IRIT at TREC Microblog 2011

Firas Damak, Lamjed Ben Jabeur, Guillaume Cabanac, Karen Pinel-Sauvagnat, Lynda Tamine, and Mohand Boughanem

{damak, jabeur, cabanac, sauvagnat, tamine, boughanem}@irit.fr,
IRIT/SIG-RFI,
118 route de Narbonne F- 31062 Toulouse cedex 9,
+33-5-61-55-67-65

Abstract. This paper describes the participation of the IRIT lab, university of Toulouse, France, to the Microblog Track of TREC 2011. Two different approaches were experimented by our team, which are described in the two main parts of the paper.

1 Introduction

Microblogging consists in sharing short messages (microblogs) through a social network platform. Twitter, the most popular microblogging service to date, has experienced exponential growth in recent years. Launched in October 2006, Twitter's number of users has increased from 94,000[1] in April 2007 to 200 millions in 2010[2]. It takes now one week for users to send a billion of tweets[3]. Having this important microblogging activity, users are overwhelmed by the enormous quantity of new tweets and difficulty accessing to their interesting topics. A new retrieval task consisting on tweet search is therefore necessary. This task is triggered by social and timely motivations in addition to the topically motivations characterizing the traditional Web search. Users are searching over microblogs for short, concise and real-time information which is not already indexed by Web search engines.

Among the works having addressed tweet search task, we identify two categories of approaches. The first one considers various topically, microblogging and social indicators as input parameters for machine-learned ranking system, one can fo example cite the approach in [1]. The second category of model-based approaches defines relevance as a multidimensional component and represents the different relevance factors into an integrated model [2–4]. In this paper, we present two model-based approaches to rank tweets given a topic: the first approach use three factors to evaluate relevance. Each factor is computed through a set of feature scores. These factors are named content features, Twitter features and author features. The second approach proposes a Bayesian network model that integrates textual similarity, the influence of microblogger in the network and the time magnitude of the tweet.

The remainder of this paper is organized as follows: the first approach and relative results are described in section 2, while the second approach and relative results are given in section 3.

2 Combining specific features for microblog search

2.1 Method

TREC Microblog Track organizers defined external evidence as evidence outside the Tweets 2011 corpus. Future evidence are information that would not have been available to the system

[1] http://www.usatoday.com/tech/webguide/2007-05-28-social-sites_N.html
[2] http://www.bbc.co.uk/news/business-12889048
[3] http://blog.Twitter.com/2011/03/numbers.html

at the time the query was submited. They decided that runs created using external or future evidence will be ranked separately from runs that do not. Besides future evidence, we believe that a system using external evidence runs in strict real-time sense: concretely, there is no meaning for future evidence. Moreover, external resources are commonly used in IR. For this reason, we tried to avoid future evidences in our approach.

2.1.1 Design of our approach

The first step in our approach is to index and retrieve the top-N relevant tweets for each topic using an usual plain-text search engine. The second step consists in processing these results to evaluate the feature scores (i.e., content, Twitter and author features). The final score is computed by combining the search engine score with the set of the feature scores. The top tweets are then sorted in a reverse chronological order. Before displaying results, we processed the resulting tweets with a language filter so as only those tweets written in English would be considered (see figure 1).

Our approach is designed to handle multiple search options: first of all, we had the choice between either indexing all the collection or making an index for each topic. The tweets indexed in the second case are only those published before the topic's timestamp. As we did not want to use future evidence, we only worked with this second version of index.

We also had the possibility to choose between using the topic as it was given by Track organizers, or adding keywords from our topic expansion module (formulate topic in the figure). Then, when calculating the score, we had the possibility to use only the search engine score, or to combine it with feature scores. Finally, we designed a set of refinement parameters:

– *MIN:* minimum score of the tweets to be displayed in the results.
– *MT:* the number of terms added to the topic.
– *MN:* supplies parameter (i.e., the number of results returned by the search engine that will be processed by the features module) since the feature processing of 1500 tweets took too much time on our platform (more than 30 min on a dual core machine at 2,8 GHz).

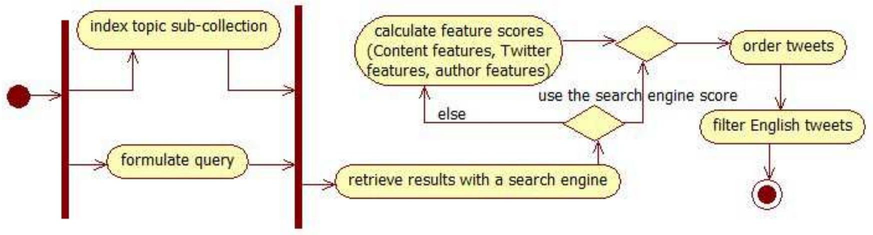
index topic sub-collection
formulate query
calculate feature scores (Content features, Twitter features, author features)
order tweets
use the search engine score
else
filter English tweets
retrieve results with a search engine

Fig. 1. UML activity diagram summing up of our approach

2.1.2 Indexing and topic search

We choose to use the Lucene platform[4] in our approach. This open source software is a high-performance, full-featured text search engine library written entirely in Java. In addition,

[4] http://lucene.apache.org/java/docs

it provides access to powerful term boosting, indexing and predefined searching features. The tokenizer was modified by organizers to preserve hashtags and mentions (i.e., words starting with the specific characters of Twitter: #,@). Nevertheless, they specified a field in the index for the *HTTPSTATUSCODE* of the tweets. The reason behind doing that is to be able to specify which type of tweets would be retrieved. Since it was announced that retweets will not be considered as relevant this year, we retained only the original tweets with *HTTPSTATUSCODE* equals to 200. Thus, we dropped out all other type of tweets (*HTTPSTATUSCODE* equals to 302, 403, and 404). However, we did not process the data to remove the implicit retweets (*HTTPSTATUSCODE* equals to 202 and starting with RT) since they could contain extra information.

Besides using the configuration aforementioned, we used term-boosting for expanding topics with keywords. We exploited it to give more importance to the topic terms and to avoid losing the initial purpose of the topic.

In the rest of the paper, the relevance score obtained by the Lucene search engine for a tweet t and a query q is denoted by $Lucene(t, q)$. The corpus of top-N relevant tweets obtained by the Lucene search engine regarding a topic q is denoted by T_q . Finally, C_q denotes the corpus of all tweets published before timestamp of a topic q. ($T_q \subseteq C_q$).

Finally, apart from indexing and retrieving, we used the Lucene search engine to calculate some feature scores. We address this in more details when explaining features in the next section.

2.1.3 Factor descriptions

In the context of microblog search, there is a set of new criteria of the new media that are imposed into the issue of retrieval. Relevance surely depends on which task the user is trying to complete. Considering that, it seems crucial to us to find and add new factors such as authority, quality, informativeness, temporality, and so on.

In our approach, we use three factors: the content factor, the Twitter factor and the author factor. Each factor is computed through a set of associated feature scores.

Content features. We used 3 content-specific features which are relative to some microblogs specificities: the wide variety of topics discussed by authors (1), the shortness of microblogs (2), and the wide variety of expressions used by authors (3).

- *Tweet popularity:* this feature estimates the popularity of a current tweet in T_q. We made the assumption that a tweet is popular if we find the same content in many other tweets. The similarity between a pair of tweets is calculated using the Lucene content-based similarity $sim(t_i, t_j)$. We denote the current tweet by t_i. The resulting formula to evaluate this feature is:

$$f_1(t_i, q) = \frac{\sum_{t_j \in T_q, i \neq j} sim(t_i, t_j)}{|T_q| - 1} \tag{1}$$

- *Tweet length:* instinctively, the lengthier a sentence is, the more information it contains. We calculate this feature by counting the number of words it contains. We denote $l(t_i)$ as the number of words a tweet t_i in T_q contains. This feature is defined as:

$$f_2(t_i, q) = \frac{l(t_i)}{max_{t_j \in T_q} l(t_j)} \tag{2}$$

- *Exact term matching:* this feature is used to promote tweets that contain terms of the topic q. $nb(t_i, q)$ denotes the number of times a term of the tweet t_i exist in the topic q:

$$f_3(t_i, q) = \frac{nb(t_i, q)}{max_{t_j \in T_q} nb(t_j, q)} \tag{3}$$

Twitter features. We consider 3 additional Twitter-specific features that may indicate the quality of the information shared through tweets.

- *URL presence:* by sharing an URL, an author would confirm the information published in his/her tweet or draw the attention of his/her followers to contents on the web. Thus, we believe that it could indicate informativeness. It is a binary feature:

$$f_4(t_i, q) = \begin{cases} 1 & \text{if } t_i \text{ contains url} \\ 0 & \text{if not} \end{cases} \tag{4}$$

- *URL frequency:* this feature aims to calculate how popular the URLs published in a tweet in the corpus C_q. $frq(url)$ denotes the number of time the url appear in the corpus C_q.

$$f_5(t_i) = \sum_{url \in t_i} frq(url) \tag{5}$$

- *Hashtag:* The # symbol, called a hashtag, is used to mark keyword or topic in a Tweet. Any Twitter user can categorize or follow topics with hashtags. It gives information to link tweets describing the same event or place to a group. $freq(h)$ denotes the frequency of a hashtag in the corpus C_q:

$$f_6(t_i) = \sum_{h \in t_i} frq(url) \tag{6}$$

Author features. To include the authority of the tweet author in the score function, we consider two author-specific features:

- *Number of tweets:* The purpose of this feature is to promote tweets published by active authors compared to tweets published by someone less active. $a(t_i)$ denotes the author of the tweet t_i. $N(a(t_i))$ is the number of tweets published by the author of the tweet t_i in the corpus C_q.

$$f_7(t_i) = N(a(t_i)) \tag{7}$$

- *Mention:* the more an author has been mentioned, the more popular he/she is.$M(a(t_i))$ denotes how many time the author of the tweet t_i has been mentioned in the corpus C_q.

$$f_8(t_i) = M(a(t_i)) \tag{8}$$

2.1.4 Formulate topics

Since microblogs are too short to express clearly their subjects, and queries are too simple to give the best representation of the information need, we found the idea of expanding topics a good solution to improve our chances of retrieving relevant tweets while containing different words from the topic.

The idea is to extract keywords from news articles published before the timestamps of a given topic. To do that, we used two APIs in which we could specify the topic and time period of the articles to extract: NYTimes API [5] and Guardian API [6]. Since the articles are chronologically retrieved by the APIs, for each topic we took the 2 first articles from each source to produce a mega-document [5]. Then we used the Alchemy API [7] to extract the top-5 keywords. Alchemy API uses deep linguistic parsing, statistical natural language processing, and machine learning to analyze the content and extract semantic metadata.

[5] http://developer.nytimes.com
[6] http://www.guardian.co.uk/open-platform
[7] http://www.alchemyapi.com

2.1.5 Scoring

Our scoring function is evaluated in two steps. The first step consists in evaluating the score of each feature (9) and the second step involves the computation of the final score (10). All the feature scores are normalized to lie between 0 and 1. We adopted a linear function for

$$features(t_i, q) = f_1(t_i, q) + f_2(t_i, q) + f_3(t_i, q) + f_4(t_i) + f_5(t_i) + f_6(t_i) + f_7(t_i) + f_8(t_i) \quad (9)$$

After normalizing the feature scores, the final score is calculated as:

$$score(t_i, q) = lucene(t_i, q) + features(t_i, q) \quad (10)$$

with $score(t_i, q) \in [0, 2]$

2.2 Results

We submitted 2 runs to TREC Microblog Track 2011: iritfd1 and iritfd2. The only difference between our two runs is that in iritfd1 we expanded topics with keywords from news articles, while in iritfd2 we did not. Thus, Run iritfd2 is thus without future or external evidence, contrary to run irifd1 that is with external but not future evidence. The same function (10) was used to evaluate the score of tweets in both runs.

To assess the contribution of the features used in our approach, we also created a post-hoc baseline run (iritfd0) where relevance is based only on scores obtained by the Lucene search engine.

For iritfd1 and iritfd2, we set our parameters as follows:

- *MIN* (minimum score of the tweets): 0.7;
- *T* (the number of terms added to the topic in iritfd1): 5;
- *N* (the number of results returned by the search engine that will be processed by the features module): 1,500.

For each topic, the system provides a chronologically ranked list of tweets (up to 1,000). The number of proposed topics was 50 but only 49 were judged. Two types of relevance judgments were calculated by organizers: allrel and highrel. The difference between them is that in highrel only the highly relevant tweets have been considered. Consequently, the number of topics in this case was only 33.

Table 1 summarizes our results. We present scores of MAP and P@30 for the both types of relevance judgements (i.e., allrel and highrel), the improvement of iritfd2 compared to iritfd0 (improvement iritfd2/iritfd0), and the improvement of iritfd1 compared to iritfd2 (improvement iritfd1/iritfd2).

We recall that iritfd2 uses features scores and thus can be directly compared with iritfd0 which did not, and that iritfd1 uses query expansion as well as features scores and thus can be directly compared to iritfd2.

Regarding the evaluation of iritfd0 and iritfd2, features clearly help to retrieve relevant tweets: the P@30 is improved by 90,49% and the MAP by 24,51% (difference statistically significant for the two cases).

Compared to iritfd2, The MAP of iritfd1 has improved by 10,20% in allrel and 19,29% in highrel. However we could not justify this improvement with the topic expansion since the difference between the two runs was not statistically significant (i.e., $p > 0.05$ with a paired Students t-test). More experiments are needed to fully understand the effect of expansion. Note that the degradation of the P@30 in highrel judgments compared to allrel judgments is more important than the MAP degradation for the same considered judgments.

This observation is valuable for the majority of the systems (the average of median P@30 is decreased from 0.2591 in allrel to 0.0686 in highrel). It could be explained by the fact that MAP is less sensitive to the number of relevant documents than the P@n.

Compared with the median across all runs submitted by any team, iritfd1 outperformed the P@30 for 27 out of 49 and the median AP for 38 out of 49 topics.

	allrel				highrel			
	P@30		MAP		P@30		MAP	
Iritfd0(baseline)	0,1346		0,1558		0,041		0,1179	
Iritfd2	0,2564		0,1940		0,0960		0,1622	
Improvement iritfd2/iritfd0	90,49%	*	24,51%	*	134,14%	*	37,57%	*
iritfd1(topic expansion)	0,2605		0,2138		0,0970		0,1935	
Improvement iritfd1/iritfd2	1,59%		10,20%		1.04%		19,29%	

Table 1. Summary of the results. * indicates a significant improvement ($p \leq 0.05$ with a paired two-tailed Students t-test)

2.3 Discussion and future works

We proposed in this section an approach that combines a set of features to rank real time microblogs. The specificity of our work is that we calculate the tweet scores by keeping in mind 3 important factors which are content features, Twitter features, and author features. These factors were calculated by combining a set of features linearly. We also proposed a topic expansion approach. However, we could not conclude about its effectiveness. There is still a lot of room for improvement. We need to evaluate the influence of each feature independently. We will test the effect of the number of keywords added to the topic, which may be detected with a more suitable topic expansion approach.

3 A Bayesian network retrieval model for tweet search

We present in this section a second tweet search approach based on Bayesian networks that integrates several relevance features namely hashtags, tweet time magnitude, tweet length and the social influence of microbloggers.

3.1 Tweet Search Model

We propose to model tweets using Bayesian networks. Such representation allows to model influenceable sources of evidence though conditional probabilities. We present in what follows the Bayesian network topology then we focus on query evaluation precess.

3.1.1 Network topology

The Bayesian network for tweet search is represented by a graph $G = (X, E)$, where nodes $X = \mathcal{Q} \cup \mathcal{K} \cup \mathcal{T} \cup \mathcal{U}$ corresponds to the set of random variables and the set of edges $E = X \times X$ represents conditional dependencies between them. $\mathcal{Q}$, $\mathcal{K}$, $\mathcal{T}$ and $\mathcal{U}$ correspond respectively to the sets of queries, terms, tweets and microbloggers.

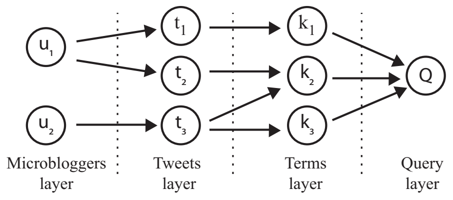

Fig. 2. The Bayesian network of tweet search

Information nodes. Bayesian network nodes are classified into 4 layers. Figure 2 presents information nodes and relationships between them.

- *The query layer* $\mathcal{Q}$ includes the query node q which is associated to a binary random variable $q \in \{0, 1\}$. The short representation of $q = 1$ is noted q and denotes "the query q is observed". Conversely, $q = 0$ is noted $\bar{q}$ and denotes "the query q is not observed". We notice that the same notation q is used to refer to the query, the associated random variable and the query node. The same notation is used for other nodes in the network.
- *The terms layer* $\mathcal{K}$ includes nodes of terms present in the tweet index. A binary random variable $k_i \in \{0, 1\}$ is associated to each term k_i.
- *The tweets layer* $\mathcal{T}$ includes tweet nodes. A binary random variable $t_j \in \{0, 1\}$ is associated to each tweet t_j.
- *The microbloggers layer* $\mathcal{U}$ includes microblogger nodes. A binary random variable $u_k \in \{0, 1\}$ is associated to each microblogger u_k.

Information edges. Each edge in the network express a conditional dependency between nodes. Edges connecting the query $q \in \mathcal{Q}$ with parent terms $k_i \in \mathcal{K}$ represent the chance of generating the query from connected term. Each term k_i is connected to parent tweets $t_j \in \mathcal{T}$ that it indexes. Edges from tweets to terms express that the event of observing a particular tweet impacts the observation of the connected term. Finally, a tweet node $t_i \in$ is connected to a single parent node corresponding to the microblogger $u_k \in \mathcal{U}$ having published t_j. This edge shows that the event of observing a tweet t_j depends on the observation event of the corresponding microblogger u_k. To avoid cycles in the graph, we assume that tweets and microbloggers are mutually independent between each other.

3.2 Query evaluation

The relevance of a tweet t_j considering a query q is assimilated to the joint probability that both events $t_j = 1$ and $q = 1$ appear. This probability is computed as:

$$P(q \wedge t_j) = \sum_{\forall \boldsymbol{k}} P(q|\boldsymbol{k}) P(u_k) \prod_{\forall i | on(i,\boldsymbol{k})=1} P(k_i|t_j) \times \prod_{\forall i | on(i,\boldsymbol{k})=0} P(\bar{k_i}|t_j) \tag{11}$$

$\boldsymbol{k}$ is a query parent configurations defined by a vector of random variables $\boldsymbol{k} = (k_1, k_2, ..., k_m)$, $k_i \in \{0, 1\}$. Considering a query $q = \{k_1, k_2\}$ composed of two terms k_1 and k_2, the set of query parent configurations is represented by $\{(k_1, k_2), (k_1, \bar{k_2}), (\bar{k_1}, k_2), (\bar{k_1}, \bar{k_2})\}$. $on(i, \boldsymbol{k}) = 1$ if $k_i = 1$ according to the query configuration $\boldsymbol{k}$ and $on(i, \boldsymbol{k}) = 0$ if $k_i = 0$.

Computing probability $P(q|\boldsymbol{k})$**.** The probability $P(q|\boldsymbol{k})$ of observing the query q having the configuration $\boldsymbol{k}$ helps to weight the different combinations of the query terms as follows:

$$P(q|\boldsymbol{k}) = \prod_{\forall i} on(i, k) \tag{12}$$

Computing probability $P(k_i|t_j)$**.** The probability $P(k_i|t_j)$ of observing a term k_i in the tweet t_j depends, on the one hand, on the term's occurrence and on the other hand on the tweet properties. This probability is computed using the term frequency $F(k_i, t_j)$, the hashtag presence $H(k_i, t_j)$, the time magnitude $T(k_i, t_j)$ and the tweet length $L(t_j)$:

$$P(k_i|t_j) = (1-\mu)F(k_i, t_j)\ H(k_i, t_j) + \mu\ T(k_i, t_j)\ L(t_j) \tag{13}$$

$$P(\bar{k_i}|t_j) = 1 - P(k_i|t_j) \tag{14}$$

with $\mu \in [0..1]$ is a smoothing parameter.

- *Term frequency* $F(k_i, t_j)$ replaces the common *tf* measure with a graduated function $F(k_i, t_j)$ that map high frequencies into a small interval:

$$F(k_i, t_j) \begin{cases} 1 - \frac{a}{tf_{k_i,t_j}}, & \text{if } k_i \text{ is present in } t_j \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

with $a \in [0..1]$ and tf_{k_i,t_j} is the frequency of the term k_i in the tweet t_j.
- *Hashtag score* $H(k_i, t_j)$ leverages the importance of hashtagged terms as follows:

$$H(k_i, t_j) \begin{cases} 1 - \frac{b}{tf_{\#k_i,t_j}}, & \text{if } \#k_i \text{ is present in } t_j \\ b, & \text{otherwise} \end{cases} \tag{16}$$

$b \in [0..0.5]$ is the default hashtag score. $tf_{\#k_i,t_j}$ is the frequency of the hashtag $\#k_i$ in t_j.
- *Time magnitude* $T(k_i, t_j)$ of tweet t_j depends on its submission time. This probability would be more important when term k_i is frequently used at tweet submission period. The time magnitude is estimated as follows:

$$T(k_i, t_j) = \frac{df_{k_i,\Gamma_j}}{|\Gamma_j|} \tag{17}$$

$\Gamma_j = \{t_k, |\theta_{t_j} - \theta_{t_k}| \le \Delta t\}$ is the set of temporal neighbors of t_j within the $2\Delta t$ time window. df_{k_i,Γ_j} is the number of tweets in Γ_j containing k_i.
- *Tweet length* $L(t_j)$ score highlights tweets closer to the average tweets length avg_{tl}. The tweet length score of a tweet t_j with a length tl_{t_j} is computed as follows:

$$L(t_j) = \frac{1}{1 + |avg_{tl} - tl_{t_j}|} \tag{18}$$

Computing probability $P(t_j|u_k)$**.** The probability $P(t_j|u_k)$ of observing a tweet t_j knowing the corresponding microblogger u_k is computed as follows:

$$P(t_j|u_k) = \frac{1}{|\mathcal{T}_{u_k}|} \tag{19}$$

$\mathcal{T}_{u_k}$ is the set of tweets published by the microblogger u_k

Computing probability $P(u_k)$**.** The probability $P(u_k)$ is interpreted as the influence of microblogger u_k on the retweet social network. This network is modeled by a graph $G = (U, R)$ where U is the set of microbloggers having published at least one tweet that contains a query term and $R = U \times U$ denotes the set of retweet relationships. A retweet relationship $(u_i, u_j) \in R$ is defined from u_i to u_j if u_i retweeted a tweet of u_j. The influence of a microblogger u_i is estimated by applying *PageRank* algorithm on the retweet network as follows:

$$Inf^p(u_i) = \frac{d}{|U|} + (1-d) \sum_{u_j \in U, u_j \to u_i} w_{i,j} \frac{Inf^{p-1}(u_j)}{O(u_j)} \tag{20}$$

$O(u_j)$ is the outdegree of node u_j. d is the random walk parameter.

3.3 Corpus indexing and tweet filtering

Tweets are indexed using NESTOR microblogging retrieval platform developed by our team. Indexing process separates URLS from tweet message and indexes only textual content. This system supports multi-language tokenisation and uses the Porter stemming algorithm for recognized English text. Moreover, the tweeting features such as retweets, mentions and hashtags are extracted and integrated as meta-data. Finally, conventional retweets starting with *"RT @username"* are detected and used in addition to Twitter native retweets.

The retrieval process is conducted with respect to the requirements of the realtime adhoc task. In fact, posterior tweets to query time are discarded by applying a time constraint when extracting tweet from inverted-index. We note also that the social retweet network is built from posterior tweets and no future information is used in the query evaluation process. avg_{tl}, Γ_j and $T(u_k)$ consider only posterior tweets to the query time. After computing relevance score $P(q \wedge t_j)$, the tweet results is filtered as follows:

- We remove all retweets
- We remove non English tweets expect mixed-language tweets where English is the principal language. Tweet language is detected using the text processing library MorphAdorner[8].
- We remove tweet including less than $\frac{|q|}{2}$ terms. $|q|$ is the query length. This helps to reduce noise in the final result set ranked by tweeting time.

3.4 Results and discussion

Tables 2 compares results obtained by different configurations of our model:

- *Nestor:* Proposed model with all features included.
- *Nestor-S:* Proposed model with social feature disabled $P(u) = 1$.
- *Nestor-T:* Proposed model with temporal feature disabled $T(k_i, t_j) = 1$.
- *Nestor-L:* Proposed model with tweet length feature is disable $L(u) = t_j$.

Experiments are conducted with the next parameters values: $\mu = 0.25$,$a = 0.25$, $b = 0.4$, $\Delta t = 1h$ and $d = 0.15$.

	Ranked by time				Ranked by score	
	All rel		High rel		All rel	
	p@30	MAP	p@30	MAP	p@30	MAP
Nestor*	0.2027	0.1305	0.0838	0.1287	0.2218	0.1384
Nestor-S*	0.2027	0.1305	0.0838	0.1286	0.2184	0.1360
Nestor-T	0.2082	0.1343	0.0585	0.0912	0.1912	0.1196
Nestor-L	0.2048	0.1306	0.0565	0.0867	0.2293	0.1426
TREC median	0.2592	0.1433	0.2646	0.1381		

Table 2. Comparison of model configurations. * Official run

Comparing overall results, the proposed model shows low performances compared to *TREC median.* However, we can draw some primary conclusion form obtained results. In comparison to the similar p@30 values shown by *Nestor* and *Nestor-S* models with tweet

[8] http://morphadorner.northwestern.edu/

time ranking, we note a slight decline of *Nestor-S*, where social feature is disabled, compared to *Nestor* model in the case where tweets are ranked by score. We conclude that the social context impacts the tweet relevance. Moreover, we note through the decline of *Nestor-T* performances observed with score ranking, that the time feature can improve the retrieval effectiveness. However, opposite behavior observed with time ranking shows that the time magnitude may affect top list ranking with some irrelevant tweets. Finally, best p@30 values presented by *Nestor-L* allow to conclude that average tweet length is not an appropriate feature for tweet search.

3.5 Conclusion and future work

We proposed in this work a tweet search model that integrates several features within a Bayesian network model namely the time magnitude, hashtags, tweet length and the social influence of microbloggers. Obtained results show that some features are useful for tweet search. Other ones show promising performances that need to be improved. In future work, we plan to conduct extended experiments analyzing the impact of each feature and also to model the temporal aspects of tweets within the Bayesian network.

References

1. Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 295–303, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
2. Rinkesh Nagmoti, Ankur Teredesai, and Martine De Cock. Ranking approaches for microblog search. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 153–157, Washington, DC, USA, 2010. IEEE Computer Society.
3. Anish Das Sarma, Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Ranking mechanisms in twitter-like forums. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 21–30, New York, NY, USA, 2010. ACM.
4. Lulin Zhao, Yi Zeng, and Ning Zhong. A weighted multi-factor algorithm for microblog search. In *Proceedings of the 7th international conference on Active media technology*, AMT'11, pages 153–161, Berlin, Heidelberg, 2011. Springer-Verlag.
5. Claus-Peter Klas and Norbert Fuhr. A new Effective Approach for Categorizing Web Documents. In *Proceedings of the 22th BCS-IRSG Colloquium on IR Research*, April 2000.