QATAR UNIVERSITY

COLLEGE OF ENGINEERING

ON RELEVANCE FILTERING FOR REAL-TIME TWEET SUMMARIZATION

BY

REEM ALI SUWAILEH

A Thesis Submitted to the Faculty of

the College of Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Masters of Science in Computing

June 2018

COMMITTEE PAGE

The members of the Committee approve the Thesis of Reem Ali Suwaileh

defended on 05/06/2018.

_____

Dr. Tamer Elsayed
Thesis/Dissertation Supervisor

_____

Dr. Abdelkarim Erradi
Committee Member

_____

Prof. Fazli Can
Committee Member

Approved:

_____

Khalifa Al-Khalifa, Dean, College of Engineering

ABSTRACT

SUWAILEH, REEM, ALI. Masters : June : 2018:, Masters of Science in Computing

Title: On Relevance Filtering for Real-time Tweet Summarization

Supervisor of Thesis: Tamer, Elsayed.

Real-time tweet summarization systems (RTS) require mechanisms for capturing relevant tweets, identifying novel tweets, and capturing timely tweets. In this thesis, we tackle the RTS problem with a main focus on the relevance filtering. We experimented with different traditional retrieval models.

Additionally, we propose two extensions to alleviate the sparsity and topic drift challenges that affect the relevance filtering. For the sparsity, we propose leveraging word embeddings in Vector Space model (VSM) term weighting to empower the system to use semantic similarity alongside the lexical matching. To mitigate the effect of topic drift, we exploit explicit relevance feedback to enhance profile representation to cope with its development in the stream over time.

We conducted extensive experiments over three standard English TREC test collections that were built specifically for RTS. Although the extensions do not generally exhibit better performance, they are comparable to the baselines used.

Moreover, we extended an event detection Arabic tweets test collection, called EveTAR, to support tasks that require novelty in the system's output. We collected novelty judgments using in-house annotators and used the collection to test our RTS system. We report preliminary results on EveTAR using different models of the RTS system.

DEDICATION

To all children in Yemen, Syria, Palestine and the Islamic world; you are my only inspiration when I lose my courage and faith. To all who believe in failure; failure does not exist!

.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Almighty Allah Who granted me the opportunity to work on this thesis (Alhamdulillah), guided me throughout my life and blessed me abundantly with notable achievements.

I extend my heartiest gratitude to my parents for their unconditional love, passionate encouragement, and continuous supplication to Allah. It gives me immense pleasure to thank my sister Aisha, for everything she did for me, my brother Mohamed, who was always available whenever I needed him, Nabil who has always been a source of inspiration for me, Salah (and his family) who is my model for success, Wafa who is my model of determination, and Yousuf who is my source of happiness.

I would like to submit my sincere and deepest gratitude to my supervisor, Dr. Tamer Elsayed. This thesis would not have accomplished success without his meticulous supervision, admirable patience, and constant support. I would not be who I am today without having the honor of being mentored by him. I am deeply indebted to him for his highly commendable efforts during my journey in research.

I am also grateful to the bigIR group, for their persistent support and providing an encouraging research environment. I thank Dr. Mucahid, Maram, Rahma, Yassmine, Mrs. Rana, Khaled, and all other members for their advice, prayers, flowers, sweets, and coffee at the right time.

I would also like to thank all my friends, who have never let me down, for their love, support, gifts, and adventures. Special thanks go to Faiza, Linah, Wassima, Nada, Sara, Rahma, Tooba, Zeineb, among others.

Finally, this work was made possible by NPRP grants # NPRP 7-1313-1-245 and # NPRP 7-1330-2-483 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Twitter, as a social media platform with a diversified-content and a critical mass user base, became as it advertises itself[1], a place to break news, to check out and join online discussions on hot topics, as well as to share everyday interests. These key characteristics make Twitter stream a major source of information, not only for normal users but also for national and international agencies such as news, relief, and work agencies, and many others.

Nevertheless, the rate and load in which the information is shared over the stream with a huge amount of noise (e.g., spam content) are major challenges for users who want to stay updated on their topics of interest. Manual methods are impractical and expensive to use for tracking topics of interest over Twitter stream. This makes the need for automatic methods that keep the user up-to-date on their interest, without overwhelming them with unrelated or redundant information, a crucial demand. The automatic systems are expected to track topics (millions) on Twitter continuous stream in real-time in parallel and filter out relevant tweets.

1.1.Real-time Tweet Summarization

In this thesis, we tackle the problem of Real-time Tweet Summarization (RTS). Given a set of interest profiles that represent users' topics of interest, the RTS system has to track these in real-time over Twitter stream, capture on-topic tweets, and filter out redundant and outdated tweets before it pushes updates to users. For instance, given a user interest on discussions about "Opinions on Al Jazeera media network", the system should

[1] play.google.com/store/apps/details?id=com.twitter.android&hl=en

detect all possible on-topic tweets including all aspects of the topic such as the public and governments' opinions, boycott and protests against Al Jazeera, legal actions or movements, etc.

Most importantly, the RTS system should take into account other objectives besides the relevance, namely the novelty and freshness of filtered tweets. Informally, to satisfy the user, the system has to filter out the redundant and outdated information and keep the on-topic summary of tweets concise and light. Figure 1 illustrates the real-time summarization task visually.

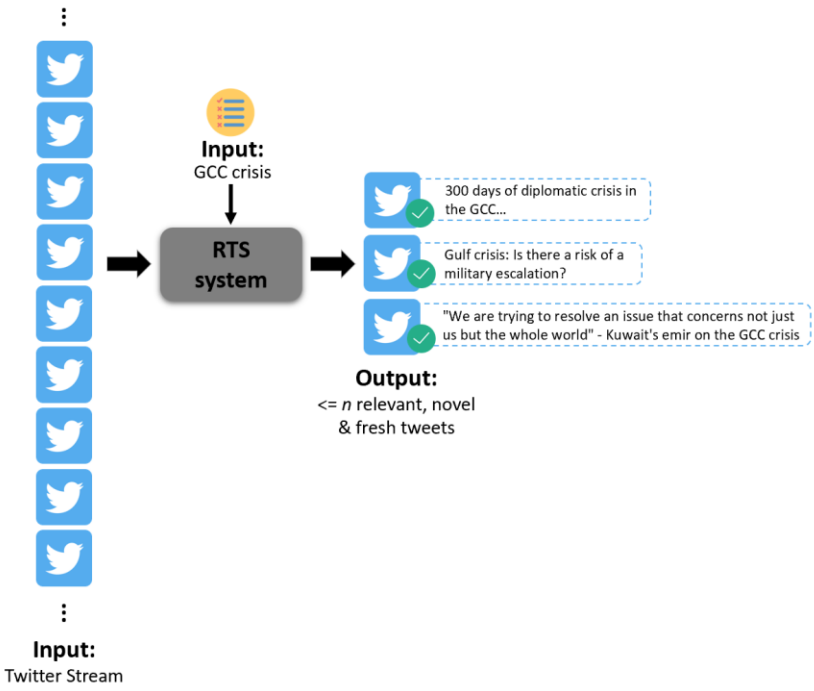

Figure 1. Real-time Tweet Summarization Task

Many challenges emerge from the nature of tweets and topics discussed on Twitter. For the tweets, the shortness of tweets has been among the challenges that Information Retrieval systems have to deal with to improve their performance. This causes the so-called "mismatch" problem in which it is difficult to identify the relevant tweets to a topic. This is mainly due to the poor representation of the topic used by the system and the difficulty to understand the natural language of the tweets. In other words, this challenge originates from the fact that different users tend to express their needs and thoughts in a different vocabulary. Thus, the retrieval models need to go beyond measuring the lexical similarity, to capture the semantic similarity between different texts to alleviate the mismatch problem.

As for the topics, the nature of a topic in terms of lifetime, difficulty, and popularity introduces different challenges to the RTS system. More specifically, topics might be discussed over hours, days, or even years and develop rapidly while they are still of interest to users. For instance, if a user is interested in following tweets about a football match, this topic would span over few hours before and after the match (or perhaps one day before and one day after depending on the popularity of the teams). Then the RTS system has to follow the topic for short period and capture the tiny portion of relevant tweets including the whole aspects of the topic (e.g., goals, results, and standings).

On the other hand, if the user is interested in the "GCC crises"[2]. Every day there are multiple subtopics and trending hashtags related to the blockade including official statements, interviews and new claims against Qatar, human rights violations, and hacking

[2] Started 5th June 2017 and still ongoing

of Qatari websites, to name a few. Hence, the RTS system has to be adaptive enough to cope with topics change over time and to satisfy the user need.

Last but not least, the RTS system should avoid complicated models and favor simple and efficient approaches that can scale to follow millions of interest profiles in parallel over the huge stream of tweets and suggest a concise and timely update to users.

In this thesis, we tackle the real-time tweet summarization problem with the aim of improving the relevance filtering in the RTS system. Specifically, we aim at mitigating two challenges related to the capturing relevant tweets to the predefined topics of interest: (1) the mismatch problem. (2) Topic development over time. Next, we list the research questions that we aim at studying through this thesis to tackle these challenges.

1.2.Research Questions

In this thesis, we address the following questions for the relevance filtering of the real-time tweet summarization system:

1. How effective are traditional different retrieval models (e.g., BM25, KL-Divergence) when we use them in relevance filtering for real-time summarization?
2. How effective are different ways of incorporating word embedding to represent the text (profiles and tweets) for different retrieval models?
3. How effective are different ways of exploiting explicit relevance feedback?
4. How effective is RTS over Arabic stream?

For the first three research questions, we consider answering them over different evaluation measures and test collections.

1.3.Solution Overview

In this thesis, we extend our RTS system that was originally intended for participation in Real-time Tweet Summarization track in TREC, specifically the mobile push notification task (scenario A).

Our main objective is to improve the relevance filtering in the RTS system. To achieve this, we have implemented three extensions over the relevance filter of the system. We first implemented three different traditional models, namely: Vector Space Model (VSM) (e.g., different terms weighting such as TFIDF), Probabilistic Model (e.g., BM25), and Language Model (e.g., Kullback–Leibler divergence). The other two extensions aim at combating tweets brevity and topic drift problems on the relevance filtering. For that, we extended the system using two ways: (1) leveraging word embeddings to semantically match tweets to profiles, and (2) exploiting relevance feedback to enhance the topic representation. In all extensions, the system only uses textual and temporal features of a tweet and ignores other features such as social signals (e.g., retweets, likes, etc.).

1.4.Contributions

In this thesis, we have three major contributions:

1. Participated (as a member of QU team) in Real-time Tweet Summarization track in three years in a row. In 2016, out system was ranked first among 19 international teams.
2. Proposed different extensions to the relevance filtering component of the core system:
    a. Adopting different traditional retrieval models.

b. Leveraging distributed word representation in vector space model (VSM).
c. Exploiting relevance feedback for profile expansion.

3. Extended *EveTAR* Arabic tweet test collection by collecting novelty judgments to enable the evaluation of RTS systems. We conducted preliminary experiments on the new test collection. This thesis reports the first RTS results on Arabic tweet stream.

We organize the remainder of the thesis as follows. We review the literature for non-microblog and microblog summarization approaches in Chapter 2. We layout the system architecture and discuss our extensions on the relevance filter of the RTS system in Chapter 3. We discuss the evaluation setup and our results in Chapter 4. Finally, we conclude and present possible future work in Chapter 5.

CHAPTER 2: RELATED WORK

In this chapter, we review the related work to tweet summarization task. In general, tweet summarization task aims at capturing on-topic tweets to compose a summary that satisfies the end-users' need, however, the task definition (in terms of objectives and type of topics) vary in literature. We organize this chapter into three main sections: (a) general applications of tweets summarization (Section 2.1.), (2) push notification systems (Section 2.2.), and Arabic tweet summarization (2.3.). Note that all approaches reviewed in the first two sections are evaluated over English tweet streams.

2.1.Topic Tracking over Twitter

Mackie et al. [23] have compared the effectiveness of eleven traditional summarization approaches used in newswire summarization including temporal, term statistics, and comprehensive approaches, to name a few. They conducted their experiments over four tweet test collections and evaluated the algorithms using ROUGE and *SIMetrix* measures. They found that the *SumBasic* and Centroid-based approaches outperform all other approaches.

Xu et al. [44] have proposed a graph-based approach to generate summaries for events of interest. Using events' information extracted from Twitter (e.g., named entities), they constructed an event graph to represent the relations between aspects of events. They apply a Pagerank-like algorithm to rank these aspects and partition the graph to detect fine-grained updates on the event. The approach was evaluated using human evaluation and highlighted the importance of standardizing the evaluation of tweet summarization task. Thus, it is hard to compare their approach to other approaches.

Shou et al. [35] proposed an approach that employs incremental clustering for continuous online tweets summarization. They evaluated the effectiveness (F1 measure) and the efficiency of their approach on a large scale. As their setup is different than the experimental setup used in TREC, it is difficult to judge the performance of their approach. Additionally, Olariu [29] has also evaluated the efficiency of his approach. He presented a bigram graph-based technique for stream summarization. The approach was able to generate abstractive summaries. Moreover, it outperformed the baseline used in terms of quality and efficiency.

2.1.1. Adaptive filtering

Lin et al. [21] investigated broad topics tracking and propose an adaptive language modeling approach to mitigate the so-called topic drift problem (i.e., capturing recency). Their experiments showed that the best results were obtained when applying the stupid back-off smoothing technique.

Differently, Albakour et al. [1] tackled the sparsity and topic drift challenges over tweet streams by applying topic expansion to handle the sparsity problem. They adopted event detection techniques to detect the time at which drift might happen in addition to smoothing techniques to combat topic drift.

More recently, Fei et al. [9] adopted a news filtering approach to tracking focused topics. They trained a classifier based on Binomial Logistic Regression (LR) to filter relevant tweets from Twitter stream. They tackled the topic drift problem using a cluster-based subtopic detection algorithm and integrate it with the LR classifier. The cluster creation draws the system's attention to focus on the new subtopics while tracking the

original topic and hence handle the topic drift dynamically. Their experiments showed high performance compared to other methods.

All the above research studies do not follow the same problem definition and do not use a unified evaluation test-bed. Real-time Tweet Summarization track in TREC conference has brought the advantage of having a common evaluation framework. This enables the researchers to compare their methods fairly and advance the state-of-the-art. We review the proposed approaches in RTS track in the following section.

2.2.Push Notifications Systems

Tweets filtering task had run in four rounds in Microblog track at the Text REtrieval Conference (TREC); TREC-2012 [36], TREC-2015 [18], TREC-2016 [20], and TREC-2017 [19]. The track design had been the same in the last three years with simple variations in evaluation setting. It has two main tasks that reflect two real scenarios of filtering systems: (1) mobile push notification (scenario A), where the system is expected to push a few tweets as notifications on the user's mobile phone, and (2) email digest (scenario B), where the system sends the user a periodic email digest (daily) that contains a summary of on-topic tweets. In this work, we focus solely on scenario A.

2.2.1. Relevance filtering

Han, et al. [11] proposed different filtering models, namely: hyperlink-extended language model (LM) based, learning-to-rank based, and hybrid models. The language model-based models exploit external resources (i.e., URLs) to estimate the relevance of tweets to profiles, while the L2R model uses the scores of different similarity functions between the profile and the tweet to estimate the relevancy. The L2R model is optimized for MAP metric using Gradient Descent.

Language modeling was also employed to estimate relevancy of incoming tweets to interest profiles by Sabhnani and Carterette [33] who used document-likelihood model and Tang., J. et al. [42] used KL-divergence (in addition to cosine distance and blending models).

Differently, Buntain and Lin [7] proposed an approach to detect the peak moments of topics discussed over tweet streams to construct a summary of tweets that were posted within these moments. They used a sliding time window to study the frequency of users discussing a tracked topic and then select the peak moments to push tweets from them.

2.2.2. Adaptive filtering

In TREC-2015, many teams have designed their systems to maximize effectiveness by adaptive summarization approaches. Among these is the approached proposed by Fei et al. [22] and Luchen et al. [40]; their approach dynamically adjusts the relevance thresholds by looking at the scores of the top *n* ranked tweets in the previous day automatically [22] [40] or manually [22]. Luchen et al. [40] conducted *post-hoc* experiments on their runs [41] to investigate the effectiveness of different threshold settings: (1) static threshold, (2) dynamic threshold without feedback and (3) dynamic threshold with feedback. The third approach outperforms all others, but it did not beat the optimal threshold that could be set for each topic independently in each day.

The availability of online explicit feedback in TREC-2017, opened the door for deeper exploration on the effectiveness of dynamic thresholding mechanisms. Sabhnani., K and Carterette., B [33] initialized the relevance threshold by averaging two median relevance scores of related tweets to all profiles obtained from Twitter Search API: the median of relevance score of all its related tweets (Upper score) and the median scores of

related tweets to all other profiles (lower score). Using the explicit feedback, they dynamically updated the upper and lower bound of the relevance threshold. Suwaileh et al. [37] used the number of pushed tweets to adjust the relevancy threshold by lowering it when too many tweets are pushed and vice versa.

2.2.3. Profile Expansion

To remedy the effect of tweets' brevity on relevance scoring, participants used Pseudo-Relevance Feedback technique to enhance both topics and tweets representations representation with IDF-cosine weighting [38] [42]. Moreover, participants attempt utilizing external resources for the same purpose. For example, [22] [48] took advantage of an external evidence to perform Web-based query expansion using Google search API. On the other hand, Sabhnani., K. and Carterette., B [33] applied profile expansion using the top k related tweets retrieved using Twitter Search API to remedy the so-called cold start problem.

2.2.4. Leveraging Word Embeddings

Distributed word embeddings were also exploited in different manners in push notification task. Zhu et al. [48] used the semantic and a quality features of tweets to capture the potentially-relevant tweets. Unlike our proposed approach that integrates the similarities of distributed word representation and vector space models, they completely relied on word embeddings representation for semantic relevance and trained a logistic regression model using the quality features (e.g., number of retweets, hashtags, URLs, meaningful words, etc.) to predict quality scores of a tweet. They ranked tweets after combining the semantic and quality scores.

Moulahi et al. [28] adapted the extended AND-Boolean relevance model in their solution and replaced the TF-IDF weighting method to represent profiles and tweets in the vector space by the word2vec model. Bagdouri and Oard [5] trained a word2vec model [26] using a Twitter corpus covering around four years to perform profile expansion using the stems of the title field of the interest profile.

2.2.5. Novelty Filtering

Thus far, we presented the literature focusing on relevance filtering of the task evaluation, now we focus on novelty filtering. Fei et al. [22] implemented a greedy approach to select the top relevant tweet after deduplicating similar tweets to the previously pushed tweets. Fan et al. [8] introduced a hierarchical learning model (HTM) that adaptively learns the distributed word representations of tweets to deduplicate semantically redundant tweets. They used TREC test collections in their experiments and showed the effectiveness of their approaches. Sabhnani., K., and Carterette., B [33] used the efficient and dynamic Quality Threshold (QT) clustering algorithm. Jaccard similarity was also used for novelty filtering [4][7][39].

2.3.Arabic Tweets Summarization

At the other end of the spectrum, a real-time tweet summarization task was studied over Arabic tweets. Magdy and Elsayed [24] proposed an adaptive filtering approach to track Arabic *broad and dynamic* topics that have a variety of subtopics (e.g., Yemen) on Arabic tweets. Their approach targets a high recall (while maintaining a good level of precision) to capture as many relevant tweets as possible to cover as many subtopics as possible.

Additionally, Alsaedi et al. [3] proposed three approaches that consider the temporal factor for *real-world event* summarization on Arabic tweets: Temporal TF-IDF, Retweet Voting (involves social factor), and Temporal Centroid Representation methods. The system uses a sliding time window to weight tweets and composes summaries using the top weighted tweets for each window. Using ROUGE-1 measure, they showed that the approaches that do not consider the social factor outperform two of the leader summarization approaches [6] [13]. Unlike our problem definition, this work targets capturing high-quality, relevant and useful tweets without a direct focus (and evaluation) on the novelty of summary that is a substantial goal in our task definition.

CHAPTER 3: APPROACH

Information needs and would like to stay up-to-date on their interests, the RTS task can be perceived as a recommendation task in which a system is required to automatically monitors a continuous stream of tweets (e.g., twitter stream) and captures tweets of interest to users.

To represent the user needs, the RTS system uses a set of interest profiles (called topics in TREC jargon)3. We adopt TREC-style profile representation that is composed of four main fields: (1) id: a unique identifier of the profile, (2) title: a short sentence of the topic, (3) description: at most a couple of sentences that put into words the information needs of the user, and (4) narrative: a paragraph of 3 or more sentences that detail the information need and describe the expected information by the user. We show an example of a profile taken from trec-2015 test collection in Figure 2:

ID	MB371
Title	self-driving cars
Description	Find information on self-driving cars.
Narrative	The user is a commuter and wants to follow the development of self-driving cars. tweets that discuss any aspect of self-driving cars are relevant, including accidents, testing, research, public opinion, legal issues, technology, safety, manufacturers, etc.

Figure 2. Example interest profile is taken from TREC-2015 RTF track.

[3] We will use the terms profile and topic interchangeably throughout the thesis.

Once the interest profiles are provided to the RTS system, it tracks them in real-time over twitter live stream and identifies a potentially-relevant set of tweets to each profile (if any). Once the system manages to extract on-topic tweets, it has to assure the novelty and freshness of these tweets before it notifies the users on their mobile phones. To satisfy users' need without overwhelming them, the system is allowed to push at most 10 tweets per day. Figure 3 illustrates the definition of RTS task including the input, processing, and output.

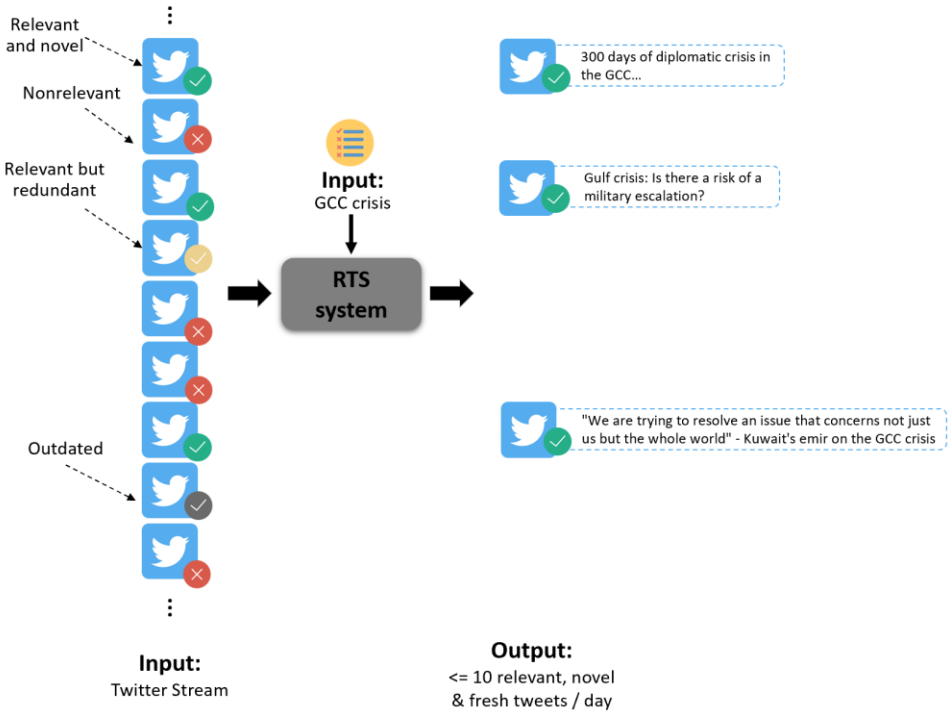

Figure 3. Real-time summarization task

In this chapter, we present our approach of solving RTS problem in Section 3.1. We layout the architecture of the core system in Section 3.2. We then describe the extensions of the relevance filter that we built over the core system aiming at answering the aforementioned research questions (Section 3.3.).

3.1.Solution Overview

RTS system typically aims at three essential objectives: (1) relevance filtering in which it captures potentially-relevant tweets, (2) novelty filtering to discard semantically redundant information, and (3) timely tweet selection to effectively push relevant, non-redundant and recent updates to users. We took into account all of these three objectives when we designed our solution as part of participation in the RTF [39] and RTS [37][38] tracks in TREC. Figure 4 depicts the design of our approach. Our approach involves a separate filter to achieve each main objective, however, this is not the optimal design as all objectives can be achieved in one component (e.g., a classifier).

Twitter stream contains a huge and diverse amount of information; where various topics are discussed with different levels of quality, focus, and redundancy. This nature of the stream is a challenge as it requires efficient filtering while tracking topics of interest.

Upon tweets arrival and before a tweet is considered for filtering, the system has to check its quality and discards low-quality (e.g., spam) tweets. This pre-qualification step empowers efficient and effective tweets filtering.

To extract on-topic tweets (relevance filtering), the system can use different attributes extracted from the tweet such as the text and the social signals. In our solution, we solely make use of tweet's text to estimate relevance using lexical similarity (i.e., text matching).

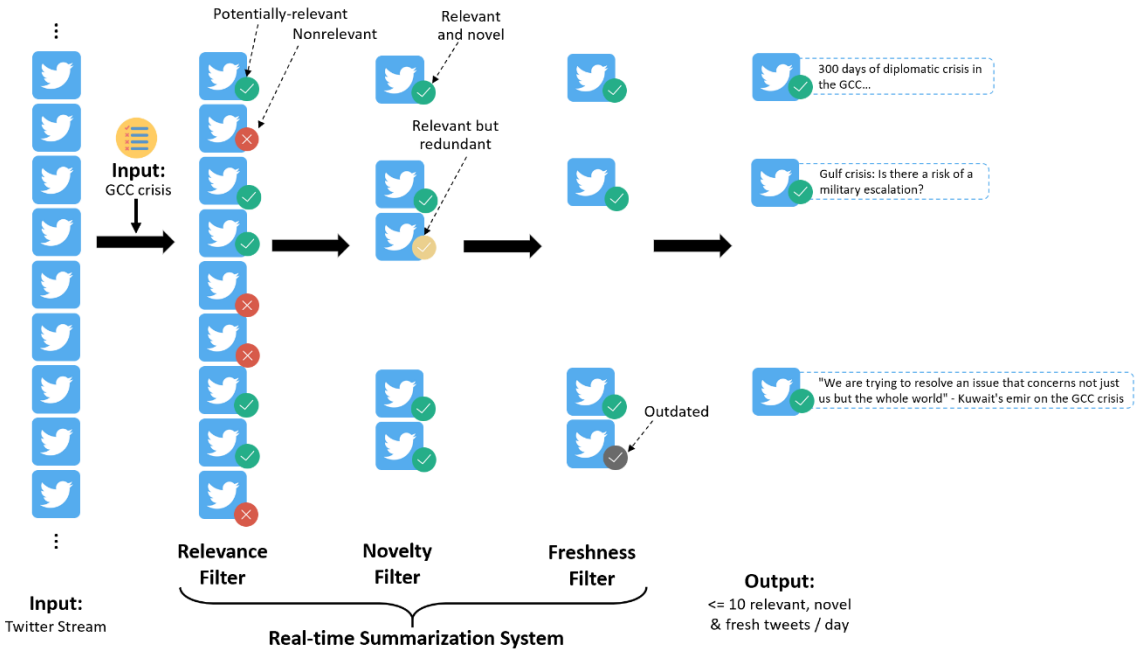

Figure 4. A high-level overview of the solution.

The system then eliminates duplicates among the set of potentially-relevant tweets. The redundancy is not limited to only exact duplicates but involves the semantically similar information conveyed by already-seen tweets. In our solution, judge the novelty of a tweet, we only consider the lexical overlap between the tweet and all already pushed tweets.

Finally, the system takes into consideration the freshness of potentially relevant and novel tweets before it decided to push any. This condition is controlled by the nomination strategy that the system follows. The system can immediately push the highly-relevant and novel tweets once identified or periodically check the potentially relevant and novel tweets and decide which tweet to push. Our system sacrifices the latency in favor of relevance and novelty and periodically pushes a tweet after ranking all relevant and novel tweets according to their relevance, novelty, and freshness.

In this thesis, we propose three extensions on the system to improve the relevance filter from two aspects. The first aspect is improving the matching of tweets against interest profiles. For this, we use distributed word representation aiming at capturing the semantic similarity, between profiles and tweets, in addition to the lexical similarity. Moreover, this system design might perform well for a static set of topics, nevertheless, the topics discussed on twitter are usually dynamic with variant lifetime. Thus, the second aspect of improvement is to let the system cope with topics' development over time by enhancing their representations on regular basis.

We next describe the architecture of the core system in depth and discuss the extension that we implemented upon the relevance filter.

3.2.Core System

In this section, we closely describe the stages that compromise the core system pipeline[4]. The system is conservative in a sense that it extensively filters out the noise and narrows down the set of candidate tweets for efficient scoring. Precisely, given a list of interest profiles (i.e., topics in traditional ad-hoc), the system tracks these profiles over twitter stream in a scalable manner and processes only the promising tweets (i.e., tweets that match at least one term of the profiles' titles) in a pipeline of multiple filters: pre-qualification, preprocessing, indexing, relevance filtering, novelty filtering, and tweets nomination.

Figure 5 gives a high-level depiction of the components of the core system architecture. For each objective of RTS task, the solution includes one or more modules

[4] Note that the design and implementation of the core system are not among the contributions of this thesis.

that aim to satisfy it.

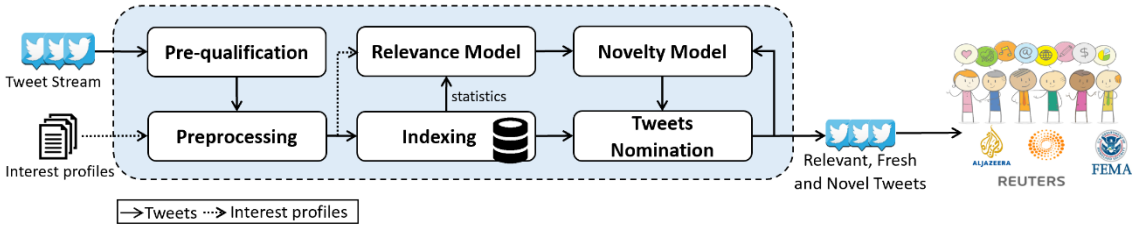

Figure 5. A high-level architecture of the core system [38].

3.2.1. Pre-qualification

While the system monitors the stream using twitter streaming API, it filters out non-English and low-quality tweets. The criteria by which we determine the quality of a tweet is based on its length and the number of hashtags and URLs it contains. Specifically, the system ignores any tweet that has less than five terms or more than one URL or more than three hashtags. Retweets are not filtered out as their underlying tweets might not be gathered from the 1% sample of twitter stream.

3.2.2. Preprocessing

Once a tweet is qualified, the system preprocesses it in a series of steps that aim at cleaning its text before scoring it for relevance and novelty. These steps include expanding the tweet with the terms appear in hashtags (i.e., after removing the '#' prefix), stemming, removing special characters (e.g., emoticons and symbolic characters), stopwords and URLs.

3.2.3. Indexing

As we acquire term statistics in the filters of the system, we initialized the system with an index of a 10-days stream of tweets prior to the beginning of the evaluation period. The system also incrementally indexes all incoming English tweets during the evaluation period.

3.2.4. Relevance Filtering

The system uses a simple vector space model (VSM) model to represent both interest profiles (represented by profiles' titles) and incoming tweets. To construct the vectors, the system computes an idf-based term weighting as follows:

$$idf(t) = log\frac{n - df(t) + 0.75}{df(t) + 0.75} \quad (1)$$

Where n is the number of tweets indexed at the time of constructing the vector, and $df(t)$ is the document frequency of the term. We chose this term weighting function due to being light-weight (which is necessary for real-time and scalable systems) and also akin to the standard tf-idf weighting function noticing that terms rarely appear more than once in a tweet due to the limited length (140 characters).

An incoming tweet is scored against a subset of matching profiles (if any) for relevance, independently, using the standard cosine similarity function as follows:

$$cos(\vec{Q}, \vec{T}) = \frac{\sum_{i\,\epsilon\,Q\cap T} q_i . t_i}{\|Q\|. \|T\|} \quad (2)$$

Where, $\|Q\|$ and $\|T\|$ are l2-norm computed as follows $\sqrt{\sum_i^q q_i^2}$.

To compute the relevance scores in real-time efficiently, the system constructs an in-memory index of profile vectors to match an incoming tweet with interest profiles. The

relevance of a tweet is determined using a relevance threshold τ_r. If the relevance score of a tweet is greater than τ_r, the system adds the tweet to the potentially-relevant tweets for the corresponding profile.

3.2.5. Dynamic Thresholding

The relevance threshold is a critical parameter in the core system. In addition to being configurable, the system also allows two options for setting it: static and dynamic. In the static threshold option, the relevance threshold τ_r is fixed at a value that is set initially. As for the dynamic threshold mode, the system starts with an initial threshold for each interest profile p_i and updates per-profile relevance threshold τ_r periodically. Contingent upon having no explicit user feedback, the system adapts itself to the topic difficulty using pseudo-relevance feedback.

Specifically, the system maintains a list of potentially-relevant tweets per profile in the last period. If the profile p_i gets no relevant tweets in the past time widow, the relevance threshold τ_r is decreased by 0.025 with a lower bound of 0.5. Otherwise, the system increases the threshold using the following equation:

$$\tau'_{r_i} = \tau_{r_i} + \min(\frac{r_{p_i}}{100}, 0.15) \quad (3)$$

Where τ_{r_i} is the current threshold of profile p_i, τ'_{r_i} is the updated threshold of that profile and r_{p_i} is the number of relevant tweets filtered for profile p_i within a time period t_t. The threshold upper bound is set to 0.95.

3.2.6. Novelty Filtering

The system then measures the novelty of the potentially-relevant tweet by

computing the overlap between it and the already-pushed tweets using a modified version of Jaccard similarity:

$$J'(q,t) = \frac{|Q| \cap |T|}{\max(|Q|,|T|)} \quad (4)$$

Where q and t are the profile and the tweet term sets, and q and t are their lengths (in terms) respectively. To consider a tweet in the tweets nomination step, it must not exceed a predefined degree of overlap, i.e., a novelty threshold τ_n, with already-pushed tweets. This way the system does not overwhelm the user with redundant notifications.

3.2.7. Pseudo-Relevance Feedback

As the explicit feedback is not always available, systems compensate by identifying a set of the potentially-relevant document using their own scoring functions. Rocchio is a typical way of utilizing user feedback for expansion [32]. We adopt a similar but lighter expansion method that controls the effect of pseudo-relevant documents on the profile representations (i.e., avoid topic drift). That is mainly because we are not certain of the actual relevance of the pseudo-relevant documents.

Once the system managed to identify a set of n pseudo-relevant tweets, we weight their terms as follows:

$$w_e(t) = n_r(t) * idf(t) \quad (5)$$

Where, $w_e(t)$ is the score of the term t in the pseudo-relevant tweet set r, $n_r(t)$ indicates the number of tweets in r that contains t, and $idf(t)$ is computed using equation (1).

We select the top k terms to add to the profile representation. The pseudo-relevant tweets are retrieved from two sources: (1) the list potentially-relevant tweets (per profile)

that were identified by the relevance filter, or (2) Twitter search service[5].

To alleviate topic drift, we reset the profile representation to the initial representation (i.e., title terms) before we apply expansion, as illustrated below.

$$\vec{q}' = \vec{q} + \beta * \vec{e} \quad (6)$$

Where $\vec{e}$ is the normalized vector of the k expansion terms, and β is a parameter used to restrict the influence of expansion terms on the new topic vector.

3.2.8. Tweets Nomination

Thus far, the system identified the relevant and novel tweets. To satisfy the users' need while avoiding overwhelming them, the system has to consider pushing a maximum of 10 tweets per day per profile. As the system has to consider the freshness of candidate tweets, it should intelligently select the optimal candidate tweets to nominate to the user. The time when the system makes the decision to elect a tweet is a critical part of the RTS system. Systems can immediately push tweets when a new candidate tweet is captured or periodically push a set of tweets.

While following all interest profiles in parallel over tweets stream, the system maintains a list of candidate tweets for each of the interest profiles. The candidate tweets are the potentially relevant and novel tweets that the system identifies so far for a specific profile. For each profile, the RTS system periodically selects the next tweet to elect to the user from the candidate list through a broker [20]. This selection filter is triggered when the systems exceed a silence period δ or it has already found l candidate tweets for that profile.

[5] https://dev.twitter.com/rest/public/search

To select the best tweet to send to the user, the system re-ranks the candidate tweets while considering their relevancy and freshness using equation (7). This re-scoring linearly penalizes the tweets based on their posting time, hence favoring fresh tweets. The top tweet is then pushed to the user.

$$S(t)' = s_r(t) * \frac{100 - (CurTime - time(t))}{100} \quad (7)$$

$s_r(t)$ is the relevance score of tweet t (computed using cosine similarity as we discussed earlier), $CurTime$ is the current system time (in minutes), and $time(t)$ is the tweet creation time (in minutes).

Thus far, we discussed the modules and features of the core system. Next, in Section 3.3., we discuss the extensions on the relevance filter that we built over the core system. In Section 3.3.1. we present several classical retrieval models that we employ in the relevance filtering (e.g., probabilistic and language models). We then discuss how we incorporate word embeddings in the classical weighting functions to expand the profiles and tweets representation in Section 3.3.2. Finally, we present methods for exploiting explicit feedback to improve system performance in Section 3.3.3.

3.3.Extensions to Relevance Filtering

In this section, we discuss the extensions we implemented to enhance the relevance filter including (1) the traditional retrieval models that we implemented to study which model performs better in RTS (Section 3.3.1), (2) incorporating the word embeddings in the retrieval models (Section 3.3.2.) to use semantic similarity, and (3) finally exploiting explicit relevance feedback (Section 3.3.3.).

3.3.1. Traditional Retrieval Models

In addition to the simple VSM with idf-based term weighting function employed in the relevance filter of the core system (Section 3.2.), we explore other retrieval models such as language modeling (e.g., Kullback–Leibler (KL) divergence) and probabilistic models (e.g., bm25). In the following subsections, we describe each model closely.

Vector Space Model

In the vector space model, a document is represented as a weighted vector that aims at capturing the importance of terms that compose the text. Several term weighting functions have been proposed for text processing [34]. Among these are the tf-idf and bm25 weighting functions.

To construct the query and document vectors, we used the following weighting functions:

- *tf-idf*: the term frequency-inverse document frequency is a traditional weighting function that compromises the term importance in the represented document (occurrence) and in the collection (rarity). We use the following variant of tf-idf that uses log normalization weighing scheme:

$$tfidf(t,d) = (1 + \log(tf(t,d)) . \log(\frac{N}{df(t)}) \quad (8)$$

Herein, t is the weighted term, d is the document to be weighted (profile or tweet), and n is the number of documents in the whole document collection. $tf(t,d)$ is term frequency of weightedthe term in the document d.

- *bm25* [31]: computes the term weight as follows:

$$bm25(t,d) = \frac{tf(t,d)}{tf(t,d) + k + \left(b \cdot \frac{|D|}{avgdl}\right)} \cdot \frac{N - df(t) + k}{df(t) + k} \quad (9)$$

The variables denote the same meaning as equation 9. *d* is the length of document *d* in words, and *avgdl* is the average document length in the text collection from which documents are drawn. *k* and *b* are free parameters.

- *uniform*: in this term weighting function we treat all terms as equally-important.

$$uniform(t) = 1 \quad (10)$$

We use a combination of these functions to represent the query and document vectors and compute the similarity using cosine similarity.

Okapi BM25 Probabilistic Model

Okapi bm25 scoring function [31] is a probabilistic model that uses the collection statistics to score documents against a query. It was empirically proven to perform well in practice [47].

$$BM25(Q,D) = \sum_{i=1}^{|Q|} bm25(q_i, D) \quad (11)$$

Where $bm25(q_i, D)$ is $q_i's$ bm25 term frequency in the document d, |d| is the length of the document d in words, and avgdl is the average document length in the text collection from which documents are drawn. k and b are free parameters.

Language Modeling

To study the effectiveness of language modeling in RTS problem, we implemented the Kullback–Leibler (KL) divergence with Dirichlet smoothing [46]. KL-divergence measures the variance between two probability distributions (the query and tweet in our context). We compute *kl*-divergence as in the following equation:

$$KL - Div(Q, D) = \sum_{t \in V} p\left(t|\hat{\theta}_Q\right).log(1 + \frac{tf(t, D)}{\mu\, p(t|C)}) + log\frac{\mu}{\mu + |D|} \qquad (12)$$

Wherein, V is the vocabulary size, d is document length, and μ is the smoothing factor.

The query-likelihood $p\left(t|\hat{\theta}_Q\right)$ and the collection-likelihood $p(t|C)$ are estimated using the maximum likelihood estimation (MLE):

$$p\left(t|\hat{\theta}_Q\right) = \frac{tf(t, Q)}{|Q|}, p(t|C) = \frac{tf(t, C)}{\sum_{t' \in C} tf(t', C)} \qquad (13)$$

Where $|Q|$ is the query length and n is the document collection size.

3.3.2. Leveraging Word Embeddings

Among the challenges of processing twitter stream is the shortness of tweets which causes the vocabulary mismatch problem. To increase the likelihood of matching the tweets against profiles (to estimate relevance), one can consider the context of words to learn their semantic meaning instead of relying only on term overlap using collection statistics. To articulate the problem, consider an information need for "apple reviews". When estimating relevance by only the occurrence of topic terms in documents, this may not satisfy the user need as it is not clear if the user is interested in finding information about "apples", a type of fruits, or "apple" the company.

To alleviate the mismatch problem, we use distributed word representation of text, the so-called word embeddings representation. Word embeddings aim to represent the words in the vector space by their context and hence enable the relevance filter to go beyond measuring the lexical similarity (e.g., tf) to semantic similarity between different granularity of texts. We specifically use word2vec models due to their popularity and

effectiveness in recent literature [26]. There are two variant algorithms to train a word2vec model: continuous bag-of-words (CBOW) and skip-gram algorithms. The former predicts the word from the input context and the latter predicts the context of the input word. Informally, word2vec models give similar representations to close words that appear more frequently together in the text (similar context) than words that are rarely found close within a predefined window.

We propose a different way to represent the tweets and profiles by incorporating word embedding models with the classical term weighting functions (e.g., tf-idf and bm25). When computing the original cosine similarity between a profile and a tweet in the vector space, only the common terms between the profile and the tweet contribute to the similarity. This means a tweet that is semantically similar with zero common terms with a profile will get a score of zero using the original cosine similarity.

To make the effect of the mismatch problem less severe, we represent the tweet in relation to each profile upon arrival using the union of the tweets and profile terms. More specifically, we expand the tweet vector by profile terms that did not appear in the original text of the tweet (expansion terms). We do the same for the profile vector. We weight the original terms using the classical weighting functions (Section 3.3.1.). As for the expansion terms, we illustrate the weighting method visually in Figure 6.

Figure 6. illustration of the word embedding expanded term representation

In particular, to expand the tweet vector, we add the terms "gcc" and "crisis" to the tweet vector after preprocessing (because they did not appear in the original vector of the tweet). To weight these terms, we first estimate the term-frequency (tf) using the word embedding vectors. For instance, we fetch the word embeddings vectors of "crisis" term and the vectors of all the original tweet terms from the word2vec model. We compute the average pair-wise similarity between the "crisis" vector and all original tweet vectors as follows:

$$tf_{w2v}(t,D) = \begin{cases} tf = \frac{1}{D}\sum_{d_i \in D} \cos(\vec{d}_e, \vec{d}_i) & if\ tf\ \geq\ \epsilon \\ tf = 0 & otherwise \end{cases} \quad (14)$$

Herein, $tf_{w2v}(t,D)$ is the weight of the expansion term $\vec{d}_e$ that is added to the tweet vector from the query (profile). $\cos(\vec{d}_e, \vec{d}_i)$ is the cosine similarity between word2vec

vectors $\vec{d}_e$ and $\vec{d}_i$. |D| is the tweet length after expansion. ϵ is a control parameter that allows assigning weights to query terms that are highly similar to the tweet and reduce the contribution of unrelated terms by assigning them zero weights.

We then substitute the estimated $tf_{w2v}(t, D)$ in the original weighting functions, bm25 and tf-idf. Another weighting function that we use is the maximum pair-wise similarity that we compute as follows:

$$tf_{w2v}(t, D) = \begin{cases} tf = \frac{1}{D}\, max_{d_i \in D}[\cos(\vec{t}, \vec{d_i})] & if\ tf\ \geq\ \epsilon \\ tf = 0 & otherwise \end{cases} \quad (15)$$

3.3.3. Exploiting Relevance Feedback

Among the conventional IR challenges is the synonymy problem where different words might refer to the same concept. Traditionally, the problem can be tackled by global and local methods that aim at enhancing the information need representation (i.e., query) but using different sources [25]. The global method expands the query with semantically similar words using external resources such as thesaurus.

On the other hand, the local method depends completely on a set of potentially relevant documents to adjust the query. To identify the relevant documents to a query, relevance feedback is used. There are two types of feedback: (1) explicit feedback where the user identifies the relevant documents to his information need and (2) pseudo feedback where systems automatically treat the top documents as relevant. In this work, we focus on the local methods and discuss them in detail in the next subsections.

The availability of explicit relevance feedback is extremely significant to IR

systems, in general. However, there are many challenges stem from utilizing the user feedback such as the responsiveness of users, and aggregating multiple judgments of a tweet (if any), etc. For the latter challenge, we simply consider the first received judgment for each tweet and discard the reminder judgments if any. The system exploits the explicit feedback for profile expansion.

$$\vec{Q}' = \alpha \vec{Q} + \frac{\beta}{|R|} \sum_{\vec{D} \in R} \vec{D} \quad (16)$$

Where $\vec{Q}'$ is the expanded profile, $\vec{Q}$ is the current profile. Note that we do not reset the profile to its original title before expansion like what we do in the expansion using pseudo relevance feedback. r is the number of truly relevant tweets that were fetched recently and used for expansion, and β is a parameter used to control the influence of relevant tweets' terms on the new profile vector.

To cope with topic development over the stream, we consider the freshness of tweets used for expansion. To achieve this, we propose a temporal Rocchio expansion model that updates profiles as follows:

$$\vec{Q}' = \alpha \vec{Q}_0 + \frac{\beta}{|R|} \sum_{\vec{D} \in R} \vec{D} \times c \quad (16)$$

Where, $\vec{Q}'$ is the expanded query, $\vec{Q}_0$ is the original query, r is the set of recent relevant tweets. To cope with topic change, c is an exponential decay factor with a value between 0-1. The decay factor applies a temporal penalty on tweets and is computed using the following equation:

$$c = e^{-\lambda \cdot \Delta t(D)} \quad (17)$$

λ is a control parameter between 0-1, t is the time difference between the current

system time and the time of the relevant tweet d in days.

3.4.Real-Time Summarization over Arabic Tweets

We extended the RTS system to run over Arabic tweets stream. In general, the filters of the RTS system is language-independent, except the preprocessing module which requires language-dependent analysis. To test the system over Arabic stream, we extended Arabic event detection test collection called *EveTAR* [2]. The new release of *EveTAR* [12] extends the earlier release from several important points with regard to data nature and supported tasks. Among these is the introduction of novelty annotations required for real-time summarization task.

EveTAR test collection was initially designed only for event detection and ad-hoc search IR tasks [2], but not for tweets timeline generation, nor real-time tweets summarizing tasks. These two tasks naturally require novelty in systems' output not only relevance. In other words, in these tasks, systems are penalized when they return semantically-redundant tweets. Hence, we collected novelty annotations for *EveTAR* test collection by recruiting 12 in-house assessors (current or alumni students from Qatar University). We followed the definition of the semantically-similar tweets that was initially developed for TREC 2014 microblog track [17] [43] and adopted by real-time filtering and summarization tracks [18] [20] [19].

Figure 7 illustrates the web-based interface that was used to cluster relevant tweets semantically. The tweets are viewed to the annotator one at a time in chronological order (i.e., tweet creation time). The annotator can add the tweet to an existing cluster or create a new cluster if the tweet is novel. Judging tweets as redundant or novel is subjective (differs from a human to another), hence a general rule for judgment process considers a

tweet that does not add any additional information to the previously seen tweets as redundant, or novel otherwise.

We provided two on-campus training sessions (each was organized for 3 hours). After motivating, defining the annotation task, and presenting a live demo of the annotation interface, we asked annotators to take a quiz before being eligible to work on the task. The quiz consists of two training topics with an average of 150 relevant tweets. After judging the quality of assessors' annotations, we only selected nine annotators to work on the task. We kept the task instructions accessible after the training session[6] for the annotators reference while clustering.

For the 50 topics of EveTAR, the annotators clustered only 22k unique relevant tweets (retweets were excluded). On average, each topic has around 440 tweets. Each topic was annotated by one annotator. The time spent on each topic differs from one to another depending on the annotator speed, her availability and pool size. The whole annotation task finished in roughly 100 hours over 3 weeks and produced 66 clusters per topic on average. We finally propagated the labels to duplicate tweets that were excluded at the beginning.

[6] https://reemsuwaileh.github.io/evetarnovelty/training.html

Figure 7. Screenshot of the novelty annotation interface. Adopted from [43].

CHAPTER 4: EXPERIMENTAL EVALUATION

In this chapter, we discuss the experiments that we conducted to answer the research questions. We start by presenting the evaluation setup (Section 4.1.). We then describe our experiments on the extensions that we built over the relevance filter and discuss the results: (1) applying the traditional retrieval models (Section4.2.), (2) leveraging semantic similarity using word embeddings (Section 4.3), and (3) exploiting relevance feedback (Section 4.4).

4.1.Evaluation Setup

In this section we describe our evaluation test-bed including the test collections that we used, tuning and testing datasets, the evaluations measures, and the pre-trained word2vec models.

4.1.1. Text Collections

We used 4 test-collections: three TREC English test collections and one Arabic test collection (*EveTAR* [12]). The TREC collections are namely TREC-2015 [18], TREC-2016 [20], and TREC-2017 [19]. They are multi-language task-specific (RTS) test collections but used for tracking English profiles over English tweets. As the traditional TREC evaluation, a pool of potentially-relevant tweets, that is constructed using participating systems, is labeled by in-house assessors.

As for the Arabic collection, we used *EveTAR* test collection [12]. *EveTAR* is an Arabic multi-task tweets test collection. It was crawled using Twitter Tracking API[7] over a period of around one month. Using a set of significant events, a list of potentially-relevant

[7] https://developer.twitter.com/en/docs/tweets/filter-realtime/overview

tweets were retrieved using different ranking algorithms and then relevance judgments were acquired using crowdsourcing. Following TREC-style novelty evaluation, we applied the same semantic clustering over relevant tweets (refer to Section 3.4.).

We show the statistics of all the English and Arabic test collections in Table 1 and *Table 2*. Note that the TREC collections were crawled over 10 days (TREC-2015 and TREC-2016) or 8 days (TREC-2017) using Twitter Streaming API[8] at each participant end. The reported numbers are based on our local crawl.

Table 1. *Statistics of tuning and testing test collections.*

Subsets	Crawl period	Size	#profiles	#qrels	#rels
TREC-2015	20-29 Jul 2015	40M	51	94,066	8,233
TREC-2016	2-11 Aug 2016	37M	56	67,525	3,339
TREC-2017	29 Jul - 6 Aug 2017	29M	97	94,307	6,149

4.1.2. Tune and Test Setups

To tune the parameters of the relevance filter, we used three different tune-test setups:

1. Tune using TREC-2015 and test over TREC-2016.
2. Tune using TREC-2016 and test over TREC-2017.

[8] https://developer.twitter.com/en/docs/tweets/sample-realtime/overview/GET_statuse_sample

3. Tune using TREC-2015 and test over TREC-2017.
4. We used the best settings from TREC-2015 setups and test over *EveTAR* test collection.

Table 2. *Statistics of subsets of EveTAR test collection.*

Subsets	Crawl period	Size	#profiles	#qrels	#rels
EveTAR-F		356M	50	61,946	24,086
EveTAR-S	30 Dec 2014	15M	50	61,946	24,086
EveTAR-S.m	-	8M	50	47,369	21,233
EveTAR-S.d	2 Feb 2015	7M	50	14,577	2,853
EveTAR-Q		60K	50	61,946	24,086

4.1.3. Evaluation Measures

In this work, we focus on the quality, not latency, measures of Real-time Tweets summarization track in TREC-2017 [19], namely: the expected gain (EG) and the normalized cumulative gain (nCG) measures. Each of these evaluation measures is evaluated for each topic for each evaluation day and then averaged over all evaluation days. The final score of a run is the average of scores over all topics.

To evaluate the novelty of the pushed tweets, the evaluation setup maintains semantic clusters of relevant tweets. Once a system returns a tweet from one semantic cluster, all other tweets that belong to that cluster are considered redundant, and hence

nonrelevant.

We evaluate the systems' performance using the following evaluation measures:

- **Expected Gain (*EG*):**

$$EG = \frac{1}{N}\sum_{t\in P} G(t) \tag{18}$$

P is the set of tweets that are pushed by the system and *N* is the number of those tweets (i.e., *N* must be $\leq$ 10 tweets).

- **Normalized Cumulative Gain (*nCG*):**

$$nCG = \frac{1}{Z}\sum_{t\in P} G(t) \tag{19}$$

Z is the maximum possible gain for that topic in that specific day based on all judged pushed tweets.

For all measures, the gain *G(t)* of a tweet is assigned one of three values based on its relevance to a corresponding topic (judgments are taken by assessors): (1) 0 if non-relevant, 0.5 if relevant, and 1 if it is highly relevant. The RTS system (push notifications scenario) is allowed to push a maximum of *n*=10 tweets daily per profile during the evaluation period. When a system exceeds the daily-quota, only the first 10 pushed tweets will be considered and the all remaining are discarded. All measures also penalize redundancy in pushed tweets by maintaining semantic clusters; once a tweet from a cluster is pushed, all upcoming pushed tweets from the same cluster are considered non-relevant.

Each of the evaluation measures has two variants of the silent day's treatment. The *variant-1* measures reward systems by a score of 1 if they kept quiet on silent days, and zero otherwise. The *variant-p* measures penalize systems by 0.1 multiplied by the number

of pushed tweets in silent days (fraction of the ten-tweet daily limit).

In all our experiments, we tune for *EG-p* and *nCG-p* measures. Although we believe that *EG-1* and *nCG-1* measures are better in modeling the user expectation from RTS systems (i.e., users expect to receive only relevant tweets if any), they are not convenient for optimization due to discontinuity [19]. However, we report them for comparison purpose since *EG-p* and *nCG-p* were not introduced until TREC-2017. Moreover, although *EG-0* and *nCG-0* measures were reported in TREC-2016 official results, we think they are not practical because systems must be rewarded if they kept silent when there are no relevant tweets. Otherwise, users will not be happy.

4.1.4. Word Embeddings Models

There are several models for distributed word representations such as word2vec model [26] [27], GloVe model [30] and dependency-based word embeddings [15] that is an improved version of word2vec. We opted to use the word2vec model since it has been greatly used recently. Word2vec has two different architectures: Continuous Bag-Of-Words (CBOW) and skip-gram models. Given a word, the former predicts the context words, while the latter does the opposite; it predicts a word given its context.

To utilize word embeddings, we have used a word2vec twitter model that was trained on approximately 400M tweets [10]. The tweets dataset used for training was crawled over around one year using the Twitter Streaming API. Before training, the dataset was preprocessed by token replacement of URLs, mentions and numbers, but not hashtags.

The model was trained using Skip-gram architecture and negative sampling algorithm. Other hyper-parameters were set to their default values expect the context window that was set to 5. The model exhibited the best performance in Part-of-Speech

tagging and Named Entity Recognition tasks among other models those were trained with different hyper-parameters settings of word2vec tool.

4.2.Traditional Retrieval Models (RQ1)

In this section we answer the research question: RQ1: How effective are different retrieval models (e.g., BM25, KL-Divergence) when we use them in relevance filtering for real-time summarization?

Experimental models

We experimented with different retrieval models (i.e., different combination of weighting and scoring functions) that we discussed. We next describe the experimental models that we used in our experiments:

- VSM models: These experimental models fall under the VSM. They use different weighting functions to represent the tweets and profiles. The relevance of a tweet to a profile is computed using *Cosine* similarity function.
    - VS-IDF: This model weights tweets and profiles using IDF term weighting function.
    - VS-TFIDF: This model weights tweets and profiles using TFIDF term weighting function.
    - VS-BM25: This model weights tweets and profiles using BM25 term weighting function. We set the *k* and *b* parameter to their default values, *k* = 2 and *b* = 0.75.
- BM25: This model uses the probabilistic model. It computes BM25 scoring function to score tweets against profiles. We keep the *k* and *b* parameter as default

values.

- KL-DIV: This model uses language modeling. It estimates relevance using KL-Divergence scoring function with Dirichlet smoothing. We set μ parameter of Dirichlet smoothing to its default value, μ=2000.

In addition to the above experimental models, we report the performance of *Silent* model. This model keeps silent during the evaluation period and pushes no tweets to all topics. Its score is only an accumulation of the gain of silent days that happen when an interest profile does not have any relevant tweets.

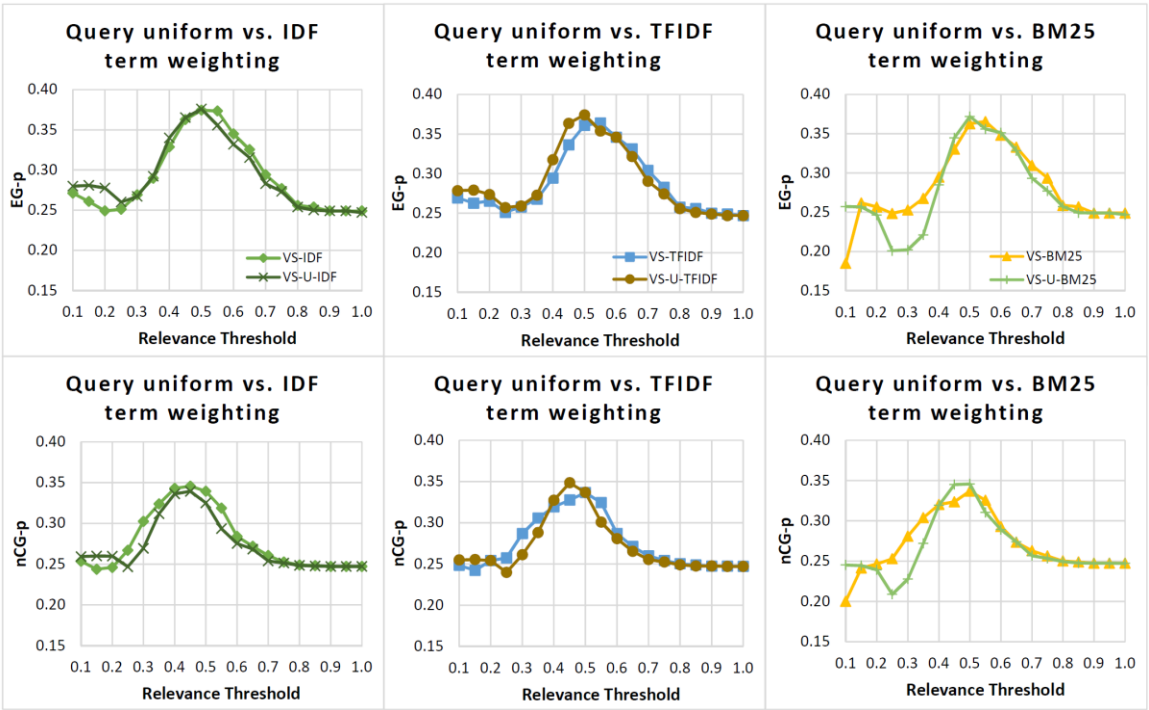

Figure 8. Performance of uniform query weighting versus other weighting functions over TREC-2015 dataset.

Query Uniform Weighting

We also experimented with uniform query weighting for the VSM. In Figure 8, we show the *EG-p* and *nCG-p* performance of uniform query weighting versus other weighting functions over TREC-2015 dataset[9]. We found a slight difference over other weighting functions. Thus, we only consider the non-uniform weighting functions.

Best Tuned Relevance Threshold

For all the above experimental models, we tune the relevance threshold using a grid search method over TREC-2015 and TREC-2016 datasets. We vary the threshold between 0.1-0.9 with a step of size 0.05 for the experimental models that use VSM in the relevance component. We change the relevance threshold between 1-6 for BM25 probabilistic IR model. KL-Divergence model that uses LM is tuned for 0.1-6 range of relevance threshold with a step size 0.05. Table 3 reports the best relevance threshold for each experimental model for different evaluation measures. The best thresholds are almost consistent over VSM experimental models. The difference between thresholds over datasets is not large, however, it falls around the mid-range of the tuning ranges that we used.

Sensitivity of Evaluation Measures to Relevance Threshold

Figure 9 shows the sensitivity of *EG* and *nCG* families of evaluation measures to relevance threshold over TREC-2015 dataset. We noticed similar observations across all datasets. We can clearly see that the range of the best values of relevance threshold differs from measure to another. For example, for *EG-p* measure, the best range of threshold for

[9] We found the same observation over other evaluation measures and datasets.

VSM models is between 0.45 and 0.65. On the other hand, the best range for BM25 and KL-Divergence models are between 0.8-1.0 and 2.5-3.5, respectively. As for *EG-1* measure, these ranges are a bit higher by around 0.1 for VSM and 1 for both BM25 and KL-Divergence.

Table 3. *The best relevance threshold across different evaluation measures*

Models	TREC-2015				TREC-2016			
	EG-p	EG-1	nCG-p	nCG-1	EG-p	EG-1	nCG-p	nCG-1
VS-IDF	0.50	0.55	0.45	1.00	1.00	0.60	0.50	0.60
VS-TFIDF	0.55	0.65	0.50	1.00	1.00	0.70	0.50	0.70
VS-BM25	0.55	0.65	0.50	1.00	1.00	0.70	0.50	0.60
BM25	3.00	4.00	2.50	5.00	3.00	3.50	2.50	3.00
KL-DIV	0.85	1.00	0.85	1.00	0.95	2.50	0.50	2.50

Results

We next report the results of testing each experimental model over test collections and list our observations. For all the aforementioned tuning-testing settings discussed in Section 4.1.2., we show the results in Figure 10 and Figure 11.

We used a two-tailed paired t-test, with a significance level α, to indicate statistically-significant improvements of experimental models compared to the *Silent*

model.

Among the observation we found:

- The VSM retrieval models are almost better in all cases when using the *EG*-measures family. KL-DIV model exhibits better performance using *nCG*-measures family compared to its performance when using *EG*-measures family.
- *KL-DIV* model performs poorly compared to other runs when tested over TREC-2016; mostly it is comparable to or worse than the *Silent* run across all evaluation measures. Nevertheless, it exhibits better performance when tested over TREC-2017 dataset, regardless of which datasets were used in tuning.
- For all models, the ranges of *variant-p* of the evaluation measures are higher than *variant-1*, which could be an indicator of the prevalence of silent topics in all cases.
- The performance of all models on TREC-2017 test collections is not affected by which test collections is used for tuning using *variant-p* evaluation measures, however, for *variant-1* measures, the results of different models are more consistent when tuning on TREC-2016. The BM25 models, perform poorly using *variant-1* measures.
- No model is consistently the best across different evaluation measure.
- No model is consistently the best across different test collections.

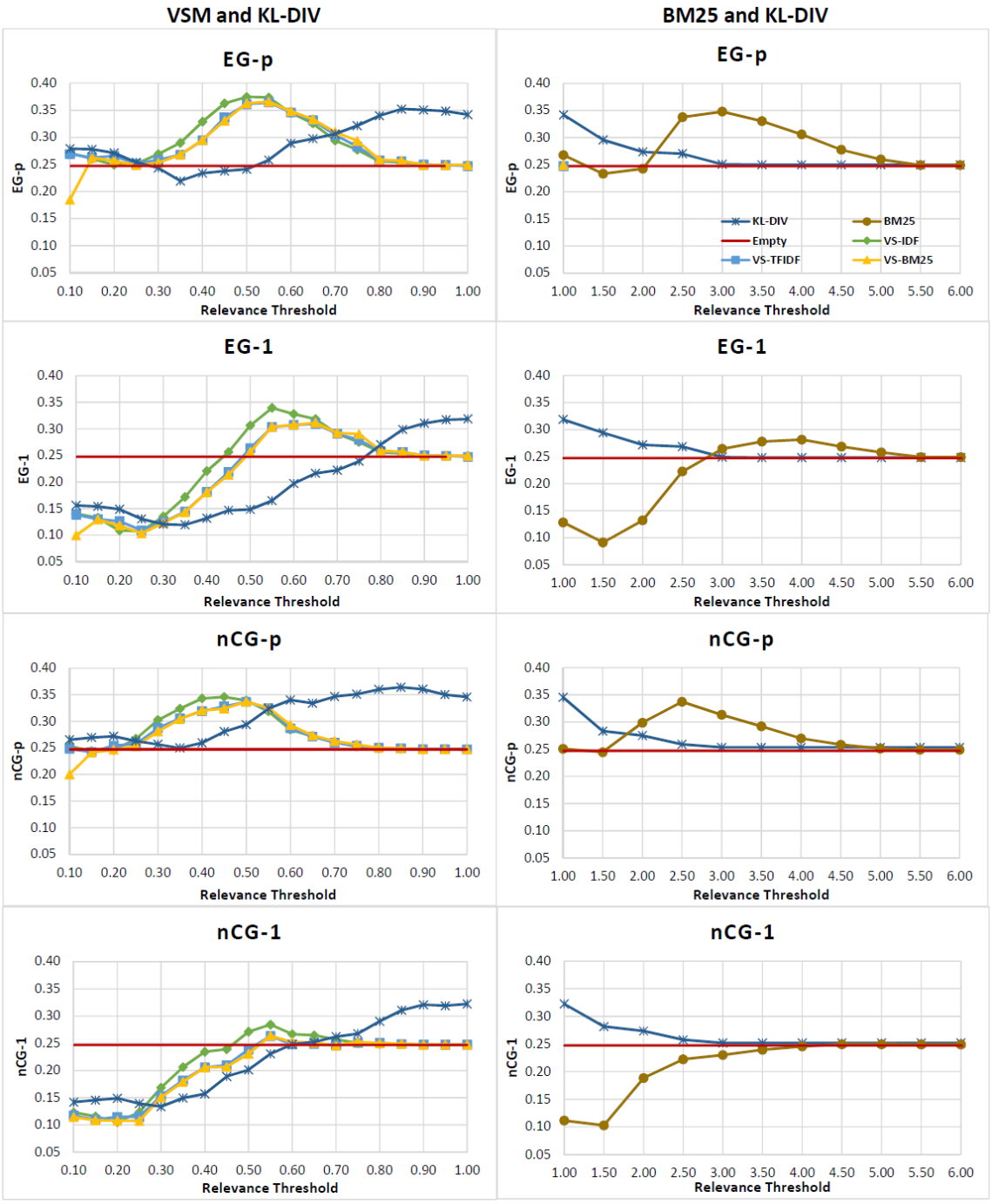

Figure 9. Tuning relevance threshold using different retrieval functions and evaluation measures over TREC-2015 dataset.

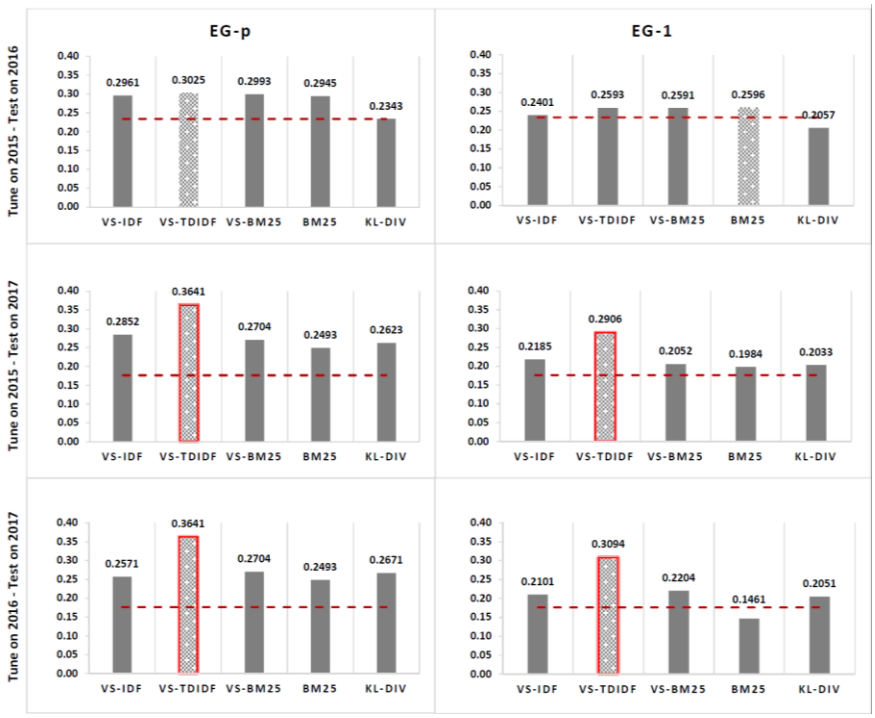

Figure 10. Testing *EG* results of experimental models that use traditional retrieval models. Bars with borders indicate statistical difference over the ***Silent*** model.

Since there is no model that is consistently the best across test collections and evaluation measure, we averaged the performance of each experimental model across the three test collections. We show the results in Figure 12 using the average performance, we can clearly notice that the VS-BM25 experimental model is the best across different evaluation measures. We show here only the *variant-p* measures. other measures exhibit similar performance.

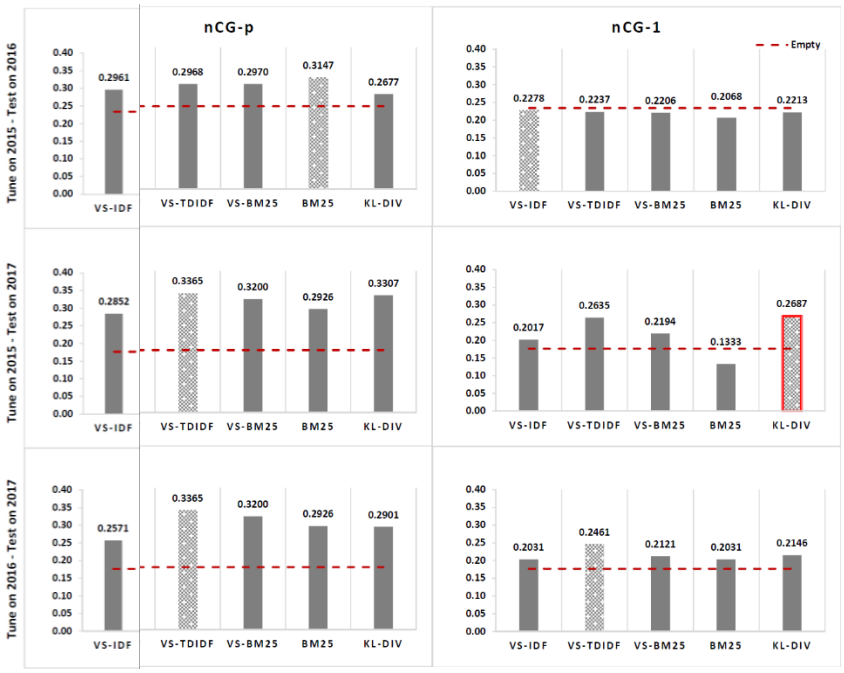

Figure 11. Testing *nCG* results of experimental models that use traditional retrieval models. Bars with borders indicate statistical difference over the ***Silent*** model.

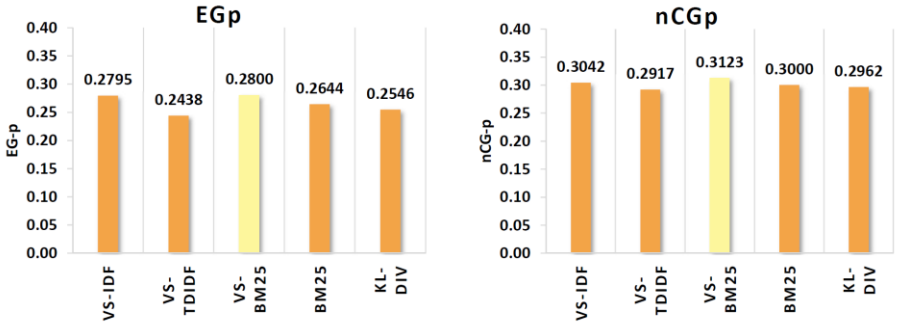

Figure 12. Average performance of traditional experimental models across TREC test collections.

4.3.Leveraging Word Embeddings (RQ2)

In this section, we answer the research question: RQ2: How effective are different ways of incorporating word embedding to represent the text (profiles and tweets) for different retrieval models?

Experimental Models

We experimented with expanded representation for tweets and profiles using word embeddings. In these models, the system expands the profile and tweet vectors in tandem with the missing terms from the union set of their terms. We describe these experimental models in the following:

- E-BM25-*max*: In this model, we estimate the *tf* of a non-matching query term by its maximum similarity to the terms of the tweet to be scored. This similarity is computed between word embeddings vectors. We then substitute this similarity in the BM25 classical model and consider it as the weight of the query term in the query vector. We do the same for the tweet vector.
- E-BM25-*avg*: In this model expand the profile and tweet vectors using the same way as the above run, but instead of max-pairwise similarity, we use the average-pairwise similarity.
- E-TFIDF-*max*: This model is similar to the E-BM25-*max* model, however it used TFIDF term weighting instead of BM25.
- E-TFIDF-*avg*: This model is the same as E-BM25-*max*, but it uses the average-pairwise similarity instead of the max-pairwise similarity.

Tuning and Testing

Similar to the traditional models, we tuned the relevance threshold for word embedding experimental models. We varied the threshold between 0.1-0.9 with a step of size 0.1. Additionally, we used ϵ parameter that controls the influence of word embedding similarity on the term weighting. We set the value of $\epsilon = 0.5$.

In Figure 13, we show the test results of the best configurations over different test collections in comparison to three traditional experimental models, namely: VS-IDF, VS-BM25, and VS-TFIDF. We choose to compare to these models as they leverage the VSM similar to the word embedding experimental models.

Discussion

In all cases, the traditional experimental models perform better than the word embedding experimental models over all test collections and evaluation measures, except in one case (Test on TREC-2016, measure *nCG-1*) where E-BM25-max model has a negligible improvement. Upon investigation, we found the following:

- The out-of-vocabulary problem[10] has negatively affected the word embedding experimental models. For example, in TREC-2015 dataset, 42.6% of the unique terms of the dataset are out of vocabulary. Thus, using the pre-trained model is not fair enough to test the word embeddings models. Although the out-of-vocabulary problem was studied in the literature, it's out of the scope of this thesis.

[10] This problem happens when the the dataset that is used for training the word2vec model does not have a specific word. Thus, the word2vec model will not learn a representation for that word.

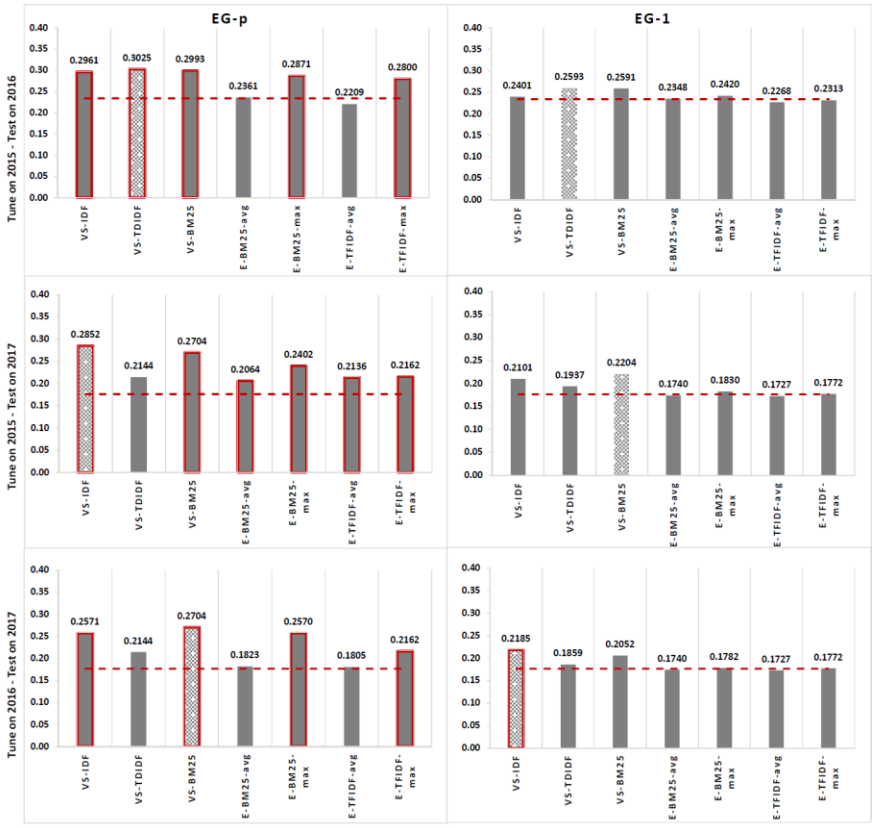

Figure 13. Testing *EG* results of experimental models that leverage word embedding. Bars with borders indicate statistical difference over the *Silent* model.

- The ϵ parameter needs tuning to study the models' behavior deeply. In Figure 15, we show the sensitivity of ϵ parameter on *EG-p* measure over TREC-2015 dataset. We noticed similar behavior using other evaluation measures. The performance of the word embeddings models is improved as ϵ value increases. The models reach the best performance when ϵ = 1 which means the word embeddings is already disabled and there is a match already between the profile and tweet representations.

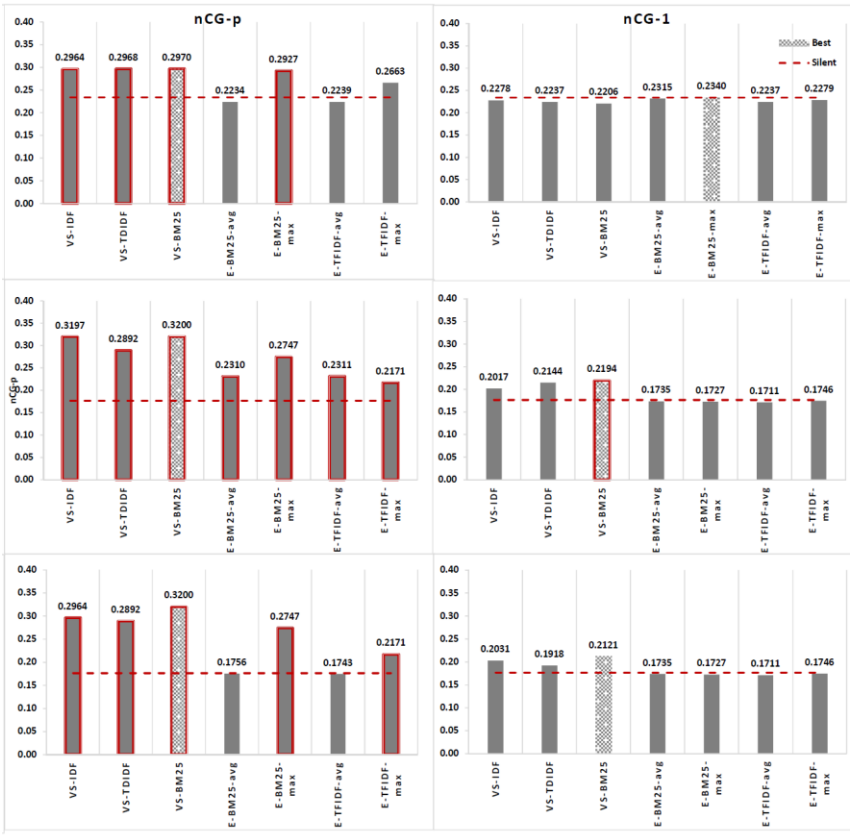

Figure 14. Testing *nCG* results of experimental models that leverage word embedding. Bars with borders indicate statistical difference over the *Silent* model.

To summarize the results of experimental models that leverage word embedding, we show the average performance across all TREC Test collection in Figure 16. The yellow bar indicates the best model and the green bar indicate the best model among models that leverage word embedding. We can notice that there is no improvement over the core system nor VS-BM25 experimental model.

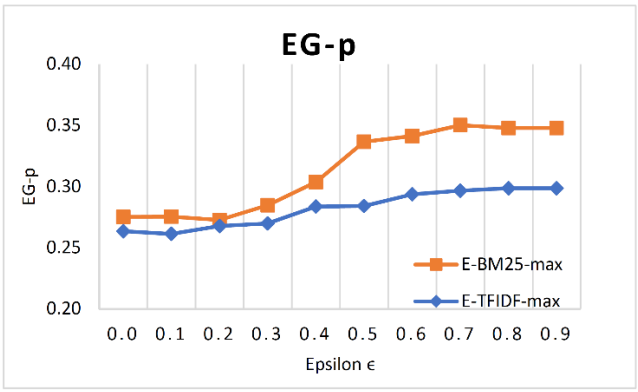

Figure 15. The effect of the ϵ parameter on RTS performance over TREC-2015 dataset.

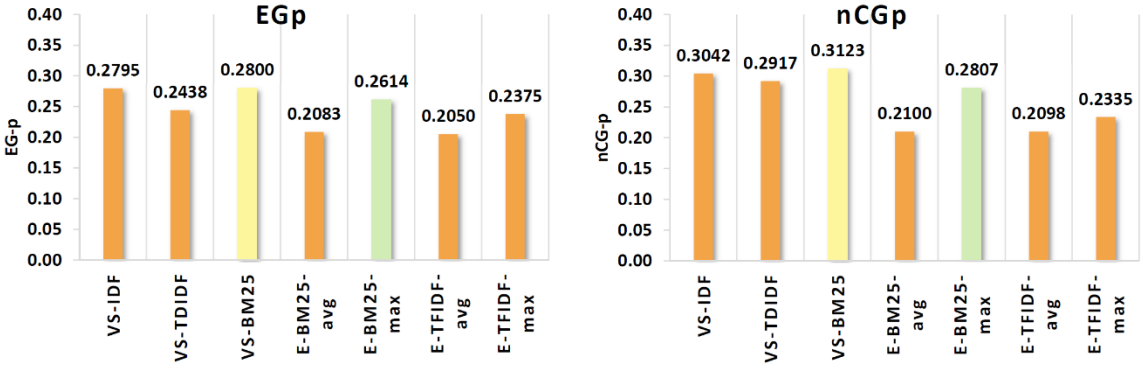

Figure 16. The average performance of embedding experimental models across TREC test collections.

4.4.Exploiting Relevance Feedback (RQ3)

In this section, we answer the research questions: RQ3: How effective are different ways of exploiting explicit relevance feedback?

To evaluate the experimental models that exploit relevance feedback, we used two experimental setups to simulate the feedback arrival to the system:

1. *Immediate feedback*: This is the optimal scenario where the system receives an immediate feedback for each tweet it pushes.
2. *Random feedback*: This is a compromise between optimal scenario (i.e., Immediate feedback) and the real scenario where latency is variable depending on user availability and responsiveness. The system selects *m* tweets from the *n* pushed tweets since the last feedback fetch. The value of *m* is between 1- *n*.

Note that the feedback is only received for tweets already pushed to the system. All the above models perform expansion periodically. Specifically, every 15 minutes, the system fetches feedback from a broker (we simulate the broker in our experiments)[11].

As for the temporal decay parameters (Equation 17), we use the day as a time unit to penalize documents.

Experimental Models

- EXP-*imed*: This model applies profile expansion using a modified version of Rocchio. The feedback arrival follows the "Immediate feedback" method.
- EXP-*rand*: This run is similar to the above run, but it exploits the feedback that is received using the "Random feedback" method.

[11] Broker is a server where tweets are pushed to and systems can fetch explicit relevance feedback from.

- TEXP-*imed*: This model applies profile expansion with temporal decay to penalize old documents (i.e., decrease their contribution to the profile representation to avoid drift). It uses the "Immediate feedback" method for simulating feedback arrival.
- TEXP-*rand*: This run is similar to the previous run but simulates the feedback arrival using "Random feedback" method.

Tuning and Testing

As there is no one experimental model, among the traditional and word embedding models, has exhibited a better performance over the core system, across different evaluation models and test collections, we elected to compare the above experimental models with the relevance filter of the core system (VSM with *idf*-based terms weighting). We opted to tune these experimental models on *EG-p* measure (the official measure used to rank systems in TREC-2017).

Among the parameters that we tuned for the feedback experimental models are β for all experimental models and λ used to compute the decay in temporal models. For feedback models with no temporal penalty, we tuned β for values between 0.1-0.9 with a step size of 0.1. For temporal feedback models, we tuned β for values between 0.2-0.8 with a step size of 0.2 and λ for values between 0.1-0.9 with a step size of 0.2. For all of the above experimental models, we set α=1. We tune the feedback experimental models using *EG-p* only. In It can also be clearly noticed that the best value of β is 0.1 in the feedback models without temporal decay. Although we control the influence of positive feedback using low weights, it still affects the performance negatively.

Table *4*, we report the performance of the feedback models with their best configuration.

Discussion

Looking at the results, the feedback models did not exhibit any improvement over the core system, VS-IDF model using *EG-p* measure, but the feedback improves in *nCG-p* in these results. Nevertheless, as we tune using *EG-p* measure only, we cannot draw a clear conclusion for other measures and we report the *nCG-p* just for reference.

It can also be clearly noticed that the best value of β is 0.1 in the feedback models without temporal decay. Although we control the influence of positive feedback using low weights, it still affects the performance negatively.

Table 4. *Results of feedback models with the best configurations over TREC-2016 text collection. The Best score per column is* ***boldfaced****.*

Model			EG-p	nCG-p
VS-IDF	-	-	0.3046	0.2923
EXP-imed	0.1	-	0.2905	0.3019
EXP-rand	0.1	-	0.2913	0.301
TEXP-imed	0.4	0.1	0.2987	0.2932
TEXP-rand	0.4	0.1	0.2988	0.2934

Furthermore, the model that applies temporal penalty on documents, it performs better than other feedback models. This shows the importance of the freshness of feedback used to update profiles representation over time. This raises the question of what unit of time should the system use to penalize documents. We believe it has to be tuned per topic as topic development depends on the topic type and hence differs from topic to another.

We investigated the issue of poor performance and found that the expansion causes topic drift as we add all terms appear in positive feedback. For example, for a topic with a title (i.e., query) "Self-Driving Cars", the tweet in Figure 17 is relevant and hence added to the topic representation. Later, tweets that have the terms "Audi" and "Race" will match, which are not necessarily related to the original topic. To mitigate the drift, the terms extracted from the positive feedback has to be ranked using a weighting function before adding only the top terms.

Figure 17. Example of a relevant tweet to "Self-driving cars" topic.

4.5.Participation in TREC

We participated in TREC Real-time Filtering (RTF) and Real-time Summarization (RTS) tracks in three years in a row: TREC-2015 RTF [39], TREC-2016 RTS [38] and TREC-2017 RTS [37]. In this section, we discuss our participation push notification task (scenario A), report our results, and highlight the lessons we learned.

4.5.1. Real-time Filtering (RTF) in TREC-2015

In 2015, we built the baseline of our RTS experimental system. Our focus was to study the different techniques for adaptive filtering: (1) Pseudo Expansion, and (2) Dynamic Thresholding.

Official Runs

We submitted three different runs [39]:

- QUBaseline: This run uses a static relevance threshold τ_r= 0.6 across profiles over the evaluation days. The interest profiles are represented using all the profile's fields: title, narrative and description. This run uses only the top 8 terms extracted from the description and narrative fields and controls their influence on profile representation using a parameter σ = 0.2. The profile is periodically expanded using a maximum of 4 expansion terms extracted from pseudo relevant tweets (i.e., the potentially-relevant tweets identified by the system) and weights them using a control parameter, γ = 0.2.
- QUDyn: In addition to using the same settings of QUBaseline run, this run enables the dynamic thresholding feature of adaptive filtering. It dynamically updates the relevance threshold for each profile separately. The relevance threshold set initially

to a high value τ_r = 0.8.

- QUDynExp: Akin to the QUDyn run, this run periodically expands the profiles using a set of pseudo tweets. It uses 10 expansion terms extracted from 12 pseudo-relevant tweets. The system selects 10 expansion terms from narrative and description fields to include in the profile representation and set the values of the control parameters and γ to 0.3.

All the above runs, represent profiles and tweets using *idf*-based weighing in the vector space. To filter the relevant and novel tweets, the system used Cosine similarity and variant of Jaccard. The novelty threshold was set to τ_r = 0.6 for all runs.

Evaluation Measures

The runs were evaluated using two primary measures [16], namely: Expected Latency-discounted Gain (*ELG*) and Normalized Cumulative Gain (*nCG*) [18].

These measures evaluate both quality (relevance and novelty) and latency of the system output over the evaluation period. For the quality aspect, they are computed as *EG* and *nCG* measures discussed in Section 4.1.3. The novelty is evaluated using semantic clusters maintained over the evaluation period. For the latency aspect, the gain *G(t)* of a tweet is reduced using the following time penalty:

$$latency = \max(0, \frac{100 - delay}{100})$$

Where the delay is the difference in minutes between tweet creation time and push time of the tweet.

Official Results

The participant systems were evaluated over an evaluation period that spans 10 days (20th-29th July 2015). Each team had to monitor the 1% sample of Twitter live Stream and track 250 interest profiles. After the evaluation period ended, NIST assessors judged only 51 profiles and created the semantic clusters for the relevant tweets of these profiles to be used in batch evaluation setup.

Table 5. *Official results of our runs of the tweet push notification scenario in TREC-2015. The Best score per column is* ***boldfaced****.*

Run	rank	ELG	nCG
QUBaseline	4	**0.2750**	**0.2347**
QUDyn	21	0.1850	0.1762
QUDynExp	22	0.1848	0.1763
UWaterlooATDKz*	1	0.3175	0.3127

* Indicates the best official automatic run using *ELG* measure.

Lessons

- We have learned many lessons from our participation in TREC-2015 and the post-hoc failure analysis that we conducted. We briefly list them below:
- Simple weighting function, e.g., idf -based weighting functions, performs well in

RTS task.

- A simple representation of profiles, using only the title field, can achieve better matching of relevant tweets.
- Filtering out low-quality tweets, e.g., spam, is important for both the quality and latency of the pushed notifications system.
- Pseudo Expansion harms the system performance as it causes topic drift. Although we conducted further analysis and tuned the parameters, the performance is still poor.

4.5.2. Real-time Summarization (RTS) in TREC-2016

Taking into account the lessons that we learned in TREC-2015, we opted to explore different expansion methods in our participation in TREC-2016. We further improved the system to filter out low-quality tweets; scores only tweets that match at least one profile. The system after all modifications is the same as the core system described in Section 3.2.

Official Runs

We submitted the following runs [38]:

- QUBaseline: This run uses only the title field to represent the interest profiles. This run disables all adaptive filtering features in the system and uses solely the core components of the system. It uses static relevance and novelty thresholds τ_r = τ_n= 0.6.
- QUExpP: This run is similar to QUBaseline, except that it performs pseudo profile expansion hourly
- Using the top p = 20 potentially-relevant tweets identified by the relevance filter, it

extracts the top k = 2 terms to update the profile. To control the effect of the expansion terms, we set γ = 0.2.

- QUExpT :This run expands profiles using pseudo results retrieved from Twitter search API. It issues the profile's title as a search query and retrieves the top p = 20 pseudo results. After ranking the terms of returned results, it extracts the top term (k=1) and adds it to the profile.

Evaluation Measures

The evaluation design in this year followed two modes: online and batch evaluation. We report here only the batch evaluation results [20].

The batch measures were further improved in regard to the silent day's treatment. Additionally, a new measure was proposed to evaluate quality called Gain minus Pain (*GMP*) Note that we do not report results of *GMP* measure as we did not use it in our experiments.

EG and *nCG* measures have two variants in regard to the treatment of silent days: (1) the measures that score all systems by zero in a silent day regardless if they pushed any tweet or not (*EG-0* and *nCG-0*). and (2) the measures that reward systems that kept silent in silent days by a perfect score of 1, and zero otherwise (*EG-1* and *nCG-1*).

Unlike the previous year, the latency was reported separately using two measures, the mean (*MLT*) and median (*MedLT*) latency measures. These measures compute the difference between the push time of a tweet and the first tweet in its corresponding cluster (i.e., reference tweet). However, we don't report them here as our focus is the quality measure, evaluating relevance and novelty.

Official Results

In *Table 6*, we show the quality results of our submitted runs for the push notification scenario in comparison to the baseline run that was provided by the track organizers. QUBaseline scored the best in the push notification task among all other automatic runs. It is unexpected to see again the expansion negatively affects the performance. We perhaps need to do extensive and careful tuning for the expansion parameters and study the effect per profile.

Table 6. *Official quality results of our runs of the tweet push notification scenario in TREC-2016. The Best score per column is* ***boldfaced.***

Run	rank	EG-1	EG-0	nCG-1	nCG-0
QUBaseline[※]	1	**0.2643**	**0.0321**	**0.2479**	0.0157
QUExpP	2	0.2519	0.0233	0.2413	0.0127
QUExpT	3	0.2552	0.0230	0.2455	0.0133
Baseline[*]	19	0.2289	0.0253	0.2330	**0.0295**
Median Oracle	-	0.2335	-	0.2303	-
Best Oracle	-	0.3816	-	0.4576	-

[*] Waterloo baseline provided by the track organizers.

[※] Indicates the best official run using *EG*-1 measure.

Lessons

- Simplicity is invaluable; using a simple term weighting and a straight-forward pipeline is an effective approach for RTS task.
- Filtering our noise is crucial to the effectiveness and the efficiency of a push notification system.
- Conservative matching with only the title field of the interest profiles is effective as RTS task is a precision-oriented and the evaluation measures favor few relevant and novel tweets over many relevant tweets.
- The analysis showed that expansion causes topic drift. We need to explore the weighting functions used to extract the expansion terms from the pseudo results.

4.5.3. Real-time Summarization (RTS) in TREC-2017

Akin to TREC-2016 evaluation design, the track has two modes: the batch and online modes. We report solely the results of the batch evaluation. Most importantly, the participant system had access to explicit relevance feedback from online users [19]. We used the text collections of previous years for tuning and we report the test official results below.

Official Runs

We submitted three automatic runs described below:

- QUBaseline: This run uses the default settings of the core system.
- QUExp: This run uses a similar configuration to QUBaseline run, except that it utilizes the live relevance feedback to perform expansion. It hourly updates the representation of profiles using only the positive relevance feedback. It weights the

feedback by a factor β = 0.2 to mitigate topic drift.

- QUExpDyn: This run uses a similar configuration to QUExp and it dynamically updates the relevance threshold per profile.

In all runs, we set both relevance and novelty thresholds the same value τ_r= τ_n= 0.6.

Official Results

- The evaluation measures used in this year are the same as the measures we use in this thesis (refer to Section4.1.3.). We show our quality results in Table 7.

*Table 7*We compare the official runs to median scores provided by the track organizers and best runs in the track according to EG-p measure. Out of 188 tracked topics in the evaluation period, only 96 topics were judged and used for batch evaluation. Note that, unlike previous years, the evaluation period spans 8 days only.

Looking at the official measure used in ranking systems, QUExpDyn is the best among our runs, however, it exhibits poor performance compared to the best run, HLJIT-Run2. Although the system succeeded for the first time to beat our own baseline (QUBaseline), it is still not able to take full advantage of the explicit feedback in both relevance and novelty filters. Additionally, all QU runs outperform all Median scores in all measures.

Lessons

- Using original Rocchio, by adding all relevant documents to the topic representation, is not a good idea as it leads to topic drift, especially when the profile representation gets larger.

- The system still does not utilize the relevance feedback properly for the relevance filter.
- The "Tweet nomination" component should focus on only quality rather than attempting maximizing quality and latency in tandem.
- The more tweets the system pushes, the lower the performance it achieves.

Table 7. *Official TREC 2017 quality results of QU runs for the push notifications scenario (batch evaluation). Best value per column is* ***boldfaced****.*

Run	rank	EG-p	EG-1	nCG-p	nCG-1
QUBaseline	12	0.2422	0.2146	0.226	0.1984
QUExp	13	0.2356	**0.2185**	0.2159	0.1987
QUExpDyn	11	0.2547	0.2068	0.2475	**0.1996**
HLJIT-Run2z*	1	**0.3630**	0.2088	**0.2808**	0.1266
Median	-	0.2194	0.1951	0.2095	0.1826

* Indicates the best official run using *EG-1* measure.

4.6.Real-time Summarization over Arabic Tweets

In this section, we aim at answering the research question (RQ3): How effective is RTS over Arabic stream?

In addition to the *Silent* model, we evaluated two different experimental models

over *EveTAR* test collection, namely Vector Space Model (*QUBaseline* and *QUExpP*) These models were ranked the top two automatic runs in the "mobile push notification" task in RTS track (Section 4.5.2) [38]. As our system is language-independent, except the preprocessing component, we experiment with tuning and testing across different languages.

In Table 8 and Table 9, we report the performance of these models over all subsets of *EveTAR*. It can be noticed that QUBaseline model outperform others, this is a similar observation to what we found for TREC-2016 English collection (Section 4.5.2). Moreover, the results of all runs across all versions of *EveTAR* are comparable, except for *EveTAR-S.d* subset. This happened because the titles of topics are written in Modern Standard Arabic (MSA), while tweets of the *EveTAR-S.d* version are all in dialectal Arabic. As this caused the *mismatch* problem, the system did not manage to identify many relevant tweets when using this version.

Furthermore, RTS is a precision-oriented task, hence, systems that push less but relevant tweets, are probably of better performance. Similarly, the Silent model has also scored a high performance on *EveTAR-S.d* subset due to the lack of relevant tweets.

Table 8. *Testing results over EveTAR using EG-1. The best result per version per evaluation measure is **boldfaced**.*

Version	QUBaseline	QUExpP	Silent
EveTAR-F	**0.2688**[*]	0.2384[*]	0.1600
EveTAR-S	**0.2799**[*]	0.2455[*]	0.1600
EveTAR-S.m	**0.2975**[*]	0.2620[*]	0.1800
EveTAR-S.d	**0.4813**[*]	0.4741[*]	0.4320
EveTAR-Q	**0.2807**[*]	0.2479[*]	0.1600

[*] Indicates significant improvement over the ***Silent*** model.

Table 9. *Testing results over EveTAR using nCG-1. The best result per version per evaluation measure is **boldfaced**.*

Version	QUBaseline	QUExpP	Silent
EveTAR-F	**0.2469**[*]	0.2333[*]	0.1600
EveTAR-S	**0.2557**[*]	0.2365[*]	0.1600
EveTAR-S.m	**0.2766**[*]	0.2620[*]	0.1800
EveTAR-S.d	**0.4737**[*]	**0.4737**[*]	0.4320
EveTAR-Q	**0.2569**[*]	0.2362[*]	0.1600

[*] Indicates significant improvement over the ***Silent*** model.

CHAPTER 5: CONCLUSION

This thesis focuses on improving the relevance filter of the push notification system. We first extend the system using different traditional retrieval models such as Vector Space Models (VSM) (e.g., different terms weighting such as TFIDF), Probabilistic Model (e.g., BM25), and Language Model (e.g., Kullback–Leibler divergence). We further proposed two extensions that tackle the brevity and topic drift challenges on RTS task: (1) we leveraged word embeddings to utilize semantic similarity with lexical matching, and (2) we exploited the explicit relevance feedback to update topic representation and allow the system to cope with topic development over the stream.

We have conducted extensive experiments on different experimental models for the three groups of extensions and presented the results. Generally, all experimental models perform comparably to the core system across test collections and evaluation measures with no statistically significant improvement over the baseline (the core system). The experimental models that employ word embeddings to estimate semantic similarity did not show better performance over the core system. However, these models were not exploited to their full extent due to the so-called out-of-vocabulary problem. We list few enhancements and recommendations in the future work section below to tackle this problem.

As for the feedback experimental models, the blind usage of the explicit positive feedback did not show improvement over the baseline (core system performance). It increased the problem of topic drift rather than solving it. A better approach could be using conservative and selective methods to decide: (1) whether to apply expansion or no, and

(2) what and which expansion terms the system should use to update the profile. We list our recommendations in the future work section.

Moreover, we extended an Arabic Tweets test collection, EveTAR, that was built initially to evaluate event detection systems by collecting novelty judgments. We hired in-house annotators to semantically cluster relevant tweets of each interest profile. These judgments allow us to evaluate novelty in systems' output beside relevance. We conducted preliminary experiments over *EveTAR* for the task of RTS and reported first results of such system on Arabic stream.

5.1.Future Work

There are still many directions of research work on RTS task that remain to future. We list our future work briefly in the following:

- First of all, we plan to perform extensive failure analysis on all components of the system to better understand the results we obtained. We are mainly interested in studying the effect of the nomination component on the system performance. We believe we cannot isolate the system components and study each separately as the evaluation measures evaluate the decisions made by all of them altogether.
- Since there is no one model that is the best consistently, we need to perform failure analysis aiming at studying the performance of the proposed model based on different criteria such as the profile length, the intensity and frequency of the incoming tweets, and other parameters that might influence the models' performance.

- As for the traditional models and utilizing explicit feedback, we plan to experiment with the so-called Relevance Model [14] when utilizing the live explicit feedback. We further plan to study the system performance using a better simulation of feedback arrival such as Poisson distribution.
- As for utilizing word embeddings, we plan to build our own word embedding models with more data than what was used to build the pre-trained model. We also plan to train our own neural network models with the objective of learning relevance instead of word proximity [45] and study the effect of these models on our system performance.
- As for improving the expansion methods, we plan to make our approach more conservative and selective. In other words, the expansion should be performed after predicting the queries performance (old and new queries), to decide whether to update the topic representation or not. This would implicitly consider the topic difficulty before performing expansion.
- As for Arabic, we plan to do extensive experiments including tuning and testing for all the experimental models that we evaluated on English test collections. We also plan to study Arabic-specific linguistic processing that might help to improve the system performance.
- As we only use the text of the tweet to estimate its relevance, we plan to go beyond using only the textual features of tweets and study the effect on the system when using the social signals (e.g., retweets, likes, etc.).

- Most interestingly, we plan to apply the learning technique to tackle the real-time summarization problem such as learning to rank, deep learning, etc.
- All reported experiments do not include any adaptive filtering techniques. We plan to explore techniques for per-topic dynamic thresholding to enhance the relevance filter with and without expansion.
- Last but not least, we plan to look into the efficiency aspects of the RTS system, in contrast to the effectiveness, including scalability, computation cost, etc. We aim at comparing the relative gain of each experimental model to their computational overhead. Such analysis would enable us to pick the best experimental model that balances between effectiveness and efficiency.

5.2.Related Publications

[1] Maram Hasanain, Reem Suwaileh, Tamer Elsayed, Mucahid Kutlu, and Hind Almerekhi. EveTAR: Building A Large-scale Multi-task Test Collection over Arabic Tweets. Information Retrieval Journal, pages 1–30, 2017.

[2] Reem Suwaileh and Tamer Elsayed. Exploiting Live Feedback for Tweet Real-time Push Notifications. In Proceedings of the 26th Text REtrieval Conference, TREC'17, 2017.

[3] Reem Suwaileh, Maram Hasanain, and Tamer Elsayed. Light-weight, Conservative, yet Effective: Scalable Real-time Tweet Summarization. In Proceedings of the 25th Text REtrieval Conference, TREC'16, 2016.

[4] Reem Suwaileh, Maram Hasanain, Marwan Torki, and Tamer Elsayed. QU at TREC-2015: Building Real-time Systems for Tweet Filtering and Question Answering. In Proceedings of the 24th Text REtrieval Conference, TREC'15, 2015.

REFERENCES

[1] M-Dyaa Albakour, Craig Macdonald, Iadh Ounis, et al. On Sparsity and Drift for Effective Real-Time Filtering in Microblogs. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 419–428. ACM, 2013.

[2] Hind Almerekhi, Maram Hasanain, and Tamer Elsayed. EveTAR: A New Test Collection for Event Detection in Arabic Tweets. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 689–692. ACM, 2016.

[3] Nasser Alsaedi, Pete Burnap, and Omer Rana. Automatic Summarization of Real World Events using Twitter. In *Tenth International AAAI Conference on Web and Social Media*, 2016.

[4] Mossaab Bagdouri and Douglas W Oard. CLIP at TREC 2015: Microblog and LiveQA. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC*, pages 17–20, 2015.

[5] Mossaab Bagdouri and Douglas W Oard. CLIP at TREC 2016: LiveQA and RTS. In *Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC*, 2016.

[6] Hila Becker, Mor Naaman, and Luis Gravano. Selecting Quality Twitter Content for Events. In *Proceedings of the 4th International AAAI Conference on Web and Social Media, ICWSM*, 11, 2011.

[7] Cody Buntain and Jimmy Lin. Burst Detection in Social Media Streams for Tracking Interest Profiles in Real-time. In *Proceedings of the 39th International*

ACM SIGIR conference on Research and Development in Information Retrieval, pages 777–780. ACM, 2016.

[8] Feifan Fan, Yansong Feng, Lili Yao, and Dongyan Zhao. Adaptive Evolutionary Filtering in Real-time Twitter Stream. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1079–1088. ACM, 2016.

[9] Yue Fei, Yihong Hong, and Jianwu Yang. Handling Topic Drift for Topic Tracking in Microblogs. In *Proceedings of the 37th European Conference on Information Retrieval*, pages 477–488. Springer, 2015.

[10] Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. Multimedia Lab @ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153, 2015.

[11] Zhongyuan Han1, Song Li, Leilei Kong, Liuyang Tian, and Haoliang Qi. HLJIT at TREC 2017 Real-Time Summarization. In *Proceedings of the 26th Text REtrieval Conference*, TREC, 2017.

[12] Maram Hasanain, Reem Suwaileh, Tamer Elsayed, Mucahid Kutlu, and Hind Almerekhi. EveTAR: Building a Large-Scale Multi-Task Test Collection over Arabic Tweets. *Information Retrieval Journal*, pages 1–30, 2017.

[13] David Inouye and Jugal K Kalita. Comparing Twitter Summarization Algorithms for Multiple Post Summaries. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 298–306. IEEE, 2011.

[14] Victor Lavrenko and W Bruce Croft. Relevance Based Language Models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.

[15] Omer Levy and Yoav Goldberg. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Volume 2: Short Papers), vol. 2, pages 302-308. 2014.

[16] Jimmy Lin. TREC 2015 Track Guidelines. https://github.com/lintool/twitter-tools/wiki/TREC-2015-Track-Guidelines.

[17] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. Overview of the TREC-2014 Microblog Track. Technical report, Mayland University College Park, 2014.

[18] Jimmy Lin, Miles Efron, Yulu Wang, Garrick Sherman, and Ellen Voorhees. Overview of the TREC-2015 Microblog Track. In *Proceedings of the 24th Text REtrieval Conference*, TREC ’15, 2015.

[19] Jimmy Lin, Salman Mohammed, Adam Roegiest, Royal Sequiera, Luchen Tan, Nimesh Ghelani, Mustafa Abualsaud, Richard McCreadie, Dmitrijs Milajevs, and Ellen Voorhees. Overview of the TREC-2017 Real-Time Summarization Track. In *Proceedings of the 26th Text REtrieval Conference*, TREC, 2017.

[20] Jimmy Lin, Adam Roegiest, Luchen Tan, Richard McCreadie, Ellen Voorhees, and Fernando Diaz. Overview of the TREC-2016 Real-Time Summarization Track. In *Proceedings of the 25th Text REtrieval Conference*, TREC, 2016.

[21] 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, pages 422–429. ACM, 2011.

[22] Feifan Fan Yue Fei Chao Lv, Lili Yao Jianwu Yang, and Dongyan Zhao. PKUICST at TREC 2015 Microblog Track: Query-Biased Adaptive Filtering in Real-time Microblog Stream.

[23] Stuart Mackie, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Comparing Algorithms for Microblog Summarisation. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 153–159. Springer, 2014.

[24] Walid Magdy and Tamer Elsayed. Unsupervised Adaptive Microblog Filtering for Broad Dynamic Topics. *Information Processing & Management*, 52(4):513–528, 2016.

[25] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

[27] 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.

[28] Bilel Moulahi, Lamjed Ben Jabeur, Abdelhamid Chellal, Thomas Palmer, Lynda Tamine, Mohand Boughanem, Karen Pinel-Sauvagnat, and Gilles Hubert. IRIT at TREC Real-time Summarization 2016. In *Proceedings of the 25th Text REtrieval Conference*, TREC, 2016.

[29] Andrei Olariu. Efficient Online Summarization of Microblogging Streams. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 236–240, 2014.

[30] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods on Natural Language Processing, EMNLP*, pages 1532–1543, 2014.

[31] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at TREC-3. *NIST Special Publication Sp*, 109:109, 1995.

[32] Joseph John Rocchio. Relevance Feedback in Information Retrieval. *The SMART retrieval system: experiments in automatic document processing*, pages 313–323, 1971.

[33] Karankumar Sabhnani and Ben Carterette. University of delaware at TREC 2017 Real-time Summarization Track. In *Proceedings of the 26th Text REtrieval Conference*, TREC, 2017.

[34] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008.

[35] Lidan Shou, Zhenhua Wang, Ke Chen, and Gang Chen. Sumblr: Continuous Summarization of Evolving Tweet Streams. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 533–542. ACM, 2013.

[36] Ian Soboroff, Iadh Ounis, Craig Macdonald, and Jimmy Lin. Overview of the TREC-2012 Microblog Track. In *Proceedings of the 21st Text REtrieval Conference*, TREC, page 20, 2012.

[37] Reem Suwaileh and Tamer Elsayed. Exploiting Live Feedback for Tweet Real-time Push Notifications. In *Proceedings of the 26th Text Retrieval Conference*, TREC, 2017.

[38] Reem Suwaileh, Maram Hasanain, and Tamer Elsayed. Light-weight, Conservative, yet Effective: Scalable Real-time Tweet Summarization. In *Proceedings of the 25th Text REtrieval Conference*, TREC '16, 2016.

[39] Reem Suwaileh, Maram Hasanain, Marwan Torki, and Tamer Elsayed. QU at TREC-2015: Building Real-Time Systems for Tweet Filtering and Question Answering. In *Proceedings of the 24th Text REtrieval Conference*, TREC, 2015.

[40] Luchen Tan, Adam Roegiest, and Charles LA Clarke. University of Waterloo at TREC 2015 Microblog Track. In *Proceedings of the 24th Text REtrieval Conference*, TREC, 2015.

[41] Luchen Tan, Adam Roegiest, Charles LA Clarke, and Jimmy Lin. Simple Dynamic Emission Strategies for Microblog Filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, SIGIR, pages 1009–1012, 2016.

[42] Jizhi Tang, Chao Lv, Lili Yao, and Dongyan Zhao. Pkuicst at TREC 2017 Real-time Summarization Track: Push Notifications and Email Digest. In *Proceedings of the 26th Text REtrieval Conference*, TREC, 2017.

[43] Yulu Wang, Garrick Sherman, Jimmy Lin, and Miles Efron. Assessor differences and user preferences in tweet timeline generation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 615–624. ACM, 2015.

[44] Wei Xu, Ralph Grishman, Adam Meyers, and Alan Ritter. A preliminary study of tweet summarization using information extraction. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 20–29, 2013.

[45] Hamed Zamani and W Bruce Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 505–514. ACM, 2017.

[46] Chengxiang Zhai and John Lafferty. Model-based Feedback in the Language Modeling Approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410. ACM, 2001.

[47] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, pages 268–276. ACM, 2017.

[48] Xiang Zhu, Jiuming Huang, Sheng Zhu, Ming Chen, Chenlu Zhang, Li Zhenzhen, Huang Dongchuan, Zhao Chengliang, Aiping Li, and Yan Jia. NUDTSNA at TREC 2015 microblog track: A Live Retrieval System Framework for Social Network Based on Semantic Expansion and Quality Model.