Report IR

Group 31

Nathan Buskulic
4947916
Tu Delft

Marjolein Knijff
4153634
Tu Delft

ABSTRACT

This is a short abstract

1 INTRODUCTION

This paper intent to reproduce the results of the paper *Supervised Query Modeling Using Wikipedia* [**?**]. We consider that finding techniques to improve results of ambiguous queries could lead to a big improvement of search engines and thus help the community in general and this paper propose an elegant machine learning solution to do so. One example of an ambiguous query that the original paper takes is the query *the secret garden*. The query could lead to either the novel, the musical or the movie. It could also lead to more general results given the very general terms of the query.
Several approaches exists to try to solve this problem (or at least help). One possibility is to use properties of the web graph for example. However we are interested in modifying the query in such a way that it will improve our results. Modifying the query in any way can harm the results (FIND PAPER), we thus need a way of making sure we are improving the query and not harming it. The solution here is to use a semantically inform way to improve the query. This is done by training a machine learning algorithm (an SVM with polynomial kernel) to link queries to relevant wikipedia articles that will then help to improves the query model. Since we aimed to reproduce a paper but that the time was limit, some differences has to be noted and will be explained in more detail in further sections. These differences are visible for example in the choice of features for the SVM, the ranking method or in the benchmark that we did. It is however interesting to see how these discrepancies affect the results and to what extends the difference between our results and the original results are due to these discrepancies.

2 BACKGROUND

what important works does this project build on

3 APPROACH : HOW TO MODEL THE QUERY

For this paper we used a language modeling framework which has been proved to be competitive with state of the art Information Retrieval techniques. We are interested in knowing the probability that the query is generated from the document :

$$P(D|Q) \propto \log(P(D)) + \sum_{t \in Q} P(t|\theta_Q) \log(P(t|\theta_D)) \tag{1}$$

In our model we use a uniform prior probability for all document. In order to generate the language model θ_D of a document, we apply bayesian smoothing with dirichlet prior. The model of the query is generated by using linear interpolation :

$$P(t|\theta_Q) = \lambda_Q P(t|\tilde{\theta}_Q) + (1 - \lambda_Q) P(t|\hat{\theta}_Q) \tag{2}$$

In that formula, $P(t|\tilde{\theta}_Q)$ describes the probability of a term given the estimated query language model of the initial query. $P(t|\hat{\theta}_Q)$ is the extended part of the query which is calculated by using :

$$P(t|\hat{\theta}_Q) = \frac{1}{|R|} \sum_{D \in R} P(t|D) P(Q|D) \tag{3}$$

where R is a set of relevant documents. If $\lambda_Q = 1$, we get a simple likelihood that do not uses any relevant documents to expand the query, this will serve as our baseline to compare our approach. The relevant document that we need are obtained by linking queries to wikipedia article using machine learning, the details of this process will be described in the next section. Counter to the original paper, we will not use any pseudo relevance feedback because we did not have the time to implement them correctly. Thus our results will be focus on the comparison between the baseline and the new model that includes wikipedia articles.

3.1 Get relevant wikipedia articles from queries

In order to get relevant wikipedia articles, we needed to build a system that was able to predict for any given query, which articles would be relevant. This kind of task requires to use machine learning classifiers where in our case the input is features from the query and the current article and the output will be a binary classification measure, 1 if the article is relevant to the query, 0 otherwise. Since the number of training samples we can get is limited due to the time it takes to decide which article is relevant for each query and the fact that we will have a high number of features, we want a classifier able to have good results in that kind of environment. The choice was made to use an SVM that is known (FIND PAPER) to be efficient in high dimensionnal space even with a limited number of training samples. This is due to the fact that only a few training samples from the dataset (the support vector) are needed to build the classification hyperplane. We use a simple polynomial kernel, further work could be done to investigate the effect of different kernel on this task, indeed having only a polynomial kernel could maybe represent poorly the true distribution of the data.

The dataset we used in our experiments is two ad-hoc TREC test collections : *TREC Terabyte 2004–2006 (.GOV2)*. In order to train the SVM we created a training dataset of queries to wikipedia articles mapping. This was done by asking member of the team to manually link each query of the *TREC Terabyte 2004* to the relevant wikipedia articles. For new queries, we first do a retrieval run on wikipedia and take the first 10 results. Each of this new result is presented to the trained SVM which will predict the relevance of the article.

Features used with the SVM

Features	
Len(Q)	The number of words of the query
TF(Q,D)	Relative phrase frequency of Q in D, normalized by length of D
Pos_n(Q,D)	Position of nth occurrence of Q in D, normalized by length of D
SPR(Q,D)	Spread (distance between the last and first occurrences of Q in c)
DCQ(Q,D)	Does D contain Q?

In the original paper a lot of different features where used, however because of technical difficulty and time limitations, we decided to only implement a subset of these features that we think represents already well a good part of the information of the true distribution. These features are shown in Table 1. Since it is highly improbable that the full qury will be in a document, we get all of these features for each term of the query as well as for the full query and we combine tose into one high dimensionnal space.

EXPLAIN HERE ALL THE FEATURES

4 EXPERIMENTS

describe your experiments, the results and discuss them

5 CONCLUSIONS

describe what you learnt/found and what avenues for future work you see