The University of Stavanger at the TREC 2016 Tasks Track

Darío Garigliotti
University of Stavanger
dario.garigliotti@uis.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

Abstract: This paper describes our participation in the Task understanding task of the Tasks track at TREC 2016. We introduce a general probabilistic framework in which we combine query suggestions from web search engines with keyphrases generated from top ranked documents.

1 Introduction

The aim of the TREC Tasks track (Yilmaz et al., 2015) is to devise evaluation methodology for evaluating task-based retrieval systems. In our participation we focus on Task understanding, one of the problems addressed by the track. Specifically, it asks for a ranked list of keyphrases "that represent the set of all tasks a user who submitted the query may be looking for" (Yilmaz et al., 2015). The goal is to provide a complete coverage of subtasks for an initial query, while avoiding redundancy. The suggested keyphrases are judged with respect to each subtask on a three point scale (non-relevant, relevant, and highly relevant). Subtasks are only used in the evaluation, and not available when generating the keyphrases. In addition to the initial query string, the entities mentioned in there are also made available (identified by their Freebase IDs). The quality of the ranked list of keyphrases is evaluated using diversity-aware metrics.

2 Approach

Our approach to task understanding makes use of two keyphrase sources: (i) keywords extracted from relevant documents and (ii) web search engine suggestions. Both sources are combined linearly in a probabilistic scoring model for estimating the probability that a keyphrase q was generated by the initial query q_0: $P(q|q_0)$. The overview of our approach is depicted in Fig. 1. Formally:

$$P(q|q_0) = \sigma P_s(q|q_0) + (1-\sigma)P_k(q|q_0) \qquad (1)$$

where the keyphrase generation probabilities from web search engine suggestion (subscript s) and keyword extraction (subscript k) are combined using the mixture weight σ.

The web search engine suggestions are the top-10 query suggestions from RESTful suggestion APIs of the Google and Bing search engines, given the original query.

Estimating $P_k(q|q_0)$ entails three main steps. First, we obtain the top-10 results from web search engines using the original query and extract a set of keywords from them. Second, we construct possible keyphrase candidates from the original query and the extracted keywords. In the final, third step, we score each candidate keyphrase by combining our confidence scores from the previous steps in a generative probabilistic framework.

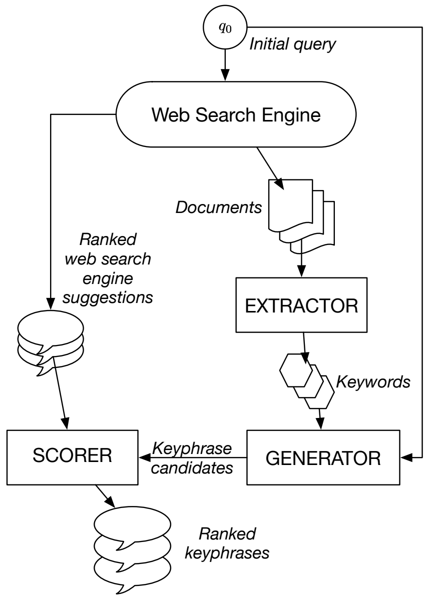


Figure 1: Our task understanding approach.

2.1 Extracting keywords

We issue the initial query (q_0) against two web search engines (Google and Bing) and collect from each the top-K documents ($K = 10$). Each document is represented with two fields: $d_{snippet}$ holds the snippet that was displayed on the SERP and $d_{content}$ stores the full document content (stripped from HTML elements). We extract *keywords* from each document field using the RAKE keyword extraction system (Rose et al., 2010). We note that these keywords are actually phrases, i.e., may consist of more than a single term. For each keyword k extracted from document field d_f, the associated confidence score is denoted by $s(k, d_f)$. We filter the extracted keywords from noise by retaining only those

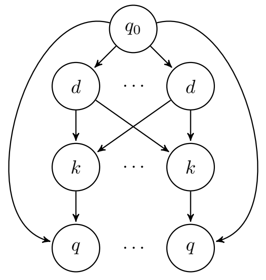


Figure 2: Graphical representation of our generative probabilistic model for generating keyphrases.

that: (i) have an extraction confidence above a given threshold; (ii) are at most 5 terms long; (iii) each of the terms has a length between 4 and 15 characters and is either a meaningful number (i.e., max. 4 digits) or a term (excluding contains noisy substrings and reserved keywords from mark-up languages).

2.2 Generating candidate keyphrases

Given the initial query and the extracted keywords, we form a weighted set of candidate keyphrases. We consider three different ways of combining a keyword k and the initial query q_0:

(1) adding k as a suffix to q_0;

(2) adding k as a suffix to an entity mentioned in q_0;

(3) returning k as-is, without considering q_0.

If the initial query contains multiple entities, then (2) is performed for each of the mentioned entities. Each generated keyphrase q is assigned a weight based on which rule generated it:

$$s(q,q_0,k) = \begin{cases} \alpha, & \text{if } q = q_0 \oplus k, \\ \beta, & \text{if } q = e \oplus k, \\ \gamma, & \text{otherwise.} \end{cases} \tag{2}$$

where $\alpha+\beta+\gamma = 1$. When a keyphrase could be generated by multiple rules then we take the one with the highest weight.

We use a custom operator for concatenation, $\oplus$, which first removes the longest common subphrase from the right-side concatenation term. For example: "*aa bb*" $\oplus$ "*bb cc*" $=$ "*aa bb cc*" and *"choose bathroom decor"* $\oplus$ *"bathroom decor style"* = *"choose bathroom decor style."* Such a removal strategy is motivated by the fact that in English most of the possible (non-adjectival) refinements are syntactically performed as an addition to the right.

2.3 Scoring keyphrases

We now introduce the generative probabilistic model $P_k(q|q_0)$, which operates as follows. First, we consider the documents that are relevant to the initial query. Then, we take the keywords that are generated by these documents. Finally, we use both the initial query and the keywords as generators of keyphrases. The graphical representation of the model is depicted in Figure 2. Formally:

$$P(q|q_0) = \sum_d P(d|q_0) \sum_k P(q|q_0,k) P(k|d). \tag{3}$$

This model has three components:

- $P(d|q_0)$ expresses the relevance of document d to the initial query q_0. Since we do not have relevance scores for documents, we make the simplifying assumption that all documents are equally important, i.e., set this probability uniformly across documents. This is reasonable as we only consider documents from the first search result page.

- $P(k|d)$ is the keyword generation probability, which is estimated according to:

$$P(k|d) \propto \sum_{f\in F} \lambda_f s(k,d_f), \tag{4}$$

where $s(k,d_f)$ is the confidence score from keyword extraction, F is the set of document fields (*snippet*, *content*), and λ_f are the corresponding field weights ($\sum_{f\in F}\lambda_f = 1$). To simplify notation, we write λ to denote $\lambda_{snippet}$, which implies that $\lambda_{content} = 1-\lambda$.

- $P(q|q_0,k)$ is the keyphrase generation probability, which is set proportional to the weight of the rule that was used when creating that candidate keyphrase (cf. Eq. (2)):

$$P(q|q_0,k) \propto s(q,q_0,k). \tag{5}$$

We omitted the normalizer terms in Eqs. (4) and (5) for readability.

3 Official Runs and Results

We submitted the following three runs:

`UiS4` For a given initial query, we use uniformly the web search engine suggestions with the highest priority in the ranking. Formally, $P_s(q|q_0)$ is uniformly distributed in Eq. (1), and σ is chosen such that $\sigma P_s(q|q_0)$ is larger than the maximum of the scores of any other possible candidate. Next in the ranking follow the keyphrases generated as described in Sect. 2.2, using only the extracted keywords as-is (i.e., by making $\gamma = 1$ in Eq. (2)); λ is set such that keywords originating from the *snippet* field have priority over the ones from the *content* field.

`UiS8` This approach is essentially the same as `UiS4`, but here the web search engine suggestions are ranked with a score $P_s(q|q_0)$ proportional to their rank position, as retrieved from the suggestion APIs. As before, they are still placed in positions higher than any other keyphrase. The other keyphrases in the ranking are obtained and scored as before.

Table 1: Configuration settings used in our runs.

Run	Web search engine suggestions	Generated keyphrases
`UiS4`	Set	$\gamma = 1$
`UiS8`	List	$\gamma = 1$
`UiS9`	Not used	$\alpha, \beta > \gamma$

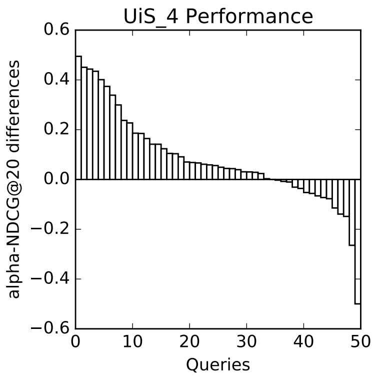
Table 2: Evaluation results for our runs, as well as the mean and maximum scores across all participants. All numbers are averaged over the set of test queries.

Run	α-NDCG@20	ERR-IA@20
`UiS4`	0.6596	0.5333
`UiS8`	0.6985	0.5670
`UiS9`	0.6056	0.4738
Participants Mean	0.4941	0.4149
Participants Max	0.7664	0.6821

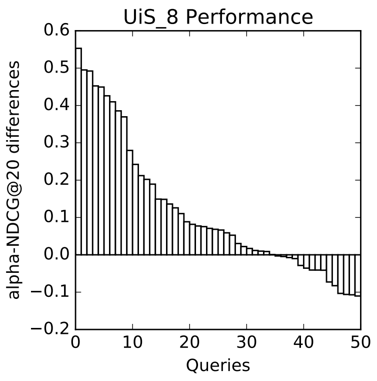
`UiS9` In this approach we obtain keyphrases by using all the generation rules, cf. Sect. 2.2. The scoring schema is such that it forces to rank first keyphrases that are not from the last rule (i.e., by setting α and β in Eq. (2) such that $\alpha > \gamma$ and $\beta > \gamma$) and from any field; then follow the extracted keywords as-is from the *snippet* field; and finally the extracted keywords from the *content* field. We do not make use of web search engine suggestions at all, i.e. $\sigma = 0$ in Eq. (1).

Table 1 presents a more concise description of our three systems, by indicating how each of the two main keyphrase sources contributes to the final ranking.

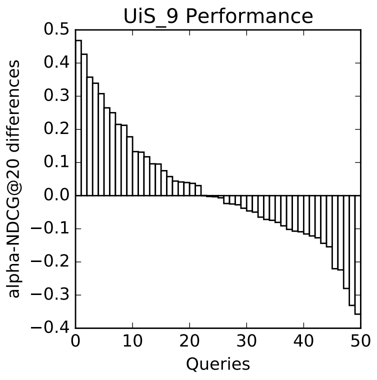
Apart from the evaluation scores of our runs, the only results currently available from the track are the minimum, mean, and maximum performances of each query across all the participating systems. Table 2 presents these results.



(a) UiS4



(b) UiS8



(c) UiS9

Figure 3: Evaluation of the performances of our approaches on the 2016 dataset.

4 Analysis

In this section we summarize the main insights from our results. Since the ground truth dataset is not released yet, we perform a simple evaluation by obtaining, for each of our submissions, the differences per query between our α-NDCG@20 and the mean performance according to the same metric. Fig. 3 shows these differences for each of our three approaches.

After comparing Fig. 3a and Fig. 3b, it is clear that the web search engine suggestions, while keeping their relative ranking positions as they are provided, contribute substantially. This might also give some clues about the nature of the ground truth dataset, as most of its results come from the same kind of search engine suggestions. The lack of such suggestions in `UiS9` is somehow compensated with our keyphrase generation method if comparing both averaged performances in Table 2. On the other hand, relying only on generated keyphrases (`UiS9`) is not very effective on its own; roughly half of the queries have a negative difference. Our other two systems increasingly from `UiS4` to `UiS8` perform with a good amount of large positive differences.

5 Conclusions

We have described our participation in the TREC 2016 Tasks track. We have introduced a general probabilistic framework in which we combine query suggestions from web search engines with keyphrases generated from top ranked documents. Our initial results have been promising.

In the future we plan to extend our approach to incorporate diversity, and also consider additional sources for extracting keywords and methods for generating keyphrases.

6 References

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. In *Text Mining: Applications and Theory*.

Yilmaz, E., Verma, M., Mehrotra, R., Kanoulas, E., Carterette, B., and Craswell, N. (2015). Overview of the TREC 2015 tasks track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*.