 477–488. Springer, 2015.

[52] Paul Ferguson, Neil OHare, James Lanagan, Alan F Smeaton, Owen Phelan, Kevin McCarthy, and Barry Smyth. Clarity at the trec 2011 microblog track. In *Proceedings of the 20th Text REtrieval Conference*, 2011.

[53] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[54] Jon Fiscus, George Doddington, John Garofolo, and Alvin Martin. Nists 1998 topic detection and tracking evaluation (tdt2). In *Proceedings of the 1999 DARPA Broadcast News Workshop*, pages 19–24, 1999.

[55] Jonathan G Fiscus and George R Doddington. Topic detection and tracking evaluation overview. In *Topic detection and tracking*, pages 17–31. Springer, 2002.

[56] PR Freeman. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, pages 189–206, 1983.

[57] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[58] Catherine Grady and Matthew Lease. Crowdsourcing document relevance assessment with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 172–179. Association for Computational Linguistics, 2010.

[59] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 368–378. Association for Computational Linguistics, 2011.

[60] Bo Han, Paul Cook, and Timothy Baldwin. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics, 2012.

[61] Zhongyuan Han, Xuwei Li, Muyun Yang, Haoliang Qi, Sheng Li, and Tiejun Zhao. Hit at trec 2012 microblog track. In *Proceedings of the 21th Text REtrieval Conference*, 2012.

[62] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st International Conference on World Wide Web*, pages 769–778. ACM, 2012.

[63] Yihong Hong, Yue Fei, and Jianwu Yang. Exploiting topic tracking in real-time tweet streams. In *Proceedings of the 2013 International Workshop on Mining Unstructured Big Data Using Natural Language Processing*, pages 31–38. ACM, 2013.

[64] Botao Hu, Yuchen Zhang, Weizhu Chen, Gang Wang, and Qiang Yang. Characterizing search intent diversity into click models. In *Proceedings of the 20th International World Wide Web Conference*, pages 17–26, 2011.

[65] Bernard J. Jansen and Marc Resnick. An examination of searcher's perceptions of nonsponsored and sponsored links during ecommerce web searching. *Journal of the American Society for Information Science and Technology*, 57(14):1949–1961, 2006.

[66] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[67] Eric C. Jensen, Steven M. Beitzel, Abdur Chowdhury, and Ophir Frieder. Repeatable evaluation of search services in dynamic environments. *ACM Transactions on Information Systems*, 26(1), November 2007.

[68] Jiepu Jiang and James Allan. Necessary and frequent terms in queries. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1167–1170. ACM, 2014.

[69] Yun Jin, Sung Hyon Myaeng, and Yuchul Jung. Use of place information for improved event tracking. *Information Processing & Management*, 43(2):365 – 378, 2007. Special issue on AIRS2005: Information Retrieval Research in Asia.

[70] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002.

[71] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161, 2005.

[72] Max Kaufmann and Jugal Kalita. Syntactic normalization of twitter messages. In *Proceedings of International Conference on Natural Language Processing*, 2010.

[73] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, 16(2):138–178, 2013.

[74] Adam Kilgarriff. Comparing corpora. *International Journal of Corpus Linguistics*, 6(1):97–133, 2001.

[75] Pyung Kim and Sung Hyon Myaeng. Usefulness of temporal information automatically extracted from news articles for topic tracking. *ACM Transactions on Asian Language Information Processing*, pages 227–242, 2004.

[76] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics, 2010.

[77] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[78] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 630–631. Society for Industrial and Applied Mathematics, 2005.

[79] Alexander Kotov, Vineeth Rakesh, Eugene Agichtein, and Chandan K Reddy. Geographical latent variable models for microblog retrieval. In *Advances in Information Retrieval*, pages 635–647. Springer, 2015.

[80] Alexander Kotov, Yu Wang, and Eugene Agichtein. Leveraging geographical metadata to improve search over social media. In *Proceedings of the 22nd International Conference on World Wide Web Companion*, pages 151–152. International World Wide Web Conferences Steering Committee, 2013.

[81] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[82] Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th International World Wide Web Conference*, pages 571–580, 2010.

[83] Giridhar Kumaran and James Allan. A case for shorter queries, and helping users create them. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 220–227, 2007.

[84] Giridhar Kumaran and Vitor R Carvalho. Reducing long queries using query quality predictors. In *Proceedings of the 32nd International ACM Conference on Research and Development in Information Retrieval*, pages 564–571. ACM, 2009.

[85] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, pages 591–600. ACM, 2010.

[86] Leah S. Larkey, Fangfang Feng, Margaret Connell, and Victor Lavrenko. Language-specific models in multilingual topic tracking. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 402–409, New York, NY, USA, 2004. ACM.

[87] Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas. Relevance models for topic detection and tracking. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 115–121. Morgan Kaufmann Publishers Inc., 2002.

[88] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM Conference on Research and Development in Information Retrieval*, pages 120–127. ACM, 2001.

[89] Matthew Lease, James Allan, and W Bruce Croft. Regression rank: Learning to meet the opportunity of descriptive queries. In *Advances in Information Retrieval*, pages 90–101. Springer, 2009.

[90] Baoli Li, Wenjie Li, and Qin Lu. Enhancing topic tracking with temporal information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 667–668, New York, NY, USA, 2006. ACM.

[91] Changliang Li, Bo Xu, Gaowei Wu, Xiuying Wang, Wendong Ge, and Yan Li. Obtaining better word representations via language transfer. In *Computational Linguistics and Intelligent Text Processing*, pages 128–137. Springer, 2014.

[92] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 165–174. ACM, 2016.

[93] Yan Li, Zhenhua Zhang, Wenlong Lv, Qianlong Xie, Yuhang Lin, Rao Xu, Weiran Xu, Guang Chen, and Jun Guo. Pris at trec 2011 microblog track. In *Proceedings of the 20th Text REtrieval Conference*, 2011.

[94] Nut Limsopatham, Richard McCreadie, M Albakour, Craig Macdonald, Rodrygo L Santos, Iadh Ounis, et al. University of glasgow at trec 2012: Experiments with terrier in medical records, microblog, and web tracks. In *Proceedings of the 21th Text REtrieval Conference*, 2012.

[95] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[96] Jimmy Lin and Miles Efron. Overview of the trec-2013 microblog track. In *Proceedings of the 22th Text REtrieval Conference*, volume 2013, 2013.

[97] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. Overview of the trec-2014 microblog track. In *Proceedings of the 23th Text REtrieval Conference*, 2014.

[98] Jimmy Lin, Miles Efron, Yulu Wang, Garrick Sherman, and Ellen Voorhees. Overview of the trec-2015 microblog track. Technical report, DTIC Document, 2014.

[99] Jimmy Lin, Rion Snow, and William Morgan. Smoothing techniques for adaptive online language models: Topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 422–429, New York, NY, USA, 2011. ACM.

[100] Lu Liu, Lifeng Sun, Yong Rui, Yao Shi, and Shiqiang Yang. Web video topic discovery and tracking via bipartite graph reinforcement model. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1009–1018, New York, NY, USA, 2008. ACM.

[101] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 359–367. Association for Computational Linguistics, 2011.

[102] Samuel Louvan, Mochamad Ibrahim, Mirna Adriani, Clara Vania, Bayu Distiawan, and Metti Z Wanagiri. University of indonesia at trec 2011 microblog track. In *Proceedings of the 20th Text REtrieval Conference*, 2011.

[103] Marco Lui, Ned Letcher, Oliver Adams, Long Duong, Paul Cook, Timothy Baldwin, and NICTA Victoria. Exploring methods and resources for discriminating similar languages. In *Proceedings of the 25th International Conference on Computational Linguistics*, page 129, 2014.

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

[105] Zongyang Ma, Aixin Sun, and Gao Cong. On predicting the popularity of newly emerging hashtags in twitter. *American Society for Information Science and Technology*, pages 1399–1410, 2013.

[106] Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. Tagging your tweets: A probabilistic modeling of hashtag annotation in twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 999–1008. ACM, 2014.

[107] Walid Magdy and Tamer Elsayed. Adaptive method for following dynamic topics on twitter. In *Proceedings of the 8th International Conference on Weblogs and Social Media*, 2014.

[108] Juan Martinez-Romo and Lourdes Araujo. Updating broken web links: An automatic recommendation system. *Information Processing and Management*, 48(2):183–203, 2012.

[109] Massimo Melucci. Weighted rank correlation in information retrieval evaluation. In *Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology*, pages 75–86, 2009.

[110] Donald Metzler and Congxing Cai. Usc/isi at trec 2011: Microblog track. In *Proceedings of the 20th Text REtrieval Conference*. Citeseer, 2011.

[111] Donald Metzler and W Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM Conference on Research and Development in Information Retrieval*, pages 472–479. ACM, 2005.

[112] Jun Miao, Jimmy Xiangji Huang, and Jiashu Zhao. Topprf: A probabilistic framework for integrating topic space into pseudo relevance feedback. *ACM Transactions on Information Systems (TOIS)*, 34(4):22, 2016.

[113] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing*, pages 404–411. Association for Computational Linguistics, 2004.

[114] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[115] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[116] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 12th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[117] Alistair Moffat, Falk Scholer, and Paul Thomas. Models and metrics: IR evaluation as a user process. In *Proceedings of the 17th Australasian Document Computing Symposium*, pages 47–54, 2012.

[118] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, 27(1):2:1–2:27, December 2008.

[119] Masaki Mori, Takao Miura, and Isamu Shioya. Topic detection and tracking for news web pages. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '06, pages 338–342, Washington, DC, USA, 2006. IEEE Computer Society.

[120] Marc A. Najork. Comparing the effectiveness of HITS and SALSA. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 157–164, 2007.

[121] Ramesh Nallapati, James Allan, and Sridhar Mahadevan. Extraction of key words from news stories. Technical report, DTIC Document, 2004.

[122] Kyosuke Nishida, Takahide Hoshide, and Ko Fujimura. Improving tweet stream classification by detecting changes in word probability. In *Proceedings of the 35th international ACM Conference on Research and Development in Information Retrieval*, pages 971–980. ACM, 2012.

[123] Iadh Ounis, Craig Macdonald, Jimmy Lin, and Ian Soboroff. Overview of the trec-2011 microblog track. In *Proceedings of the 20th Text REtrieval Conference*, 2011.

[124] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. Technical report, Stanford University, 1999.

[125] Jiaul H Paik and Jimmy Lin. Do multiple listeners to the public twitter sample stream receive the same tweets? In *Proceedings of the SIGIR 2015 Workshop on Temporal, Social and Spatially-Aware Information Access*, 2015.

[126] Swit Phuvipadawat and Tsuyoshi Murata. Breaking news detection and tracking in twitter. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pages 120–123. IEEE, 2010.

[127] Paul Rayson and Roger Garside. Comparing corpora using frequency profiling. In *Proceedings of the Workshop on Comparing Corpor*, WCC ’00, pages 1–6, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[128] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, July 2009.

[129] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

[130] Stephen Robertson. On GMAP: and other transformations. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 78–83, 2006.

[131] Stephen E Robertson and Ian Soboroff. The trec 2002 filtering track report. In *Proceedings of the 11th Text REtrieval Conference*, 2002.

[132] Dmitri Roussinov, Michael Chau, Elena Filatova, and José Antonio Robles-Flores. Building on redundancy: Factoid question answering and the “other”. In *Proceedings of the 14th Text REtrieval Conference*, 2005.

[133] Tetsuya Sakai and Zhicheng Dou. Summaries, ranked retrieval and sessions: A unified framework for information access evaluation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 473–482, 2013.

[134] Tetsuya Sakai and Noriko Kando. On information retrieval metrics designed for evaluation with incomplete relevance assessments. *Information Retrieval*, 11(5):447–470, October 2008.

[135] Tetsuya Sakai and Ruihua Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1043–1052, 2011.

[136] Mark D Smucker and James Allan. Find-similar: similarity browsing as a search tool. In *Proceedings of the 29th Annual International ACM Conference on Research and Development in Information Retrieval*, pages 461–468. ACM, 2006.

[137] Mark D. Smucker and Charles L. A. Clarke. Stochastic simulation of time-biased gain. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2040–2044, 2012.

[138] Mark D. Smucker and Charles L.A. Clarke. Time-based calibration of effectiveness measures. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in information retrieval*, pages 95–104, 2012.

[139] Mark D. Smucker and Chandra Prakash Jethani. Human performance and retrieval precision revisited. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602, 2010.

[140] Ian Soboroff, Iadh Ounis, J Lin, and I Soboroff. Overview of the trec-2012 microblog track. In *Proceedings of the 21th Text REtrieval Conference*, volume 2012, 2012.

[141] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd International ACM Conference on Research and Development in Information Retrieval*, pages 841–842. ACM, 2010.

[142] Ilija Subašić and Bettina Berendt. Peddling or creating? investigating the role of twitter in news reporting. In *Advances in Information Retrieval*, pages 207–213. Springer, 2011.

[143] Mingxuan Sun, Guy Lebanon, and Kevyn Collins-Thompson. Visualizing differences in web search algorithms using the expected weighted Hoeffding distance. In *Proceedings of the 19th International World Wide Web Conference*, pages 931–940, 2010.

[144] Luchen Tan and Charles L. A. Clarke. Succinct queries for linking and tracking news in social media. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1883–1886, 2014.

[145] Luchen Tan and Charles L. A. Clarke. A family of rank similarity measures based on maximized effectiveness difference. *IEEE Transactions on Knowledge and Data Engineering*, pages 2865–2877, 2015.

[146] Luchen Tan, Adam Roegiest, and Charles L. A. Clarke. University of Waterloo at TREC 2015 Microblog Track. In *Proceedings of The 24h Text REtrieval Conference*, 2015.

[147] Luchen Tan, Haotian Zhang, Charles L. A. Clarke, and Mark D. Smucker. Lexical comparison between wikipedia and twitter corpora by using word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 657–661, 2015.

[148] Yi-jie Tang, Chang-Ye Li, and Hsin-Hsi Chen. A comparison between microblog corpus and balanced corpus from linguistic and sentimental perspectives. In *Proceedings of the 5th AAAI Conference on Analyzing Microtext*, pages 68–73, 2011.

[149] Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. # twittersearch: a comparison of microblog search and web search. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 35–44. ACM, 2011.

[150] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 33–40. Association for Computational Linguistics, 2003.

[151] Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. Linking online news and social media. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 565–574. ACM, 2011.

[152] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[153] Karin Verspoor, K Bretonnel Cohen, and Lawrence Hunter. The textual characteristics of traditional and open access scientific journals are similar. *BMC Bioinformatics*, 10(1):183, 2009.

[154] Ellen M. Voorhees. Overview of the TREC 2005 robust retrieval track. In *Proceedings of the 14th Text REtrieval Conference*, 2005.

[155] 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 Conference on Research and Development in Information Retrieval*, pages 615–624. ACM, 2015.

[156] Charles L. Wayne. Multilingual topic detection and tracking: Successful research enabled by corpora and evaluation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, 2000.

[157] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):20, 2010.

[158] W John Wilbur and Leona Coffee. The effectiveness of document neighboring in search enhancement. *Information Processing and Management*, 30(2):253–266, 1994.

[159] Xiaobing Xue, Samuel Huston, and W Bruce Croft. Improving verbose queries using subset distribution. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1059–1068. ACM, 2010.

[160] J Yamron, I Carp, L Gillick, S Lowe, and P Van Mulbregt. Topic tracking in a news stream. In *Proceedings of DARPA Broadcast News Workshop*, pages 133–136, 1999.

[161] Jonathan P Yamron, Ira Carp, Lawrence Gillick, Stewe Lowe, and Paul van Mulbregt. A hidden markov model approach to text segmentation and event tracking. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal*, volume 1, pages 333–336. IEEE, 1998.

[162] JP Yamron, S Knecht, and P Van Mulbregt. Dragons tracking and detection systems for the tdt2000 evaluation. In *Proceedings of Topic Detection and Tracking Workshop*, pages 75–80. Citeseer, 2000.

[163] Yiming Yang, Tom Ault, Thomas Pierce, and Charles W Lattimer. Improving text categorization methods for event tracking. In *Proceedings of the 23rd Annual International ACM Conference on Research and Development in Information Retrieval*, pages 65–72. ACM, 2000.

[164] Yiming Yang, Jaime G Carbonell, Ralf D Brown, Thomas Pierce, Brian T Archibald, and Xin Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, pages 32–43, 1999.

[165] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. Query by document. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pages 34–43. ACM, 2009.

[166] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in information retrieval*, pages 587–594, 2008.

[167] Emine Yilmaz, Evangelos Kanoulas, and Javed A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 603–610, 2008.

[168] Hong Yu, Zhang Yu, T Liu, and S Li. Topic detection and tracking review. *Journal of Chinese Information Processing*, 6(21):77–79, 2007.

[169] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. Who, where, when and what: discover spatio-temporal topics for twitter users. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 605–613, 2013.

[170] Quan Yuan, Gao Cong, Kaiqi Zhao, Zongyang Ma, and Aixin Sun. Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users. *ACM Transactions on Information Systems (TOIS) - Special Issue on Contextual Search and Recommendation*, 33(1):2:1–2:33, 2015.

[171] Jiayue Zhang, Sijia Chen, Yue Liu, Jie Yin, Qianqian Wang, Weiran Xu, and Jun Guo. Pris at 2012 microblog track. In *Proceedings of the 21th Text REtrieval Conference*, 2012.

[172] Kai Zhao, Hany Hassan, and Michael Auli. Learning translation models from monolingual continuous representations. In *Proceedings of 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[173] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. Topical keyphrase extraction from twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 379–388. Association for Computational Linguistics, 2011.

[174] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval*, pages 338–349. Springer, 2011.

[175] Xiaoqi Zhao and Keishi Tajima. Online retweet recommendation with item count limits. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pages 282–289. IEEE Computer Society, 2014.

[176] Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang, and Tingting He. Bi-transferring deep neural networks for domain adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. ACL, 2016.

[177] Guangyou Zhou, Zhao Zeng, Jimmy Xiangji Huang, and Tingting He. Transfer learning for cross-lingual sentiment classification with weakly shared deep neural networks. In *Proceedings of the 39th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 245–254. ACM, 2016.

[178] Bolong Zhu, Jinghua Gao, Xiao Han, Cunhui Shi, Shenghua Liu, Yue Liu, and Xueqi Cheng. Ictnet at microblog track trec 2012. In *Proceedings of the 21th Text REtrieval Conference*, 2012.