On-line learning for Web query generation: finding documents matching a minority concept on the Web

Rayid Ghani[1], Rosie Jones[1], and Dunja Mladenic[1 2]

[1]Carnegie Mellon University (USA),[2]J. Stefan Institute (Slovenia)
{Rayid.Ghani, Rosie.Jones, Dunja.Mladenic}@cs.cmu.edu

Abstract. This paper describes an approach for learning to generate web-search queries for collecting documents matching a minority concept. As a case study we use the concept of text documents belonging to Slovenian, a minority natural language on the Web. Individual documents are automatically labeled as relevant or non-relevant using a language filter and the feedback is used to learn what query-lengths and inclusion/exclusion term-selection methods are helpful for finding previously unseen documents in the target language. Our system, CorpusBuilder, learns to select "good" query terms using a variety of term scoring methods. We present empirical results with learning methods that vary the time horizon used when learning from the results of past queries. Our approaches generalize well across several languages regardless of the initial conditions.

1 Introduction

In this paper we describe an approach for learning to automatically generate web queries to retrieve documents in a minority language. We explore different term-selection methods and lengths for generating queries and use on-line learning to modify the queries based on feedback by the language filter. We show that starting from a single document or a set of keywords in the target concept, our methods can learn to generate queries that can acquire a reasonable number of documents in Slovenian from the Web and that our approach also generalizes to other languages that are also minority languages on the Web.

Glover et al. [7] use machine learning to automatically augment user queries for specific documents with terms designed to find document genres, such as home-pages and calls for papers. Rennie et al. [9] use reinforcement learning to help a crawler discover the kinds of hyper-links to follow to find research papers. WebSail [4] uses reinforcement learning based on relevance feedback from the user. Our approach differs from WebSail in that we derive our learning signal automatically from a language filter, and does not require any user input. Boley et al. [2] proposed to use the most-frequent words for query generation for their WebACE system. However, they did not evaluate a system employing automatic query-generation. Ghani et al. [5] described an algorithm for building a language-specific corpus from the World-Wide Web. However, their experiments were limited to a small closed corpus of less than 20,000 documents, vastly limiting the generalization power of their results to the Web. They also did not investigate the use of learning.

2 CorpusBuilder Architecture

Our system, CorpusBuilder, iteratively creates new queries, in order to build a collection of documents in a single language. The target language is defined by one or more initial documents provided by the user, and the language filter. At a high level, CorpusBuilder works by taking as initial input from the user two sets of documents, relevant and non-relevant. Given these documents, it uses a term selection method to select words from the relevant and non-relevant documents to be used as inclusion and exclusion terms for the query, respectively. This query is sent to the search engine and the highest ranking document is retrieved, passed through the language filter and added to the set of relevant or non-relevant documents according to the classification by the filter. The process is then iterated, updating the set of documents that the words are selected from at each step.

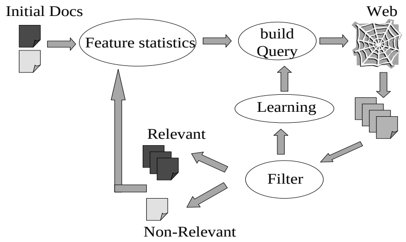

Fig. 1. System Architecture

We generate queries using conjunction and negation of terms. Query term selection methods were as follows. As a baseline we used *uniform* (UN) selecting k terms from the relevant documents, with equal probability of each term being selected. *term-frequency* (TF) selects the k most frequent terms from the relevant documents. *probabilistic term-frequency* (PTF) selects k words from the relevant documents according to their frequency and has been shown to perform better than simple frequency on a similar problem [5]. *rtfidf* (RTFIDF) selects the top k words ranked according to their rtfidf scores. *odds-ratio* (OR) selects the k terms with highest odds-ratio scores and has been shown to outperform other methods on document categorization when dealing with a minority concept [8]. *probabilistic odds-ratio* (PO) selects words with probability proportional to their odds-ratio scores.

We used van Noord's TextCat implementation [10] of Cavnar and Trenkle's character n-gram based algorithm [3] which was shown to be over 90% accurate on a variety of languages and document lengths.

3 Fixed Query Parameters

To understand the behavior of our term-selection methods we conducted exhaustive experiments for each length separately. These experiments used three different initial documents and the variance in the results was small. The evaluation measures we used were (a) percentage of documents retrieved in the target class and (b) percentage of web queries retrieving documents in the target class. Our experiments show that *odds-ratio* (OR) is consistently the best with respect to both evaluation measures. Different methods also varied in the query-length at which they attained their peak performance: *term-frequency* with length 4, *probabilistic term-frequency* and *odds-ratio* length 3 while *probabilistic odds-ratio* with length > 1 gives a very high number of rather strict queries for a very small number of target language documents. The detailed results are reported in [6].

4 Learning query parameters

As described in section 3, different methods excel with different query lengths. As we retrieve more documents, our system may explore different parts of the Web and perform better using different querying mechanisms. This observation motivates a family of algorithms that have access to the same term-selection methods as before and can learn the ideal method and length at different points in time. We describe these learning algorithms in the next section and also report experimental results.

Our queries can be described by four parameters: two for Inclusion and Exclusion Term-Selection Method and two for Inclusion and Exclusion Length. Since we believe that the target concept is shifting and a query method that works well in the beginning in one part of the feature space may not work well later during the process, we incorporate some randomness in our learning methods. Instead of learning the four parameters for a query directly, we focus on learning the success rate for each term-selection method and length (0—10) and then by imposing a multinomial distribution over all methods and lengths (their probabilities being proportional to their success rates), we can probabilistically select the parameter values. We do not use the *uniform* term-selection method in our learning experiments, since it performed poorly during experiments not involving learning.

4.1 Learning Methods

We performed experiments varying the time horizon used in our on-line learning [1]: from all available history, to a time-decaying view of the past, to a learner firmly rooted in the present. Since our target concept at every step is previously unseen documents in the minority class, the set of target positive documents is reduced at every step. Thus more recent queries may be more relevant to the current best query. At the same time, the aggregated knowledge from past queries may prove invaluable for learning about the task as a whole. **Memory-Less Learning (ML)** was designed to permit a successful querying method to continue as long as it was finding positive documents. In this method, we pick the initial method uniformly and then continue with the successful method until it fails. On failure, we pick one of the other methods with uniform probability. **Long-Term Memory Learning (LT)** estimates each method's future probability of success based equally on all past performance. We used two kinds of updating rules: additive update (LTA) and multiplicative (LTM), Winnow-like update using $\beta = 0.5$. **Fading Memory Learning (FM)** bases some of the current performance on the past, but gradually reduces the impact of learning experiences further in the past.

4.2 Results for Learning Methods

Our first set of experiments compared different methods when using the same fixed length for both inclusion and exclusion terms. We found that the best performing length is 3—5, since length 3 or higher was the best in the percentage of the target class documents, while in the number of queries length 1—5 was the best. What happens is that shorter queries are more successful in getting documents but less accurate than the longer queries.

For each document-based experiment, our system had access to one positive document in the target language (Slovenian, Croatian or Tagalog) and four negative documents in the other languages (Czech, Croatian, English, Serbian and Slovenian). Our hypothesis is that different learning methods will differ in their performance, since they use different time horizons in the on-line learning process. When comparing different learning methods, Long-Term Memory and Fading Memory learning perform better than Memory-Less (Figure 2). Long-Term Memory learning dominates both Fading Memory and Memory-Less in terms of number of documents retrieved, as well as queries issued. However, all learning methods underperform the best performing combination of parameters (*odds-ratio* using length 3—5), that we found by manually searching the parameter space exhaustively. In order to test the generalization power of our

(a) (b)

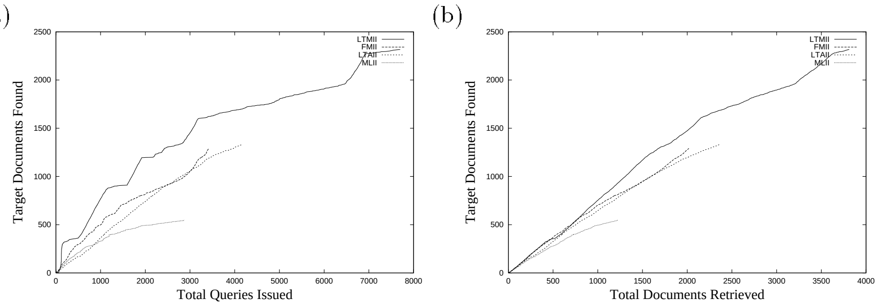

Fig. 2. Comparison of learning methods. (a) In terms of queries Long-Term Memory using Winnow-like update performs the best. (b) In terms of retrieved documents, after 700—1000 documents LTMII is again the best.

system for different target-languages, we performed experiments on two other natural languages, Croatian and Tagalog, which are also representative of minority languages on the Web. The results confirm that after 700-1000 documents and about 1000 queries issued, the methods start to differ on all three languages. The best performance is achieved by the Long-Term Memory methods.

5 Conclusions and Future Work

We found that our basic *odds-ratio* query construction method outperforms our other methods. *odds-ratio* picks inclusion query terms that are highly unique to the target language while excluding terms that are unique to non-relevant languages. Since this is the only method which uses both relevant and non-relevant documents simultaneously to select query terms, we believe that this property is the key to its success. *tf* picks terms that are frequent in the target language but not necessarily unique and hence results in queries that are not as precise as those generated by *odds-ratio*.

We also found that for experiments with learning query parameters, Memory-Less learning (`ML`) performs worse than all the other learning methods over different natural languages. This was expected since `ML` is a naive algorithm which persists with a successful mechanism until it fails and then switches to another one randomly thus ignoring past knowledge of success rates. The best performance was achieved using Long-Term Memory learning (`LT`) with either a multiplicative or an additive update rule. It accumulates all the information from earlier queries by counting the successes and failures of individual mechanisms and updates their scores accordingly. It is interesting to note that Fading Memory learning (`FM`), which relies more on recent information, performs worse than `LT`.

We tested the influence of changing the initial conditions for both fixed and learning query parameters and found that the variance in the results was small. We started the system using three different initial documents and, as an alternative, starting with nine lists each of 10 words supplied by three native speakers of Slovenian (common, unique, useful for the task).

An interesting question for future work is how our results transfer to other target concepts such as collecting documents about a topic or documents that match a user profile. Using these techniques to augment existing techniques for developing a domain specific search engine is also an interesting future direction.

References

1. Blum, A. (1996). On-line algorithms in machine learning. *In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl, 1996.*.
2. Boley, D., Gini, M., Gross, R., Han, E.-H. S., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., , & Moor, J. (1999). Document categorization and query generation on the world wide web using webace. *AI Review*, *13*, 365–391.
3. Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Proceedings of SDAIR 1994* (pp. 161–175). Las Vegas, NV.
4. Chen, Z., Meng, X., Zhu, B., & Fowler, R. H. (2000). Websail: From on-line learning to web search. *Proc. of the International Conf. on Web Information Systems Engineering*.
5. Ghani, R., & Jones, R. (2000). Learning a monolingual language model from a multilingual text database. *Proceedings of CIKM 2000*.
6. Ghani, R., Jones, R., & Mladenić, D. (2001). *Building minority language corpora by learning to generate web search queries* (Technical Report CMU-CALD-01-100).
7. Glover, E., Flake, G., Lawrence, S., Birmingham, W. P., Kruger, A., Giles, C. L., & Pennock, D. (2001). Improving category specific web search by learning query modifications. *Symposium on Applications and the Internet*. San Diego, CA.
8. Mladenic, D., & Grobelnik, M. (1999). Feature selection for unbalanced class distribution and naive bayes. *Proceedings of ICML 1999*.
9. Rennie, J., & McCallum, A. K. (1999). Using reinforcement learning to spider the web efficiently. *Proceedings of ICML 1999*.
10. van Noord, G. Textcat. http://odur.let.rug.nl/ vannoord/TextCat/.