Verbose Query Reduction by Learning to Rank for Social Book Search Track

Messaoud CHAA[1,2], Omar NOUALI[1], Patrice BELLOT[3]

[1] Research Center on Scientific and Technical Information 05 rue des 03 frères Aissou, Ben Aknoun, Alger, 16030
[2] Université Abderrahmane Mira Béjaia Rue Targa Ouzemour, Béjaïa 6000
{Mchaa, onouali}@cerist.dz
[3] Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397, Marseille, France
patrice.bellot@lsis.org

Abstract. In this paper, we describe our participation in the INEX 2016 Social Book Search Suggestion Track (SBS). We have exploited machine learning techniques to rank query terms and assign an appropriate weight to each one before applying a probabilistic information retrieval model (BM15). Thereafter, only the top-k terms are used in the matching model. Several features are used to describe each term, such as statistical features, syntactic features and others features like whether the term is present in similar books and in the profile of the topic starter. The model was learned using the 2014 and 2015 topics and tested with the 2016 topics. Our experiments show that our approach improves the search results.

Keywords: Learning to rank, verbose query reduction, Social Book Search, query term weighting, BM15.

1 Introduction

The goal of the Social Book Search Track is to adapt the traditional models of information retrieval (IR) and develop new models to support users in searching for books in LibraryThing (LT). The collection of this task consists of 2.8 million records containing both professional metadata from Amazon, extended with user-generated content, reviews form Amazon and tags from LT [4]. To evaluate systems submitted by participants at the SBS task, a set of topics have been made available which combine a natural language description of the user needs, as well as similar books of his/her topic.

The verbose natural language description of the topic in SBS made the understanding of the users' information need a very difficult task and can cause a topic drift as well as search engine performs poorly.

In this paper, we focus on query reduction and query term weighting [1, 6, 7] to improve the performance of search engine by rewriting the original verbose query into a short query that the search engines perform better with. Explicitly, instead of searching with the original verbose query, it is closest to keyword queries which contain only the important terms.

2 Proposed approach

In this section we present our approach to tackle the described problems of verbose queries. We begin by presenting the idea of our approach. Then we discuss adapting Learning to rank algorithm to weight and rank the terms of the topics. Finally, we explain the employed methodology.

2.1 Query Reduction and term weighting

The key idea of our approach is to reduce the long verbose queries by assigning a weight to query terms. This weight should reflect their importance in the query. We then filtering the less important terms and select the top terms with highest weight to replace the original query. In order to do so we must find a function *f* to weight the original terms that satisfies the following assumption:

Given an arbitrary query *Q ={q1,q2,…;qn}*, let *P(qi,qj)* denotes all possible pairs of the terms of the query. For each existing pair, if *qi* is more important than *qj* then *f(qi)* must be superior to *f(qj)*.

2.2 Learning to rank terms

In our approach, we consider the problem of weighting terms and reducing the verbose queries as a Learning to rank problem [8]. Instead of ranking documents for each topic, as we usually do, we rank the terms of the topic. This can be formally described as follows:

Given *n* training queries *qi (i = 1. . . n)*, their associated terms represented by features vectors, and the corresponding labels (degree of importance of the terms). Then a learning to rank algorithm is used to learn the ranking function. The function learned is applied to rank terms for the test topics.

2.3 Methodology

In order to learn the ranking function, we have to prepare a training data first. There are several important things to be considered:

Training phase.

- Select queries and their associated terms to be used in the training phase;
- Assign to each term a ground truth label (degree of importance)
- Extract a list of features which represent each term: such features have to be as decisive as possible for term weighting;
- Choose and apply a learning to rank algorithm.

Testing phase.

- Apply the ranking function learned in the training phase, to rank terms associated to each unseen query;
- Select the top terms of each query from the ranked list to form the new query;
- Apply an information retrieval model to machining books with this new query.

3 Experimental setup

We used the topics of 2014 and 2015 for training. As to the terms associated to each query, we selected all the terms present in the three topic fields *title, group, and narrative*, as well as the terms of similar books. The natural language processing toolkit *MontyLingua*[1] is used to analyze the text of queries and keep only the nouns and adjectives while eliminating prepositions, verbs and articles.

Regarding the ground truth label for learning and since we have for each topic the relevant books, from *Qrels2014* file, but of course not the most relevant terms, we have decided to rank, for each topic, the terms of the relevant books by using the *tf-idf* function. The label of each previously selected term will be assigned the inverse rank if the term is present in the ranked list, otherwise 0.

For the features, several different categories have been used, including Statistical, Linguistic, Field, Profile, and similar book features. Table 1 describes the features of the five categories we used.

After preparing the training data set as described previously, the learning to rank algorithm Coordinate Ascent from RankLib[2] have been used to learn the function of weighting and ranking terms, this efficient linear algorithm have been chosen due to the unbalanced data we have and in order to avoid the overfitting in the training phase.

Finally, the ranking function learned by the learning algorithm in the training phase have been used to weight and rank the terms of the 2016 topics. The top-10 ranked terms of each topic have been selected to calculate the score of books for each query. The BM15 model [5] was used to matching queries and books as well as the indexation is the same used in our participation to INEX SBS 2015. Please consult [3] for more details on this matching model and indexation process.

[1] http://alumni.media.mit.edu/~hugo/montylingua/

[2] https://sourceforge.net/p/lemur/wiki/RankLib/

Table 1. List of features categorized in five categories.

Features categories	Feature	Feature description
Statistical Features	In_Query	"1" if the term appears in the query and "0" otherwise
	Tf_Iqf (t,q,Q)	Product of Term Frequency and Inverse Query Frequency
	Tf(t,q)	Term Frequency of *t* in the topic
	Iqf(t,Q)	Inverse Query Frequency of the term among all topics
Linguistic Features	Is_Proper_Noun	"1" if the term is a proper noun and "0" otherwise
	Is_Noun	"1" if the term is a noun and to "0" otherwise
	In_Noun_Phrase	"1" if the term appears in the list of noun-phrases extracted from the query and "0" otherwise
	Nb_Noun_Phrases	The number of noun phrases in which the term appears
Field Features	In_Title_Topic	"1" if the term appears in the title of the topic and "0" otherwise
	In_Narrative_Topic	"1" if the term appears in the narrative of the topic and "0" otherwise
	In_Group_Topic	"1" if the term appears in the group field of the topic and "0" otherwise
Profile Features	In_profile	"1" if the term appears in the list of tags extracted from the profile of the user and "0" otherwise
	nTF(t,u)	The ratio of the use of term *t* to tag resources to amount of resources tagged by the user *u*
Example Feautres	In_Example_Book	"1" if the term appears in the example books and "0" otherwise
	Tf_Idf(t,d,D) (in example book)	Product of Term Frequency and Inverse Document Frequency

In order to improve the performance of our system, several combinations of features are experimented to determine the optimal set. Table 2 summarizes the different combinations.

Table 2. List of different combinations

Features Combinations	Description
Stat_features	Statistical features only
Stat_ling_features	Statistical and linguistic features
Stat_ling_field_features	Statistical, linguistic and Field features
Stat_lin_field_profile_feat	Statistical, linguistic, Field and profile features
Stat_ling_profil_expl_feat	Statistical, linguistic, profile and example features
All_features	All categories of features

4 Results

According to the number of combinations described in Table 1, six models have been learned using 2014 and 2015[3] topics and tested using 2016 topics. Table 3 and Table 4 report the evaluation results of the combinations on the training phase, based on *Qrels2014_v1* and *Qrels2014_v2* respectively.

Table 3. Evaluation results of the 2014 topics using *Qrels2014_v1*

Features Combinations	NDCG@10	MRR	MAP	R@1000
Stat_feautres	0.1078	0.2124	0.0805	0.5169
Stat_ling_features	0.1240	0.2546	0.0897	0.5366
Stat_ling_field_features	0.1103	0.2424	0.0801	**0.5401**
Stat_lin_field_profile_feat	0.1156	0.2368	0.0843	0.5249
Stat_ling_profil_expl_feat	**0.1301**	**0.2673**	**0.0945**	0.5063
All_features	0.1277	0.2626	0.0924	0.5133

Table 4. Evaluation results of the 2014 topics using *Qrels2014_v2*

Features Combinations	NDCG@10	MRR	MAP	R@1000
Stat_feautres	0.0961	0.1823	0.0719	0.4919
Stat_ling_features	0.1088	0.2100	**0.0801**	0.5042
Stat_ling_field_features	0.0973	0.1906	0.0712	**0.5105**
Stat_ling_field_profile_feat	0.0975	0.1898	0.0724	0.4940
Stat_ling_profil_expl_feat	**0.1098**	**0.2101**	0.0787	0.4791
All_features	0.1088	0.2090	0.0776	0.4827

For our participation to INEX SBS 2016 track, we submitted six runs. Table 5 shows the official evaluation results of our submissions. From the table, we note that combining linguistic features with statistical features improves the results more than using the statistical features only. In term of NDCG@10 measure, the result increases from 0.1082 to 0.1290. However, when we add the field features and profile features we obtain significantly lower NDCG@10 (0.1084 and 0.1077). We can also clearly mention that combining all features gives the best results in term of NDCG@10, MRR and MAP compared to all other combinations of features. It has 91.09% improvement on NDCG@10, 88.36% on MRR and 63.03% on MAP compared with the baseline. Finally, we can say that our approach has advantage because all the combinations of features perform better than the baseline in term of NDCG@10. For all combinations NDCG@10 is superior than 0.1077 while NDCG@10 of the baseline is 0.0820.

[3] 2015 topics are used to extract example books mentioned by a LT user for the 208 topics.

Table 5. Evaluation results of the 2016 topics

Run name	Features Combinations	NDCG@10	MRR	MAP	R@1000
stat_features	Stat_feautres	0.1082	0.2279	0.0749	0.4326
stat_ling_features	Stat_ling_features	0.1290	0.2970	0.0816	**0.4560**
Not submited	Stat_ling_field_features	0.1084	0.2408	0.0714	0.4557
topic_profil_features	Stat_lin_field_profile_feat	0.1077	0.2635	0.0627	0.4368
all_no_field_feature	Stat_ling_profil_expl_feat	0.1438	0.3275	0.0754	0.3993
all_with_filter	All features with filtering catalogued books	0.1418	0.3335	0.0780	0.3961
All_features	All_features	**0.1567**	**0.3513**	**0.0838**	0.4330
Not submited	Baseline	0.0820	0.1865	0.0514	0.4046
Improvement		91.09%	88.36%	63.03%	07.01%

5 Conclusion

We proposed a simple and effective framework to reduce queries in SBS. Several categories of features have been proposed and used, namely, statistical, Linguistic, Fields, Profile and example features. A learning to rank algorithm has been used to weight and rank terms of the query and then select only the top important to matching books. Our experiments show the effectiveness of the approach.

For perspectives, We would like to use other features in order to better understand the users' information needs and improve the performance of the system like whether the term is a name of author or is part of the title of similar book, etc. we can also use terms with n-grams instead of unigrams.

6 Reference

1. Giridhar Kumaran and Vitor R Carvalho. Reducing Long Queries using Query Quality Predictors. In Proc. of the 32nd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pages 564–571, New York, NY, USA, 2009. ACM.
2. Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. Information Retrieval, 10(3): 257-274, 2007.
3. Messaoud CHAA and Omar Nouali. CERIST at INEX 2015: Social Book Search Track. In Working Notes for CLEF 2015 Conference, Toulouse, France, September 08-11, 2015.

4. Bellot, P., Bogers, T., Geva, S., Hall, M., Huurdeman, H., Kamps, J., Kazai, G., Koolen, M., Moriceau, V., Mothe, J., Preminger, M., SanJuan, E., Schenkel, R., Skov, M., Tannier, X. and Walsh, D. (2014). Overview of INEX 2014. In Information Access Evaluation. Multilinguality, Multimodality, and Interaction (pp. 212-228). Springer International Publishing.
5. Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M. (1995). Okapi at TREC-3. NIST SPECIAL PUBLICATION SP, 109-109.
6. Michael Bendersky and W Bruce Croft. 2008. Discovering key concepts in verbose queries. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages 491–498. ACM.
7. Jae Hyun Park , W. Bruce Croft, Query term ranking based on dependency parsing of verbose queries, Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2010, Geneva, Switzerland.
8. Tie-Yan Liu (2009), "Learning to Rank for Information Retrieval",Foundations and Trends in Information Retrieval, Foundations and Trends in Information Retrieval 3 (3): 225–331