Team COMMIT at TREC 2011

Marc Bron Edgar Meij Maria-Hendrike Peetz

Manos Tsagkias Maarten de Rijke

ISLA, University of Amsterdam
http://ilps.science.uva.nl/

Abstract: We describe the participation of Team COMMIT in this year's Microblog and Entity track.

1 Introduction

Team COMMIT participated in two tracks this year: the Microblog track and the Entity track.

In our participation in the Microblog track, we used a feature-based approach. Specifically, we pursued a precision oriented recency-aware retrieval approach for tweets. Amongst others we used various types of external data. In particular, we examined the potential of link retrieval on a corpus of crawled content pages and we use semantic query expansion using Wikipedia. We also deployed pre-filtering based on query-dependent and query-independent features. Our main finding is that cutting-off the result list is difficult and crucial for good results. We also found that using external data helps with recall and precision.

For our participation in this year's Entity track we focused on the Entity List Completion (ELC) task. We experimented with a text-based and a link-based approach to retrieving entities from Linked Data (LD). Additionally, we experimented with selecting candidate entities from a web corpus. Our hypothesis is that entities occurring on pages with many of the example entities are more likely to be good candidates than entities that do not. Due to the absence of evaluation results at the time of writing, we have no preliminary conclusions yet.

The remainder of the paper consists of two largely independent sections, one for each of the tracks in which we participated, plus a conclusion.

2 Microblog Track

Our approach to the Microblog track assembles a list of feature values for each tweet and uses learning to rank to combine them.

2.1 Features

We have two classes of features: *query-dependent* and *query-independent* features. Query-dependent features aim at capturing different aspects of relevancy of the tweet to the query. Query-independent features target at encoding the usefulness of a tweet in the information seeking process, and can be thought as information quality indicators.

2.1.1 Query-dependent features.

We consider the following six query-dependent features: temporal query modeling, semantic query expansion, link retrieval, HITS scores, boolean, and Indri retrieval scores. We look at each feature in turn.

Our first feature, temporal query modeling, aims at capturing the dynamics of topics in Twitter [3]; the probability of a term given a query, $P(t|\theta_Q)$, is given by the weighted mixture of the original query Q and the expanded query $\hat{Q}$, controlled by parameter α. To construct the expanded query, we rank terms according to Eq. 1, and select the top k terms. This model tries to take into account the dynamic nature of microblogging platforms: while a topic evolves, the language usage around it is expected to evolve as well. Consequently, selecting terms temporally closer to query time could result in terms that are more relevant for that point in time. Term scoring becomes a function of time:

$$score(t,Q) = \log\left(\frac{|N_c|}{|\{d : t \in d, d \in N_c\}|}\right) \cdot \sum_{\{d \in N_c : q \in Q \text{ and } t,q \in d\}} e^{-\beta(c-c_d)}, \quad (1)$$

where c is query submission time, c_d is post d's publication time, N_c is the set of posts that are posted before time c and q. The parameter β controls the contribution of each post to the score of term t based on their posted time. We select only those terms as candidate expansion terms, that occur in more than φ posts. In our experiments we set $k = 10$, $\beta = 0.01$, $\varphi = 10$. The feature value is the retrieval score $score(t,Q)$.

As a second query modeling feature, we use semantic query modeling (SQM) [**?**]. SQM leverages the anchor texts of incoming hyperlinks to Wikipedia articles, including not only "normal" hyperlinks, but also redirects, and alternative

titles of pages. It does so in two steps. First, the anchor texts are used to identify, and score relevant Wikipedia articles for all possible term n-grams in a query. Then, for each of these articles, we again use the incoming anchor texts. In this step, however, we use them to determine the parameters of a language model for each article. The language models are subsequently combined for all n-grams in the query, yielding as end result a semantically-informed language model of the query, i.e., a semantic query model. The feature value is the retrieval score for the expanded query and the tweet.

Our third feature is link retrieval. We extracted, and crawled all links from the tweets after resolving the link's original URL if it was shortened. Some links were not accessible either due to URL "unshortening" services blocking pages with doubtful content, or because the link was simply broken. For those links we ended up having content for, we indexed them using Indri. We use language modeling and SQM to model queries, and fire them against this link index. The retrieved links are mapped back to the tweets that contain them, producing a ranking of tweets. The feature value score for a tweet is the retrieval score for the link.

We also consider the retrieval scores of different rankers, such as strict boolean matching of the query terms, and Indri's retrieval model.

Our sixth, and final, query-dependent feature is based on HITS scores. We start with retrieving a set of tweets for a query, using standard language modeling. Then, we create a weighted directed graph, with the nodes being the authors of the retrieved tweets. The edges denote the number of times a user retweeted another user's tweet. Based on this graph, we calculate the HITS authority and hub scores for every user in the graph.

2.1.2 Query-independent features.

Next we discuss our query-independent features: *textual features* and *Twitter-specific features*.

We use *textual features* to decide the potential interestingness of a tweet [4]. Following [5], we believe that the ratio of capital to lowercase letters in a tweet is important: tweets with only capital letters (shouting) and all lowercased tweets indicate a different author profile than tweets with correct casing. Another measure of interestingness can be the language density in a tweet, namely, the sum of $tf \cdot idf$ values of non-stopwords, divided by the number of stopwords they are apart, squared [2]. We trained a spam classifier using semi-manually selected spam tweets (making use of the `@spam` hashtag) and assigned each tweet a spam value. We also included features that encode whether a tweet is a question or an exclamation, and the number of links in the tweet.

Our second set of query-independent features is *Twitter-specific features*. Some inherent characteristics of tweets include the use of hashtags, the mention of usernames in a tweet and the possibility to retweet a tweet. We encode these characteristics into features, such as whether a tweet is a direct message, the number of hashtags, and the number of usernames in a tweet. We also exploit Twitter's network properties. In particular, we set up a graph similar to the HITS graph mentioned above using all tweets in the collection. We compute the PageRank scores of the nodes (authors) and use them as a feature for tweets. These scores denote the importance of a user by their likelihood of being retweeted. Other network features we take into account include the number of friends and followers, and the number of tweets a user uttered before the current tweet.

2.1.3 Combination of features.

Query dependent features (except for the HITS feature) were combined together using a learning to rank (LR) method, trained on an in-house developed set; see below. The query independent features (including the HITS feature) were used for learning over a set of relevant tweets, useful for filtering out low quality tweets before retrieval. Here, we used the relevance assessments for our training set to create a set of tweets that were labeled as relevant and non-relevant. Based on that, we trained a Random Forest classifier, and kept only the tweets in the collection classified as relevant. Given each ranked result list of tweets, we calculated the z-score of the scores. All tweets with a z-score greater than ζ were in the final result set.

2.2 Runs

We submitted four runs, as listed in Table 1. For all runs, we use language identification [**?**] to identify, and keep only the English language tweets in the collection. Duplicate tweets are removed, and the oldest tweet in the set of duplicates is kept. Retweets are also removed, however, we store the retweet information. In ambiguous cases, e.g. where comments were added to a retweet, the tweet is kept. Hashtags remain in the tweet as simple words, i.e. we simply removed the leading hashtag. Finally, we performed punctuation and stop word removal, based on a collection based stop word list. To prevent future information from leaking into our collection, we created separate indexes for every query.

For training we developed an in-house dataset capturing the period January 1st, 2011 to March 15, 2011. The dataset consists of 39 queries, with their timestamps for a dataset of 12.7 million tweets. For each query, we selected tweets published before the timestamp of the query. Each query had 100 annotated tweets. To prevent leaking information, the sizes of the training sets vary between 2,363 and 1,422 tweets, depending query time .

Our baseline (COMMITbase) uses temporal query expansion and returns only tweets that contain a link. The second run (COMMITexp) uses learning to rank with all query-dependent features, except for the link mapping: temporal query expansion, semantic query modeling, boolean matching, and language modeling. The third run (COMMITlinks) uses the same features as in COMMITexp, however it in-

Name	Description	External data
COMMITbase	TQM, no links	No
COMMITexp	LR with query dependent, no links and HITS	Yes
COMMITlinks	LR with query dependent, no HITS	Yes
COMMITfilter	Query independent and HITS filtering, then COMMITlinks	Yes

Table 1: Overview of our Microblog runs.

cludes the link mapping using SQM, and a standard language model on the link corpus. In our final run (COMMITfilter) we applied pre-filtering. The ranked lists from all runs are further curated. We normalize the retrieval scores using z-scoring, and discard tweets with low z-score. Based on manual inspection of the outcomes of our preliminary experiments on our training queries, we set $\zeta = 0.6$. All runs except the baseline use external data.

While P@30 is the official metric, we also look at other metrics to better understand the performance of our systems. We report on MAP and precision at 5 (P@5) and 30 (P@30). We use the Student's t-test to evaluate statistical significance, and denote statistically significant increase in performance with $^{\blacktriangle}$and $^{\triangle}$($p < 0.01$ and $p < 0.05$ respectively). Likewise, $^{\triangledown}$and $^{\blacktriangledown}$denote a statistically significant decrease.

2.3 Results

In this section, we report on the performance of our runs, and we discuss two main points of interest for our participation in the Microblog track: a) whether retrieving links of a tweet as a feature increases performance, and b) the effect of pre-filtering of tweets.

Table 2 lists the results for our four runs, for two relevant sets: a) all relevant, and b) highly relevant. A general comment is that COMMITexp, COMMITlinks, and COMMITfilter show similar performance in all metrics. Compared to the baseline (COMMITbase) all show marginally higher P@5 (not statistically significant), and lower P@30 and MAP (statistically significant). Looking at the reasons of this performance differences, we find that all runs return on average less tweets than the baseline (20 vs. 135). By cutting off very early, we do not retrieve enough tweets. Essentially, with the scores of the runs other than the baseline are clustered around the mean and do not have such a high variance. In future work, we aim at defining ζ as a function of the scores shape distribution.

Looking at the results of our three runs, we find that in very tight cut-off thresholds, using more corpora has no effect on P@5, but query expansion such as SQM, does not decrease recall. COMMITfilter shows that filtering in tight cut-off thresholds does has a negative effect in performance. In some cases there are less that 5 tweets returned, and further filtering on those small sets is likely to decrease P@5 and recall. We can however see a small improvement (though not significant) from COMMITexp to COMMITlinks in MAP. A possible explanation is the large number of tweets in the evaluation set with at least one link (70% and 80% of relevant and highly relevant tweets, respectively), which shows to help increase recall. Retrieving highly relevant tweets is more difficult. COMMITlinks has a higher precision than the baseline, however not for two queries. Indeed, for those two queries, we failed to retrieve the one or two relevant tweets. Obviously, the cut-off procedure is too strict again: returning more tweets would increase recall and thus MAP. We believe P@30 may not be the right metric for the task, given the small number of average highly relevant tweets (11). Our runs show an increase in P@5 and P@10, stemming from a strict cut-off criterion aiming at selecting highly relevant tweets. Similar to retrieving all tweets, we have an improvement using link mapping, indeed a higher percentage of highly relevant tweets in the ground truth have links ($\sim$80%).

3 Entity Track

In this year's edition of the Entity Track we focus on the Entity List Completion (ELC) task. In the ELC task the goal is to find entities in structured data given a source entity, relation, target type and example entities.

3.1 Entity List Completion Approach

The corpus consists of a new Linked Data (LD) crawl. To process and index the data code was provided for two methods of indexing: entity centric (ED) and document centric (DE). In our experiments we use the entity centric index. The LD crawl consists of RDF triples, where an RDF triple consists of a "subject," "predicate" and an "object." A subject is always a URI and represents a "thing" (in our case: an entity). Subject URIs serve as unique identifiers for entities. An object is either a URI referring to another "thing" or a string , holding a literal value. Predicates are also always URIs and represent relations between subjects and objects. The entity centric representation assumes an entity to be represented by the set of all triples that have the same URI as subject.

Entity resolution. Another change this year is that the URIs of the example entities are no longer provided in the topics. Instead the homepage and the name of example entities are given. To resolve entity names to URIs we use the anchor text in Wikipedia. Given an entity name (e) we find the set of all Wikipedia pages (WP) that have e as anchor

	All relevant			Highly relevant		
Run	P@30	MAP	P@5	P@30	MAP	P@5
COMMITbase	0.4279	0.2746	0.5551	0.0952	0.1422	0.1633
COMMITexp	0.3034$^{\blacktriangledown}$	0.1502$^{\blacktriangledown}$	0.5633	0.0626$^{\triangledown}$	0.1071	0.1755
COMMITlinks	0.3082$^{\blacktriangledown}$	0.1519$^{\blacktriangledown}$	0.5633	0.0646$^{\triangledown}$	0.1081	0.1755
COMMITfilter	0.2946$^{\blacktriangledown}$	0.1465$^{\blacktriangledown}$	0.5551	0.0592$^{\blacktriangledown}$	0.1035	0.1755

Table 2: Results for our runs.

text. We then pick the Wikipedia page that is most commonly referred to by a link with anchor text e:

$$\arg\max_{wp \in WP} \text{COMMONNESS}(e, wp),$$

where $\text{COMMONNESS}(e, wp)$ is defined as:

$$\text{COMMONNESS}(e, wp) = \frac{|L_{e,wp}|}{\sum_{wp'} |L_{e,wp'}|},$$

and $|L_{e,wp}|$ indicates the number of times entity name e is linked to Wikipedia page wp. To map the URL of a Wikipedia page to a URI in the LD crawl we replace the `http://en.wikipedia.org/wiki/` part of the URL by `http://dbpedia.org/resource/`. In the following we only consider example or candidate entities that have been successfully mapped to a DBpedia URI.

Text-based approach. The text-based approach follows the intuition that entities with links to objects that have overlapping terms with the narrative and source entity should be ranked higher than entities that do not. To represent entities we use the terms present in the entity centric representation. Entities are indexed based on the terms occurring in the literals and URIs in each entity representation. We rank entities according to the similarity between the entity's textual representation (e_d) and a query (Q) containing the source entity name (E_{text}) and relation (R). The similarity is calculated using the vector space model with standard tf-idf term weighting:

$$sim_{text}(e, Q) = \frac{\vec{V}(e_d) \cdot \vec{V}(E_{text}R)}{|\vec{V}(e_d)| \cdot |\vec{V}(E_{text}R)|},$$

where the numerator is the dot product of the vectors, $|\vec{V}|$ is the length of $\vec{V}$ and $\vec{V}(E_{text}R)$ is the term vector containing the terms from E_{text} and R.

We additionally filter entities retrieved by the text-based approach on type. The type is given in the topic, e.g., Company. We construct a URI by appending the type to `http://dbpedia.org/ontology/`. Only entities that have this URI as an object in their representation are kept, otherwise they are removed from the ranking.

Link-based approach. The link-based approach follows the intuition that candidate entities that are more similar to the example entities should be ranked higher than entities that are less similar. In this setting we use the links in the entity centric representation, i.e., an entity representation consists of all RDF triples which have the entity's URI as subject (i.e., outlinks) or object (i.e., inlinks). Together these triples form the link-based representation of an entity ($e_l = \{r_1, \ldots, r_m\}$, where r_i is an RDF triple).

Under this representation, entities consist of sets of triples. The set of example entities becomes a set of sets of triples ($X = \{x_1, \ldots, x_n\}$ and $x_i = \{r_1, \ldots, r_k\}$). We rank entities according to the similarity between the entity's link-based representation e_l and a query (Q) containing the set of example entities (X). The similarity is calculated according to the weighted Jaccard similarity between the triples of an entity and the triples of the example entities:

$$sim_{link}(e, Q) = \frac{\sum_{r \in \bigcup_{x \in X}(x \cap e_l)} w(r)}{\sum_{r \in \bigcup_{x \in X}(x \cup e_l)} w(r)},$$

where $\bigcup_{x \in X}(x \cap e_l)$ is the union of the triples that the examples in X and e_l have in common, $\bigcup_{x \in X}(x \cup e_l)$ is the union of all the triples in X and e_l, and $w(r)$ is a weight function that determines the importance of a link. We set the weight proportional to the number of times a triple occurs in the representation of the example entities:

$$w(r) = \max\left(1, \sum_{x \in X} n(r, x)\right)$$

Here $n(r,x)$ is 1 if a triple r occurs in the representation of example x and 0 otherwise.

Candidate selection from a web corpus. Both the text and link-based approaches consider all entities in the LD crawl as candidate entities and rank them either based on the textual representation or overlap with the example entities. Given the sparse nature of LD, differentiating between candidates becomes more difficult when the number of candidates increases. To limit the set of candidates we propose to only consider entities that occur on webpages associated with the source entity.

To find webpages associated with the source entity we look at three types of pages: pages that are part of the homepage of the source entity (HP); Wikipedia pages that link to the homepage of the source entity (WP_l); and Wikipedia pages that are relevant to a query (WP_q), where the query consists of the source entity (E_{text}) and the relation (R).

To obtain candidate entities we apply a named entity recognizer to these pages. To further limit the set of candidate

entities we rank each page in the HP and WP_l set based on the number of example entities that occur on the page:

$$\text{rank}(\text{page}) = |\{e : e \in C_{\text{page}} \cap e \in X_{\text{names}}\}|,$$

where X_{names} is the set of names of the example entities and C_{page} is the set of entities that occur on page. To rank the WP_q pages we use the query likelihood, i.e., the probability that the query q (E_{text} and R) is generated by a Wikipedia page:

$$P(q|\theta_{wp}) = \prod_{t \in q} P(t|\theta_{wp})^{n(t,q)},$$

where θ_{wp} is the language model representation of wp, and $n(t,q)$ is the number of times t occurs in q. We chose the top 5 highest ranked pages from each source and use the entities in those pages as candidates.

3.2 Runs and Results

We submitted the following four runs:

ilpsTextFilt This run is created using the text-based method to find entities, i.e., querying the entity centered index with the relation and source entity. Entities are filtered based on the DBpedia type specified in the topics.

ilpslinkcand This run is created using the link-based method; entities are ranked based on the link overlap with the example entities.

ilpslinkOL This run also uses the link-based method to rank entities. Candidate entities are harvested from 3 sources: the homepage domain of the source entity, pages linking to the source entity homepage domain, and DBpedia pages relevant to the narrative and source entity.

ilpsPMIcMNZ Our final run combines the ranked lists from the text and the link based runs. Instead of using standard COMBMNZ we weigh candidates found by each of the methods by co-occurrence in the homepage.

At the time of writing of these working notes no evaluation results are available yet for the ELC runs.

4 Conclusion

In this paper we reported on our participation in the Microblog and Entity tracks. For the Microblog track we found that a simple cut-off based on the z-score is not sufficient: for differently distributed scores, this can decrease recall. A well set cut-off parameter can however significantly increase precision, especially if there are few highly relevant tweets. Filtering based on query-independent filtering does not help for already small result list. With a high occurrence of links in relevant tweets, we found that using link retrieval helps improving precision and recall for highly relevant and relevant tweets. Future work should focus on a score-distribution dependent selection criterion. For the Entity track there are no analyses or conclusions to report yet; at the time of writing no evaluation results are available for the Entity track.

Acknowledgments

This research was partially supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the PROMISE Network of Excellence co-funded by the 7th Framework Programme of the European Commission under grant agreement nr 258191, the LiMoSINe project co-funded by the 7th Framework Programme of the European Commission under grant agreement nr 288024, the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.-814, 612.061.815, 640.004.802, 380-70-011, 727.011.005, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP project funded by the CLARIN-nl program, under COMMIT project Infiniti and by the ESF Research Network Program ELIAS.

5 References

[1] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[2] Lee, G. G., Seo, J., Lee, S., Jung, H., hyun Cho, B., Lee, C., Kwak, B.-K., Cha, J., Kim, D., An, J., Kim, H., and Kim, K. (2001). Siteq: Engineering high performance qa system using lexico-semantic pattern matching and shallow nlp. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*, pages 442–451.

[3] Massoudi, K., Tsagkias, E., de Rijke, M., and Weerkamp, W. (2011). Incorporating Query Expansion and Quality Indicators in Searching Microblog Posts. In *ECIR 2011: 33rd European Conference on Information Retrieval*, Dublin. Springer, Springer.

[4] Naveed, N., Gottron, T., Kunegis, J., and Alhadi, A. C. (2011). Bad news travel fast: A content-based analysis of interestingness on twitter. In *WebSci '11: Proceedings of the 3rd International Conference on Web Science*.

[5] Weerkamp, W. and de Rijke, M. (2008). Credibility improves topical blog post retrieval. In *Proceedings of ACL-08: HLT*, pages 923–931, Columbus, Ohio. Association for Computational Linguistics, Association for Computational Linguistics.