Real Time Filtering of Tweets Using Wikipedia Concepts and Google Tri-gram Semantic Relatedness

Anh Dang[1], Raheleh Makki[1], Abidalrahman Moh'd[1], Aminul Islam[1], Vlado Keselj[1], and Evangelos E. Milios[1]

[1]{anh,niri,amohd,islam,vlado,eem}@cs.dal.ca
[1]Dalhousie University, 6050 University Avenue, Halifax, NS, B3H 4R2, Canada

***Abstract*—This paper describes our participation in the mobile notification and email digest tasks in the TREC 2015 Mircoblog track. The tasks are about monitoring Twitter stream and retrieving relevant tweets to users' interest profiles. Interest profiles contain the description of a topic that the user is interested in receiving relevant posts in real-time. Our proposed approach extracts Wikipedia concepts for profiles and tweets and applies a corpus-based word semantic relatedness method to assign tweets to their relevant profiles. This approach is also used to determine whether two tweets are semantically similar which in turn prevents the retrieval of redundant tweets.**

I. Introduction

The rapid growth of microblogs' popularity greatly promotes the importance of information retrieval systems that suggest relevant content to users with respect to their interests in real-time. The TREC 2015 Microblog track presents two scenarios for the real-time retrieval task: mobile notification (Scenario A) and email digest (Scenario B). In both scenarios, the goal is to retrieve interesting and novel tweets relevant to users' interests profiles which are the description of the topic the user is interested in. In the mobile notification scenario, selected tweets by the system should be pushed to the user's mobile phone as notification relatively short after these tweets are published, while in the email digest scenario, interesting tweets are accumulated in an email and then delivered to the user at the end of the day.

This year's Microblog track guidelines[1] specifies that each team can submit up to three runs per scenario and each run should be assigned to one of the three different categories of the amount of human involvement. In the automatic run, no human intervention is allowed, while in the manual preparation and manual intervention run, human supervision is acceptable before the start of the evaluation period and all the time respectively. We participated in both mobile notification and email digest scenarios and submitted the runs under our group name "DALTREC".

Our proposed approach for this year's filtering task is based on using Wikipedia and Google Trigram for calculating the semantic relatedness between tweets and profiles and also between tweets themselves. Due to the short and noisy nature of Twitter posts and the vocabulary mismatch between the interest profiles and tweets, it is important to consider semantic similarity of tweets to profiles in order to achieve proper recall. We apply the proposed approach to both scenarios considering both automatic and manual preparation runs. In addition, to compare the results of the similarity computation with a standard Ad Hoc information retrieval method, we also applied the Lucene[2] implementation of unigram language model to Scenario B and submitted the results.

The rest of this paper is organized as follows. The proposed strategies are explained in Section II. Section III describes the evaluation metrics used for ranking participants in this year's track. The results of our participation is discussed in Section IV and conclusions are provided in Section V.

II. METHODOLOGY

This section describes our method for the real-time filtering task. The first 4 sections describes the core of the system for both automatic and manual runs. Section II-F describes the user involvement for our manual run submissions and Section II-G explains a language model strategy that we applied to Scenario B. Our proposed framework and its sections are shown in Figure 1.

A. Data Preprocessing

For data processing, we filter out all stop words, non-English tweets, and retweets. After that each tweet and profile are represented as a bag of concepts based on Wikipedia entity linking and we compute semantic similarity between two Wikipedia bags of concepts.

B. Wikipedia Entity Linking

As the tweet content is very sparse and noisy, we try to capture the most important topics from tweets using Dexter Wikipedia entity linking [1]. For Dexter Wikipedia linking, the input is a tweet t_i and the output is a list of potential Wikipedia entity tags. Each tweet t_i is represented as a vector $\{c_{i1}, c_{i2}, \ldots, c_{in}\}$ where c_{ij} is a concept from Wikipedia extracted for tweet t_i. Furthermore, we filtered all tweets that do not have any Wikipedia entities.

C. Similarity Calculation Between Tweets and Users' Profiles

The goal of the system is to recommend interesting content to users with regard to their interests. We propose the use of semantic similarity for this task. For example, if a user is interested in the topic "airline merger", she may also be interested in "airline integration".

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

[2]https://lucene.apache.org/core/

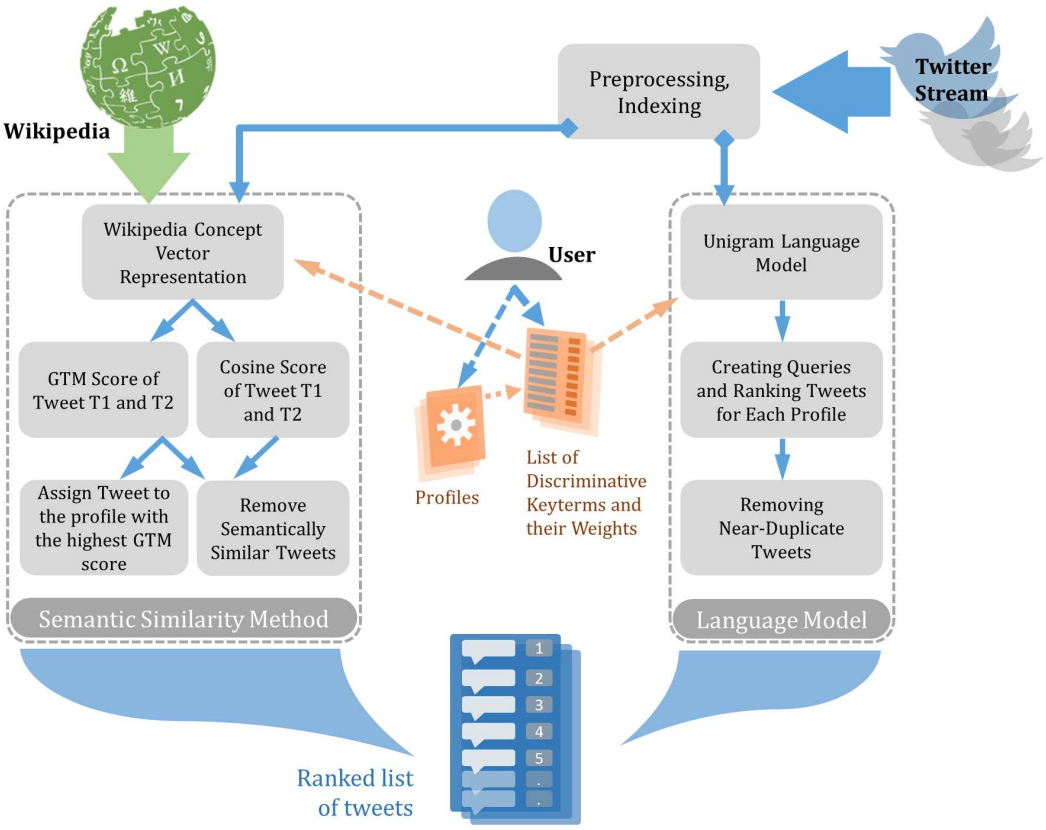

Fig. 1: The framework of the proposed methods for the real-time filtering task.

Semantic similary between two texts is computed using Google Tri-gram Method (GTM) of Aminul et. al [2]. GTM is an unsupervised corpus-based approach for computing semantic relatedness between texts. It uses the uni-grams and tri-grams of the Google Web 1T N-grams corpus [3] to calculate the relatedness between words, and then extends that to longer texts. The Google Web 1T N-grams corpus contains the frequency count of English word n-grams (unigrams to 5-grams) computed over one trillion words from web page texts collected by Google in 2006. To compute the similarity score between a tweet and a profile, we use Wikipedia entity linking method and represent them as bag of concepts and then compute the semantic similarity between two Wikipedia bags of concepts using GTM.

D. Filtering Semantically Similar Tweets

A satisfactory tweet recommendation system should not suggest redundant tweets to users. Therefore, this year's evaluation workflow considers penalties for semantically similar tweets. First, tweets are clustered semantically and within each cluster, only the earliest tweet is considered novel and other tweets in the same cluster are considered redundant.

We propose the use of both lexical similarity and semantic similarity for identifying semantically similar tweets. As described in Section II-B, we represent each tweet with their Wikipedia concepts. We use the GTM method to calculate the semantic similarity between two tweets $t_1 = \{c_{11}, c_{12}, ..., c_{1n}\}$ and $t_2 = \{c_{21}, c_{22}, ..., c_{2n}\}$. To compute lexical similarity between two texts, each tweet is represented as a IF-IDF vector, and their cosine value is considered as their similarity score. We consider two tweets to be semantically similar if both their lexical similarity score and semantic similarity score are more than predefined thresholds, i.e. $GTM(t_1, t_2) >= \alpha$ and $Cosine(t_1, t_2) >= \beta$, where α and β are two thresholds for the semantic similarity and cosine similarity respectively.

E. Automatic Submissions

Having all the modules that calculate the most similar profile to a tweet and identify semantically similar tweets, we apply the following strategies for the automatic run submissions of Scenario A and B.

For Scenario A, we compute similarity score between a tweet content and all 225 interest profiles in the dataset and assign it to the profile that achieves the highest similarity score if it is not semantically duplicated with the previously chosen tweets for that profile. In Scenario B and for each profile, we collect all the tweets that are related to this profile. Since the upper limit of the number of tweets for scenario B is 100

tweets for each profile per day, we cluster these tweets into 100 clusters at the end of the day and select a representative tweet in each cluster to be included in the result (email digest).

F. Manual Preparation

In the manual preparation type, the system can incorporate the user input before the evaluation period. Interest profiles contain discriminative keyterms that are good indicators of their topic. However, all keyterms are not of the same importance. For instance, consider a person who is interested in Sudoku puzzles. Keyterm "Sudoku" is a better indicator of the tweets related to this profile rather than keyterm "puzzle" and therefore should have a higher contribution in the calculations. Consequently, we manually constructed a list of discriminative features for each profile. Each feature in this list has a weight that indicates the importance of that feature for the specific profile. We use these weights to bias our similarity and semantically duplicated tweet calculations.

G. Language Model

As an alternative model and for the sake of comparison, we also submitted the results of applying a unigram language model to Scenario B. We used the Lucene implementation of the language model with the Bayesian smoothing using Dirichlet priors. The value of parameter μ in this smoothing technique is set to 2100 as it is used by Li et.al. for tweets [4]. In addition, it is shown that although the optimal value of μ varies between 500 and 10000, it is usually around 2000 [5]. We expect that involving the user and incorporating her knowledge improves the performance of the task. Therefore, we also consider the list of features-weights which is manually created in our language model. Consequently, we submitted our results for Scenario B and under the manual preparation category.

In this model, we first use Lucene to index the tweets posted during the evaluation period, at the end of each day. Then, for each profile, we create a query from the list of manually extracted discriminative features of that profile, and use query level boosting to set a boost for each feature based on their weights. After that, we use the unigram language model to retrieve a ranked list of relevant tweets based on their relevance score. Having the tweets ranked, we start from the top and include each tweet in the result if they meet both of these two conditions: 1- the tweet should not be a near-duplicate of what has already been included in the result list, 2- its score should be higher than a predefined threshold. If the tweet does not meet either of these conditions, we ignore it and move to the next tweet in the ranked list. This is continued until there are 100 tweets in the results, or there are no more tweets in the ranked list. Having a threshold prevents retrieving non-relevant tweets containing only one of the manually extracted keyterms that are not discriminative enough to be an indicator of the corresponding topic. For instance, if a tweet contains keyterm "puzzle" it does not necessarily mean that the tweet should be recommended to a user who is interested in Sudoku puzzles.

Although filtering semantically similar tweets is part of the task, we only applied the lexical similarity calculation to this model and considered two tweets to be near-duplicate if their cosine similarity is higher than 0.9.

III. Evaluation Metrics

This section briefly discusses the evaluation metrics considered for each scenario. These metrics are explained in detail in the track guidelines[3].

A. Scenario A - Mobile Push Notification

One of the evaluation metrics for Scenario A is Expected Latency-discounted Gain (ELG). This score is computed based on the following relation where T is the set of recommended tweets and $gain(t_i)$ indicates the relevance of the tweet to the corresponding interest profile.

$$ELG = \frac{1}{|T|} \sum_{i=1}^{|T|} \{gain(t_i)\}$$

An explanation of the gain scores is showed in Table I. It is also important to consider that only the first tweet from each semantic cluster receives any score [6].

Gain	Details
0	Not interesting, spam/junk tweets
0.5	Somewhat interesting tweets
1.0	Very interesting tweets

TABLE I: An explanation of gain scores.

To penalize the late recommendation of tweets, a latency penalty is applied. This penalty is computed as MAX(0, (100 - delay)/100), where the delay is the difference (in minutes) between the time the tweet was published and the time the system sends the notification. In addition to ELG, the normalized Cumulative Gain (nCG) is considered as another metric. This metric is computed based on the following relation, where Z is the maximum potential gain considering the 10 tweet per day limit.

$$nCG = \frac{1}{Z} \sum_{i=1}^{|T|} \{gain(t_i)\}$$

The score of each profile for a day is first computed and the final score of the profiles is an average of their daily scores across all days in the evaluation period. The score of each run will be the average of the scores across profiles [6].

B. Scenario B - Email Digest

The evaluation metric considered for profile B is the Normalized Discounted Cumulative Gain (NDCG). This metric is calculated over k top retrieved results and is one of the metrics commonly used in evaluation information retrieval systems [7]. In this year's evaluation, k is set to 10. The following two relations explains the computation of the Discounted Cumulative Gain (DCG) and its normalized version (NDCG), where T is the set of selected tweets and $IDCG@k$ is the maximum possible DCG for the top k tweets.

$$DCG@k = \sum_{i=1}^{|T|} \frac{2^{gain(t_i)-1}}{log_2(i+1)}$$

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

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

For each profile, the value of the NDCG@k is first calculated per day and its final value is the average over 10 days. Therefore, the score for a run is calculated by the average over all profiles.

IV. Results

A. Scenario A

Table II shows the results all participating teams for Scenario A. In our submitted runs for this scenario, DALTRECMA1, DALTRECMA2 and DALTRECAA1, we set the threshold $\alpha = 0.6$ and $\beta = 0.8$. We achieved 0.1753 ELG and 0.2426 nCG for the automatic run which is comparable with other systems. We submitted two manual runs, the first run achieved 0.1620 ELG, 0.1614 nCG and the second run achived 0.1822 ELG, 0.1814 nCG. One of the possible reasons for moderate results may be the threshold for the GTM score which is set to 0.6. We suspect that our system does not capture potentially interesting tweets that their score is below this threshold.

B. Scenario B

The results of Scenario B are presented in Table III. For this scenario, we set the threshold to similar values of Scenario A, i.e. $\alpha = 0.6$ and $\beta = 0.8$. We submitted two automatic runs, DALTRECAB1 and DALTRECMB1, and one manual run, DALTREC_B_PREP. Automatic systems achieved 0.1339 and 0.1323 for NDCG score and the score for our manual preparation was 0.2210. It seems that our alternative approach, language model, is worth investigating further. In addition, we would like to combine the proposed strategies and examine whether it leads to better results.

Run	Group	ELG	nCG	Type
PKUICSTRunA2	PKUICST	0.3175	0.3127	manual
UWaterlooATDK	UWaterlooMDS	0.3150	0.2679	automatic
SNACSA	NUDTSNA	0.3086	0.3349	manual
SNACS_LA	NUDTSNA	0.2863	0.2974	manual
QUBaseline	QU	0.2750	0.2347	automatic
udelRun2A	udel	0.2670	0.2064	automatic
UWaterlooATEK	UWaterlooMDS	0.2654	0.2365	automatic
IRIT-KLTFIDF	IRIT	0.2652	0.2600	manual
prnaTaskA2	prna	0.2603	0.2296	automatic
prnaTaskA1	prna	0.2597	0.2348	automatic
prnaTaskA3	prna	0.2566	0.2289	automatic
udelRun1A	udel	0.2505	0.2070	automatic
hpclab_pi_algA	HPCLAB_PI	0.2477	0.2472	manual
umd_hcil_run01	umd_hcil	0.2471	0.2471	automatic
UWaterlooATNDEK	UWaterlooMDS	0.2470	0.2170	automatic
UWCMBP1	WaterlooClarke	0.2450	0.2035	automatic
ECNURUNA1	ECNU	0.2314	0.2314	automatic
ECNURUNA2	ECNU	0.2314	0.2314	automatic
ECNURUNA3	ECNU	0.2314	0.2314	automatic
udelRun3A	udel	0.2259	0.1910	automatic
IritSigSDA	IRIT	0.2122	0.2043	automatic
umd_hcil_run02	umd_hcil	0.2020	0.2020	automatic
IRIT-RTNotif.33	IRIT	0.1950	0.1834	automatic
QUDyn	QU	0.1850	0.1762	automatic
QUDynExp	QU	0.1848	0.1763	automatic
DALTRECMA2	DalTREC	0.1822	0.1814	manual
CLIP-A-DYN-0.5	CLIP	0.1753	0.2426	automatic
DALTRECMA1	DalTREC	0.1620	0.1614	manual
CLIP-A-5.0-0.5	CLIP	0.1753	0.2426	automatic
CLIP-A-5.0-0.6	CLIP	0.1753	0.2426	automatic
DALTRECAA1	DalTREC	0.1753	0.2426	automatic
PKUICSTRunA1	PKUICST	0.1753	0.2426	automatic
PKUICSTRunA3	PKUICST	0.1753	0.2426	automatic
UWCMBP2	WaterlooClarke	0.1753	0.2426	automatic
MPII_HYBRID_PW	MPII	0.1753	0.2426	automatic
MPII_LUC_SORT	MPII	0.1753	0.2426	automatic
MPII_COMB_SORT	MPII	0.1753	0.2426	automatic

TABLE II: Results of Scenario A from [6].

V. Conclusion

This paper presents the DalTREC team participation in the mobile notification and email digest tasks of the TREC 2015 Mircoblog track. We proposed a novel approach for assigning tweets to profiles and to determine whether two tweets are semantically similar using Wikipedia as an external knowledge source and a corpus-based word semantic relatedness method. Results show that the proposed approach is comparable with other systems in this competition.

Acknowledgment

The research was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani, "Dexter: an open source framework for entity linking," in *Proceedings of the sixth international workshop on Exploiting semantic annotations in information retrieval*. ACM, 2013, pp. 17–20.

[2] A. Islam, E. Milios, and V. Kešelj, "Text similarity using google tri-grams," in *Proceedings of the 25th Canadian Conference on Advances in Artificial Intelligence*, ser. Canadian AI'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 312–317. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30353-1_29

[3] T. Brants and A. Franz, "The google web 1t 5-gram corpus version 1.1," *Technical Report*, 2006.

[4] C. Li, Y. Wang, P. Resnick, and Q. Mei, "Req-rec: High recall retrieval with query pooling and interactive classification," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 163–172.

[5] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 334–342.

[6] M. W. Y. S. G. Lin, Jimmy Efron and E. Voorhees, "Overview of the trec-2015 microblog track," in *Proceedings of the Twenty-Fourth Text REtrieval Conference (TREC 2015)*. Gaithersburg, Maryland: ACM, November, 2015.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89–96.

Run	Group	nDCG@10	Type
SNACS_LB	NUDTSNA	0.3670	manual
SNACS	NUDTSNA	0.3345	manual
CLIP-B-0.6	CLIP	0.2491	automatic
umd_hcil_run03	umd_hcil	0.2471	automatic
CLIP-B-0.5	CLIP	0.2420	automatic
PKUICSTRunB3	PKUICST	0.2343	automatic
PKUICSTRunB2	PKUICST	0.2228	manual
PKUICSTRunB1	PKUICST	0.2226	automatic
DALTREC_B_PREP	DalTREC	0.2210	manual
UWaterlooBT	UWaterlooMDS	0.2200	automatic
UWaterlooBTTND	UWaterlooMDS	0.2196	automatic
CLIP-B-0.4	CLIP	0.2117	automatic
MPII_COM_MAXREP	MPII	0.2093	automatic
hpclabpibm25mod	HPCLAB_PI	0.2046	manual
UNCSILS_WRM	UNCSILS	0.2045	automatic
udelRun2B	udel	0.2026	automatic
umd_hcil_run04	umd_hcil	0.2020	automatic
udelRun1B	udel	0.1966	automatic
UNCSILS_HRM	UNCSILS	0.1902	automatic
UNCSILS_TRM	UNCSILS	0.1890	automatic
IRIT100KLTFIDF	IRIT	0.1784	manual
udelRun3B	udel	0.1778	automatic
IRIT-RTDig.33	IRIT	0.1680	automatic
ECNURUNB1	ECNU	0.1610	automatic
prnaTaskB2	prna	0.1463	automatic
ECNURUNB3	ECNU	0.1416	automatic
DALTRECAB1	DalTREC	0.1339	automatic
BJUTllyQE	BJUT	0.1334	automatic
IritSigSDB	IRIT	0.1329	automatic
ECNURUNB2	ECNU	0.1327	automatic
DALTRECMB1	DalTREC	0.1323	automatic
QUBaselineB	QU	0.1288	automatic
UWCMBE1	WaterlooClarke	0.1232	automatic
QUFullExpB	QU	0.1196	automatic
QUExpB	QU	0.1180	automatic
UWCMBE2	WaterlooClarke	0.1035	automatic
BjutNMF1	BJUT	0.1008	automatic
BjutNMF2	BJUT	0.0685	automatic
prnaTaskB1	prna	0.0641	automatic
prnaTaskB3	prna	0.0533	automatic
MPII_LUC_MART	MPII	0.0310	automatic
MPII_COMB_MART	MPII	0.0275	automatic

TABLE III: Results of Scenario B from [6].