Query Expansion and Message-passing Algorithms for TREC Microblog Track

Dzung Hong, Qifang Wang, Dan Zhang, Luo Si

Computer Science Department, Purdue University
West Lafayette, IN 47907, USA

January 25, 2012

Abstract

This report describes the methods that our Information Retrieval Group at Purdue University used for the TREC Microblog 2011 track. The first method is the pseudo-relevance feedback, a traditional algorithm to reformulate the query by adding expanded terms to the query. The second method is the affinity propagation, a non parametric clustering algorithm that can group the top tweets according to their similarities. The final score of a tweet is based on its relevance score and the relevance score of its representative in the group. We found that query expansion is a very useful technique for microblog retrieval, while affinity propagation could achieve a comparable performance when combining with other techniques.

1 Introduction

The Information Retrieval Group at Purdue University participated in the TREC Microblog track. Our approach was to get the initial ranked list of tweets for each query using the Indri query language [Strohman et al., 2004]. We then applied a non parametric clustering algorithm called affinity propagation [Frey and Dueck, 2007] to group the top relevant tweets based on their similarities. The goal of affinity propagation was to identify the exemplar (i.e. the representative) for each tweet in the list. The score of the exemplar was then used to boost the scores of the tweets for which the exemplar represents. It is expected that doing so will increase the scores of those tweets similar to the query-relevant tweets, and lower the scores of those similar to the irrelevant ones.

2 Method

2.1 Indexing

In order to comply with the requirement of not using any future evidence, we chose to build different indexes for different queries. Specifically, for each query, we extract all the tweets posted before the issued time of that query, and build a separate index for those tweets. While this approach may require a large disk space, it is the method closest to what may occur in reality. In real systems, most indexes are built incrementally, and at one point of time, we can see only one copy of the index, which contains all the tweets appeared so far.

2.2 Query reformulation

We explored using the two following techniques to reformulate the query

1. **Exploring tags:** Many tweets contain tags, which are marked by the '#' symbol. We extracted those tags and used them as an additional type of information for the query.

2. **N-gram:** We grouped any two consecutive words in the query, and added them to the query.

For example, with the original query "BBC World Service staff cuts", our reformulated query in the Indri language would become

#weight(1 #combine(BBC.content World.content Service.content staff.content cuts.content) λ_1 #combine(BBC.tag World.tag Service.tag staff.tag cuts.tag) λ_2 #combine(#2(BBC World) #2(World Service) #2(Service staff) #2(staff cuts)))

where ".content" indicates that the words must appear in the text of the tweet, and ".tag" indicates that the words must appear as a tag. In our experiments, we set $\lambda_1 = 0.3$ and $\lambda_2 = 0.3$.

2.3 Query expansion

We applied the relevance feedback model [Lavrenko and Croft, 2001] for query expansion. According to this model, we select the expanded terms based of their goodness, which are estimated as follows.

$$P(t|Q) = \sum_{d \in \mathcal{F}} P(t|d) * P(d|Q)$$

where $\mathcal{F}$ is the set of feedback documents, usually chosen as the set of top documents for the query; $P(t|d)$ is the probability that the term t appearing in the document d, and $P(d|Q)$ is the probability that the document d is generated from the same language model as the query Q. We used the Laplace smoothing technique to estimate $P(t|d)$, i.e.

$$P(t|d) = \frac{tf_d(t) + 1}{tf_d(t) + K}$$

where $tf_d(t)$ is the term frequency of term t with respect to document d, and K is the total number of different terms in the feedback document set. The other term $P(d|Q)$ is estimated by the score of the document d with respect to the query Q. In this case, we used the Indri relevance score for the value of $P(d|Q)$. The expanded query is created by looking at the top 10 documents for each original query, and selecting the top 10 terms for expansion.

2.4 Affinity Propagation

We applied the affinity propagation algorithm [Frey and Dueck, 2007] to further cluster tweets. The idea is that tweets that are highly similar to relevant tweets should have higher chance of being relevant. We chose affinity propagation instead of the other clustering methods for the following reasons.

- Affinity propagation does not require to specify the number of cluster beforehand. This gives us more flexibility since it is hard to fix a single number of clusters for all different queries.
- While many clustering methods only consider the similarities between different data points (i.e. tweets), affinity propagation offers us a natural way to incorporate the relevance score of each tweet into the clustering algorithm. The relevance score of each tweet can effect its chance of being the representative of a cluster. Intuitively, highly relevant tweets should have a better chance of representing the other tweets.
- Affinity propagation is easy to implement and proved to be very fast in practice.

The affinity propagation algorithm can be described as follows. The final goal is to group all the tweets into different clusters, each cluster is represented by one tweet called the exemplar. The input of the algorithm is a matrix of similarities between all the tweets, where the similarity $s(i, j)$ indicates how well the tweet j is to become the exemplar of the tweet i. In particular, the value $s(i, i)$ indicates how well the tweet i is to be an exemplar. In our context, it is natural to set $s(i, i)$ to be proportional to the initial relevance score of the tweet i with respect to the working query.

This algorithm also defines the two terms for message passing: responsibility and availability. The responsibility $r(i, j)$, sent from i to j, denotes the accumulated evidence for how well the item j could be the exemplar of item i. The availability $a(i, j)$, sent from j to i, denotes the accumulated evidence for how appropriate for i to choose j as its exemplar. For each iteration, the algorithm updates all the responsibilities and availabilities of all items, until no change is made for a prespecified number of iterations. The update rules are as follows.

1. All availabilities are initialized to 0.
2. The responsibilities are updated according to the rule
$$r(i, j) = s(i, j) - \max_{k\ s.t.\ k \neq j} \{a(i, k) + s(i, k)\}$$
3. The availabilities are updated according to the rule
$$a(i, j) = min\{0, r(j, j) + \sum_{i'\ s.t.\ i' \notin \{i,j\}} \max\{0, r(i', j)\}\}$$
4. The self-availability is updated differently
$$a(i, i) = \sum_{j\ s.t.\ j \neq i} \max\{0, r(j, i)\}$$

5. Repeat steps (2)-(4) until local decisions (to be specified later) stay constant for a certain number of iterations.

At any point of the process, the value of j that maximizes $a(i,j) + r(i,j)$ identifies the exemplar of the tweet i, which could be itself. It is also considered as the local decision of i, which serves for the stopping criteria mentioned above. In order to avoid numerical oscillations, we set a damping factor λ to be 0.5. That says that the values of the above terms should be updated as a combination of 50% of their previous values and 50% of their prescribed updated values.

After the affinity propagation process stops, we update the score of each tweet by adding to its initial score the relevance score of its exemplar, weighted by some constant α. In our experiments, we set $\alpha = 0.8$.

That remains us to calculate the similarity score of two tweets. Formally, that similarity score consists of two parts: the similarity between the contents of the texts, and the similarity between the two users posting the tweets. For the first part, we built a language model for the content of each tweet, and then calculate the cosine similarity between those two models. The similarity between two users is quite the same, with the only difference is that the language model is now built based on all the tweets posted so far by the user. Finally we combine the two similarities as follows.

$$sim(i,j) = sim_{content}(i,j) + \beta sim_{user}(i,j)$$

In our experiments, β is set to be 1.5.

2.5 Other heuristics

We applied a couple of other heuristics to refine the results.

- In order to remove non-English tweets, we count the portion of English non-stopwords appearing in the tweets. Based on the fact that it is not natural to build an English sentence without using any stop-words, we remove all the tweets that have high ratio of non-stopwords in the sentence.
- Since the final judgements are based on the chronological order, not on the relevance scores, we apply a threshold on the final scores before arranging them in the chronological order.

3 Experimental Results

Table 1 shows the average precision results of our methods. All methods applied the query reformulation technique described above for the initial ranked list. The first column shows the results of applying only query reformulation with the heuristics. The remaining columns show the results of applying other algorithms on top of query reformulation, to further refine the results. The last two columns are our runs submitted to the competition. The first two columns are reported for comparison.

It is observed that query expansion produced the best precision at top 30. It can explained by the fact that tweets are generally short and share many common words with respect to a particular topic. Affinity propagation does not produce the best result by itself, but it is comparable with query expansion when two techniques are combined.

Table 1: Experimental Results of Four Methods: Query Reformulation Only, Query Expansion, Affinity Propagation and the Combination of All Methods

	Query Reformulation Only	Query Expansion	Affinity Propagation	All
P5	0.486	0.461	0.433	0.461
P10	0.455	0.457	0.427	0.455
P15	0.440	0.446	0.391	0.449
P20	0.413	0.425	0.361	0.421
P30	0.387	0.403	0.320	0.399
P100	0.116	0.121	0.096	0.120
P200	0.058	0.060	0.048	0.060
P500	0.023	0.024	0.019	0.024
P1000	0.012	0.012	0.010	0.012

4 Conclusion

This report presents our work for the Microblog retrieval task. Our approach involves applying the relevance feedback model and the affinity propagation clustering algorithm to rerank the initial list of tweets. The results show that query expansion is a very effective technique for microblog retrieval. We also applied the affinity propagation algorithm to cluster the initial ranked list of tweets, for the purpose of promoting similar tweets. However, the results of this approach are less successful. As for future work, we plan to apply some learning-to-rank algorithms for this task, as well as approaching the problem with other formal clustering algorithms.

5 Acknowledgement

This work is partially supported NSF grants IIS-0746830, CNS-1012208 and IIS-1017837. This work is also partially supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370. We also thank the kind supports from Google and Yahoo!

References

[Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science (New York, N.Y.)*, 315(5814):972–976.

[Lavrenko and Croft, 2001] Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01*, pages 120–127.

[Strohman et al., 2004] Strohman, T., Metzler, D., Turtle, H., and B, C. W. (2004). Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*.