Building Minority Language Corpora by Learning to Generate Web Search Queries

Carnegie Mellon University, Center for Automated Learning and Discovery, Technical Report

CMU-CALD-01-100

Rayid Ghani RAYID.GHANI@CS.CMU.EDU

Center for Automated Learning & Discovery, Carnegie Mellon University, Pittsburgh, PA, USA

Rosie Jones ROSIE.JONES@CS.CMU.EDU

Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Dunja Mladenic DUNJA.MLADENIC@{IJS.SI,CS.CMU.EDU}

J. Stefan Institute, Ljubljana, Slovenia and Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

The Web is an obvious source of valuable information but the process of collecting, organizing and utilizing these resources is difficult. We describe CorpusBuilder, an approach for automatically generating Web-search queries for collecting documents matching a minority concept. We use the concept of text documents belonging to 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 learns to select good query terms using a variety of term scoring methods. We find that using *odds-ratio* scores calculated over the documents acquired so far was one of the most consistently accurate query-generation methods. We also parameterize the query length using a Gamma distribution and present empirical results with learning methods that vary the time horizon used when learning from the results of past queries. We find that our systems performs well whether we initialize it with a whole document, or with a handful of words elicited from a user. Experiments applying the same approach to multiple languages are also presented showing that our approach generalizes well across several languages regardless of the initial conditions.

1. Introduction

Electronic text corpora are used for modeling language in many language technology applications, including speech recognition (Jelinek, 1999), optical character recognition, handwriting recognition, machine translation (Brown et al., 1993), and spelling correction (Golding & Roth, 1999). They are also useful for linguistic and sociolinguistic studies, as they are readily searchable and statistics can easily be computed.

Current methods for creating text corpora for specific languages require a lot of manual human effort and are very time-consuming. The Linguistic Data Consortium (LDC) has corpora for twenty languages (Liberman & Cieri, 1998) while Web search engines currently perform language identification on about a dozen of the languages they index, allowing language-specific searches in those languages. Documents in many other languages are also indexed, though no explicit labeling of the language they are written in is available.

The WWW has been gaining popularity as a resource for multilingual content. Resnik (Resnik, 1999) explored the Web as a source for parallel text and automatically constructed a parallel corpus of English and French. In this paper, We describe techniques which only require the user to give a handful of keywords or documents for automatically collecting language specific resources from the Web and present a system which automatically generates Web search queries to construct corpora for minority languages. Our proposed approach requires no human intervention once the system is provided with the initial documents and is a very cheap and fast way to collect corpora for minority languages from the Web.

In general, to quickly find documents in a specific language, we need to be able to construct queries that both find a wide range of documents in the target language, and that filter out a large proportion of more and less closely related languages. Our hypothesis is that by selecting appropriate inclusion and exclusion terms from documents already collected, and using the results of classification by a high-precision language filter, we can construct very high-precision queries automatically. This approach should work well without specialized knowledge of which languages are related.

In this paper we focus on a language-filter as a high-precision classifier, and how query-generation can bring in a much higher proportion of documents in the target language than random crawling, or use of a search engine's "Related Documents" option. 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 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.

2. Related Work

While search-engines are an invaluable means of accessing the Web for users, automated systems for learning from the Web have primarily been installed in crawlers, or spiders. A new generation of algorithms is seeking to augment the set of search capabilities by combining other kinds of topic or target-directed searches.

Glover and colleagues (Glover et al., 2001) 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 and McCallum (Rennie & McCallum, 1999) use reinforcement learning to help a crawler discover the right kinds of hyper-links to follow to find postscript research papers. Diligenti et al. (Diligenti et al., 2000) make use of hyper-link structure to learn naive Bayes models of documents a few back-links away from target documents to aid a crawler. WebSail (Chen et al., 2000) 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 do not require any user input. Boley et al. (Boley et al., 1999) proposed using the most-frequent words for query generation for their WebACE system, generating these from clusters, seeking to maximize term-frequency and document frequency of the terms selected. They used stemmed versions of words as query terms. They showed by example that automatically generated queries with a combination of conjunctive and disjunctive terms can be used to find more related documents. They used queries that used a combination of conjunctive and disjunctive terms. However, they did not evaluate a system employing automatic query-generation.

Ghani and Jones (Ghani & Jones, 2000) 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. They showed that single word queries were sufficient for finding documents in Tagalog, and that selecting the query-words according to their probabilities in the current documents

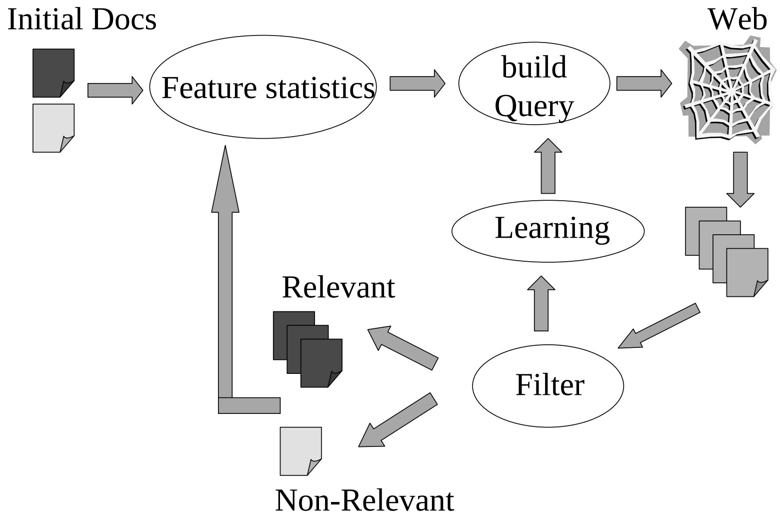


Figure 1. System Architecture

performed the best. It is important to note that their experiments were run on a small corpus[1] of Tagalog documents and other distractor documents collected from the web and stored on disk. We compared their best-performing methods against other query generation methods and lengths, on the tasks of finding both Tagalog and Slovenian documents on the Web and found that applying single-word *term-frequency* and *probabilistic term-frequency* queries to the Web for Slovenian results in relatively low precision and using our *odds-ratio* query generation method described in section 3.3 outperforms the *probabilistic term-frequency* approach with single include and exclude-word queries. Furthermore, using more words in the query (3 for inclusion and 3 for exclusion) performs better than the single word queries previously used.

3. CorpusBuilder System Description

In this section we describe the CorpusBuilder architecture, the query-generation methods and the target concept filter, which is a language-filter for different natural languages. We use the feedback from the language filter as a training signal for learning the ideal query length and query term-selection method.

3.1 General Algorithm

CorpusBuilder iteratively generates queries, in order to build a collection of documents belonging to the target concept. The target concept is defined by one or more initial documents provided by the user, and the target concept filter (a classifier with which we decide if a particular document belongs to the target concept). The initial input is two small sets of documents, relevant and non-relevant. Given these documents, a term selection method is used 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 at each step the set of documents that the words are selected from. The overall architecture is shown in Figure 1. When querying the search engine, if we issue a new query, the first hit is used but the remaining hits are stored to a file for efficiency so that in the future use we avoid consulting the search engine for the same query. If we have searched using the same query before, we take the next unseen hit from our file caching the previous response of the search engine on that query. If all hits have been seen, no hit is returned.

The general algorithm is as follows:

1. Initialization (see Section 3.2 for description)
   - RelDocumentList = a single document belonging to the target concept
   - RelWordFrequencies = word_frequencies(RelDocumentList)

[1] the corpus consisted of 500 Tagalog documents and 15000 documents mostly in English and Brazilian Portuguese

- NonRelDocumentList = one or more documents not belonging to the target concept
- NonRelWordFrequencies = word_frequencies(NonRelDocumentList)

2. Generate Query using terms from relevant and non-relevant documents (Section 3.3)
   - select k inclusion terms using generate_words(RelWordFrequencies)
   - select k exclusion terms using generate_words(NonRelWordFrequencies)
   - search the Web using a query q of the form "$+incTerm_1..+incTerm_k-excTerm_1..-excTerm_k$"

3. Retrieve a Document
   - **if** q is a new query, issue to web search engine, and take first hit URL
   - **if** we have searched using q before, take next unseen hit. If all hits have been seen, no hit is returned
   - **if** no hit, return to step 2 **else** fetch document d corresponding to URL from the web

4. Filter: run filter on d and assign d to RelDocumentList or NonRelDocumentList

5. Update RelWordFrequencies, NonRelWordFrequencies based on relevant and non-relevant documents

6. Return to step 2

We retrieve a single document in step 3 to allow the algorithm maximum opportunities to improve performance using the language filter at every step. Interesting future work would involve investigating how the number of documents retrieved before updating models could be optimized, possibly by examining the number of positive documents returned so far by the current query.

3.2 Initialization

We only describe the initialization for the relevant class, which is used for selecting inclusion terms for the query. The operation for the non-relevant class and exclusion terms is performed identically.

A small number of documents in the target class are supplied as initial documents for the positive class. We generally used a single initial document, and also experimented with using only 10 keywords in place of a whole document. Term frequencies for all initial documents in the relevant language and irrelevant languages are calculated separately. These then supply the information required by the query generation methods: term frequency probabilities from relevant documents only, and where necessary, odds-ratio and RTFIDF scores, which use statistics from both relevant and irrelevant documents. The details of the way the query generation methods use these statistics is described in section 3.3.

3.3 Query Generation Methods

Given a collection of documents classified into relevant and non-relevant, the task of a query generation method can be described as follows: examine current relevant and non-relevant documents to generate a query which is likely to find documents that are *similar* (but not the same) to the relevant documents (i.e. also relevant) and not similar to the non-relevant documents. This is analogous to sampling without replacement. The Web contains different documents in a variety of natural languages. We use query generation and a search engine to sample from this document collection. Ghani and Jones (2000) experimented with sampling with and without replacement and found that both strategies performed similarly for their dataset.

We construct queries using only conjunction and negation of terms (literals). A query is defined to consist of a set of terms required to appear in the documents retrieved (inclusion terms), and a set of terms forbidden from appearing in the documents retrieved (exclusion terms). Consequently, each query can be described by four parameters: the number of inclusion terms, the number of exclusion terms, the inclusion term selection method, and the exclusion term selection method. This contrasts with full Boolean queries, which give greater expressive power by also employing disjunction, and negation with a greater variety of scope. We chose to use only conjunction to simplify the experiments.

For experiments that do not involve learning we set the number of inclusion terms equal to the number of exclusion terms, and used a fixed term selection method throughout the entire experiment. The term selection method selects k inclusion and k exclusion terms using the words that occur in relevant and non-relevant documents, using different methods to select terms. Since our task does not have a fixed goal in terms of a single best query, we need query generation methods that adapt to the current situation where we have already acquired a set of documents from the target concept and do not want to explore the same space again.

The query generation methods we use are as follows: *uniform*, *term-frequency*, *probabilistic term-frequency*, *rtfidf*, *odds-ratio*, and *probabilistic odds-ratio*. Each is described below for inclusion terms with k being the number of terms to be generated. The operation for exclusion terms is analogous, swapping relevant and non-relevant documents where appropriate.

- *uniform* (UN) selects 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 unsmoothed maximum-likelihood probabilities, that is, with probability proportional to their frequency. More frequent words are more likely to be selected.
- *rtfidf* (RTFIDF) selects the the top k words ranked according to their rtfidf scores. The rtf score of a term is the total frequency of that term calculated over all relevant documents as classified by the language filter. The idf score of a term is calculated over the entire collection of documents retrieved, and is given by $log(\frac{total\ number\ of\ documents}{number\ of\ documents\ containing\ the\ term})$. rtfidf for a term is the product of rtf and idf.
- *odds-ratio* (OR) selects the k terms with highest odds-ratio scores. The odds-ratio score for a word w is defined as

$$log_2(\frac{P(w|relevant\ doc) * (1 - P(w|nonrelevant\ doc))}{P(w|nonrelevant\ doc) * (1 - P(w|relevant\ doc))})$$

- *probabilistic odds-ratio* (POR) selecting words with probability proportional to their odds-ratio scores.

The simplest measure used as a baseline is random selection of terms, i.e. a uniform probability distribution is imposed over all words in the vocabulary (UN). Scoring terms according to their frequency (TF) has been used as a simple measure known to give good results for feature scoring in document categorization (Mladenic & Grobelnik, 1999), (Yang & Pedersen, 1997). Using a multinomial distribution over frequency of terms (PTF - *probabilistic-term-frequency*) has been shown to perform better than simple frequency on a similar problem (Ghani & Jones, 2000). Haines and Croft (1993) show that rtfidf is a good scoring mechanism for information retrieval. Mladenic and Grobelnik (1999) have shown that scoring using *odds-ratio* (OR) achieves very good results on document categorization when dealing with a minority concept, which is exactly our problem scenario since Slovenian is a minority language on the Web. Motivated by the superior performance of *probabilistic term-frequency* over *term-frequency*, we derived a variant of *odds-ratio*, which is selecting terms according to a multinomial distribution over odds-ratio scores of terms (POR - *probabilistic-odds-ratio*).

The query generated at each step may be a novel query, or one we have already issued, either because the method is probabilistically selecting terms or because the addition of new documents did not change word distributions in a way which influences the term selection.

3.4 Recovery from Empty Query Results

In the case of a deterministic term-selection method, such as *term-frequency*, *rtfidf* and *odds-ratio*, query terms selected can change only when the underlying document statistics change through the addition of a new document. When a query adds no new documents, we need a method of altering the query in order for it to recover. We took the approach of successively incrementing a counter i, first through inclusion terms, then through the exclusion terms, taking the i through $i+k$th highest scoring terms till a query is found which returns a URL.

3.5 Language Filter

After each query is generated, it is passed to a search engine, and the next matching document is retrieved. We pass each document retrieved by a query through a language filter based on van Noord's TextCat implementation (van Noord, 1997) of Cavnar and Trenkle's character n-gram based algorithm (Cavnar & Trenkle, 1994). Cavnar and Trenkle show accuracy of over 90% on a variety of languages and document lengths. We considered a document to belong to the target language if that language was top-ranked by the language filter. To test the performance of the filter on Slovenian Web-pages, we asked a native speaker to evaluate 100 randomly selected web-pages from a list of several thousand classified as Slovenian by the language filter. 99 of these were in Slovenian, giving a precision of nearly 100%. An analogous evaluation for web pages judged to be negative shows that 90-95% of the pages classified as non-Slovenian were actually non-Slovenian. All our results are reported in terms of this automatic language classification. No additional manual evaluation was carried out.

4. Choosing Query Parameters

In Section 3.3, we described the term-selection methods for generating queries. We conducted exhaustive experiments comparing the performance of all the term-selection methods while varying the length of the queries to gain insight into their relative performance. The minority concept we use for our experiments is that of Slovenian on the Web. These experiments used three different initial documents and we found that the variance in the results was small. The evaluation measures we used were (a) percentage of documents retrieved in the target class (`PosDocs`) and (b) the number of documents in the target class per unique web query (`PosQueries`). We compared the term selection methods according to these two performance measures for each length independently.

For each document-based experiment, our system had access to one positive document in the target language. For experiments with Slovenian we supplied four negative example documents, one each in English, Czech, Croatian and Serbian. In all experiments, the language model for the language filter was also supplied.

4.1 Fixed Query Parameters

A summary of results for query-generation methods is given in Table 1, while detailed graphs of query-length, query method, documents retrieved and queries issued are shown in Figure 2. *Odds-ratio* (OR) is consistently the best with respect to both evaluation measures. Observing the number of queries issued, *odds-ratio* (OR) finds the greatest number of target documents. In terms of the number of documents examined, *odds-ratio* is again the best (length 1 — 3) except when all methods have about the same performance (length ≥ 4). For the two probabilistic methods PTF and PO the terms were probabilistically selected according to their score assigned by *term-frequency* and *odds-ratio* respectively. We also tested a variant (PORH) selecting only among the top 50% of the terms (ranked according to their scores) and as can be seen from the results in Figure 2, we found almost no difference between selecting probabilistically among all or just among the top half of the terms. Note the differences in scale on the graphs in Figure 2. Longer queries are much more likely to return no documents at all, and so can be costly. *Odds-ratio* performed best for all query-lengths except 5, where *term-frequency* found many documents with few queries, contrasting with query-length 1, where it found the least.

The comparison of different term selection methods given in Tabe 1 gives values after retrieving 3000 documents and values for 1000 issued queries, unless the result is marked by *, where the values are given

Len	Methods ordered by their performance	
	wrt PosDocs measure after retrieving 3000 docs	wrt PosQueries measure after issuing 1000 queries
1	68.8%(OR)>46%(PO)>39%(UN)>19.1%(PTF)> 8.9%(TF)	1.6(OR)>0.43(PO)>0.36(UN)>0.25(PTF)>0.09(TF)
3	82.3%(OR)>65.8%(UN)*>64.1%(PTF)>33.2%(TF)>9.4%(PO)*	1.77(OR)>0.38(PTF)>0.18(TF)>0.094(PO)>0.035(UN)
5	92.4%(UN)*>92.3%(PO)*>81.5%(OR) >77.4%(PTF)>77.0%(TF)	1.2(OR)>0.53(TF)>0.14(PTF)>0.01(PO)>0.01(UN)
10	100%(PO)*>88.7%(PTF) >79.2%(OR)>50%(UN)*>7.0%(TF)	0.02(OR)>0.01(PTF)>0(TF)>0(PO)>0.001(UN)

Table 1. Comparison of different term selection methods for query length varied from 1 to 10.

at much lower number of documents. For instance, for PO and UN on length 5, less than 100 documents and for length 10 less than 5 documents. This shows that the two methods are mostly issuing queries which do not return any document (especially for longer queries) but the few successfully queries usually return the target class documents.

A closer look at the results given in Table 2 and Figure 2 shows, for example, the following. For length 1 and issuing 1000 queries, *odds-ratio* gets over 1650 target language documents, while *term-frequency* gets only about 90 of them. All the other methods are placed between the two getting about 250 (PTF) to 430 (PO) target language documents. After retrieving around 2000 documents using length 1, *odds-ratio* achieved precision of 0.69, while *probabilistic odds-ratio* achieved 0.46, *uniform* about 0.39, *probabilistic term-frequency* about 0.19 and *term-frequency* about 0.09 (see Table 1).

For length 3, after issuing 1000 queries, *odds-ratio* gets over 1750 target language documents, while *uniform* gets only 35 of them. All the other methods are placed between the two getting 94 (PO), 182 (TF) and 388 (PTF) target language documents. After retrieving around 3000 documents using length 3, the highest precision was achieved by *odds-ratio* 0.82 and *probabilistic term-frequency* 0.64. For length 5, after issuing 1000 queries, *odds-ratio* gets over 1100 target language documents, while *probabilistic odds-ratio* and *uniform* get only about 10 of them each. *term-frequency* with 530 documents and *probabilistic term-frequency* with 143 are between them. After 15 000 queries, *term-frequency* got about 4000 target language documents, *probabilistic term-frequency* 1699, *odds-ratio* 1465 while *probabilistic odds-ratio* and *uniform* had only 33 and 15 respectively. After retrieving around 3000 documents using length 5, the highest precision was achieved by *odds-ratio* 0.82, *probabilistic term-frequency* 0.77 and *term-frequency* 0.77. With higher lengths all the methods perform worse. Setting length to 7 resulted after 1000 queries with *odds-ratio* having about 1300 documents in the target language while all the other methods had much lower performance, *probabilistic term-frequency* with 10, *probabilistic odds-ratio* 4 and *term-frequency* and *uniform* 0. After retrieving around 1000 documents using length 7, the highest precision was achieved by *odds-ratio* 0.84, *probabilistic term-frequency* 0.78 while *term-frequency* had precision 0.82 on about 300 documents. *Probabilistic odds-ratio* had the maximum precision of 1.00 but retrieving only 20 documents (all of them in the target language) and using for that 56 855 (!) queries. For length 10 after issuing 1000 queries, no documents in the target language was found by *term-frequency* and *probabilistic odds-ratio*, one document was found by *uniform*, 10 by *probabilistic term-frequency* and 16 by *odds-ratio*. After retrieving around 150 documents using length 10, *probabilistic term-frequency* achieved precision 0.89 while *term-frequency* achieved only 0.07. *odds-ratio* achieved precision of 0.79 after retrieving 755 documents. *Uniform* achieved precision 0.5 retrieving only 2 documents ad using 6000 queries, while *probabilistic odds-ratio* achieved 1.0 retrieving only 4 documents and using 49177 queries.

We found that each method performs best at a small range of query lengths. For *term-frequency*, the best performance is achieved with length 4 (when 4 terms are included and 4 terms are excluded). For *probabilistic term-frequency* and *odds-ratio* length 3 gives the best results, while for *probabilistic odds-ratio* using more than 1 include and 1 exclude term gives a very small number of the target language documents while using a very large number of queries. Figures 3 and 4 show comparison of different lengths for the *term-frequency*, *probabilistic term-frequency*, *odds-ratio*, *probabilistic odds-ratio*, *rtfidf* and *uniform* term selection methods.

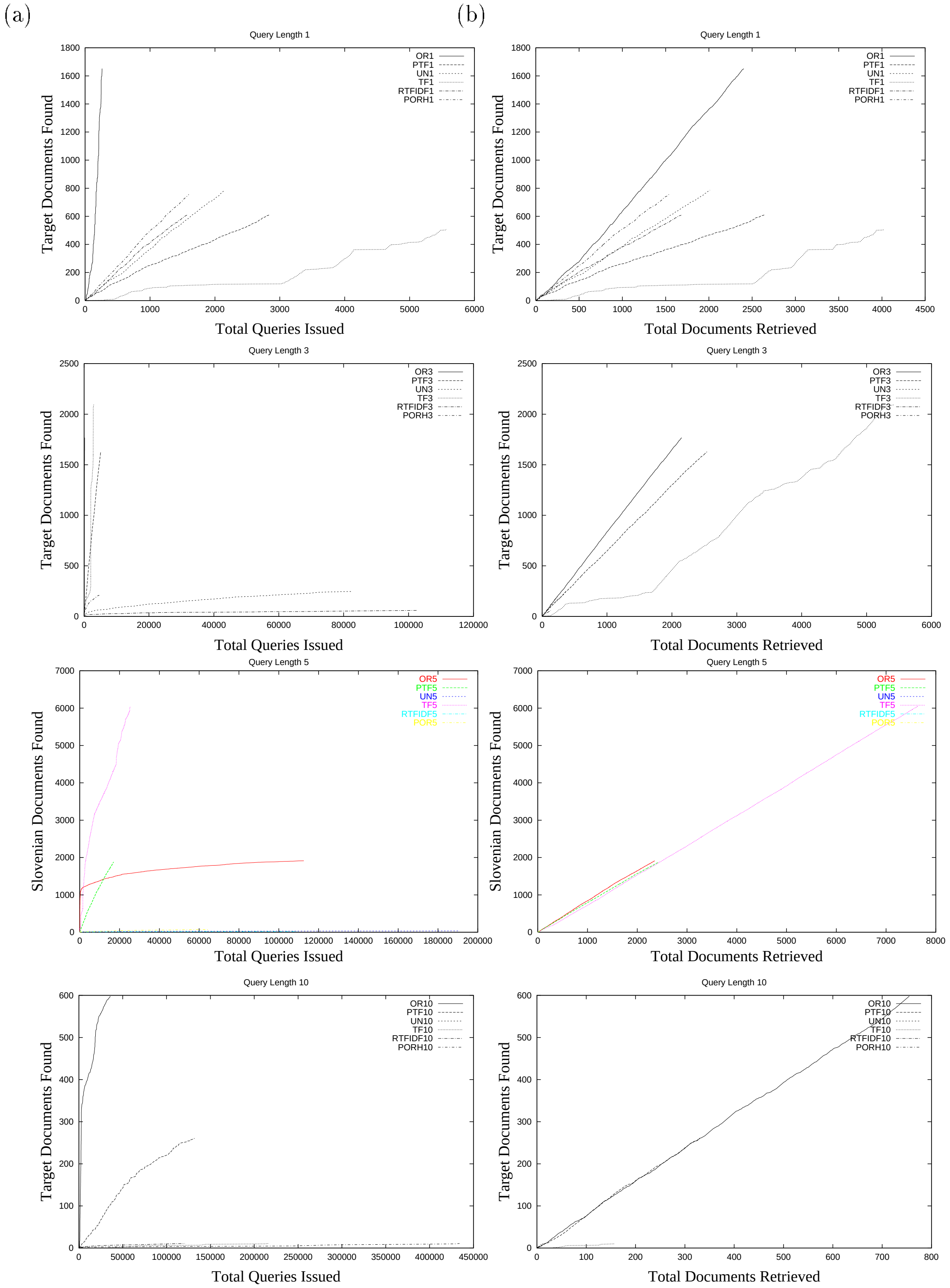


Figure 2. Comparison of different term selection methods for different query lengths measured against (a) the number of queries and (b) the number of total documents examined. (a) *Odds-ratio* consistently finds more documents given the same number of queries, except for length 5, where *term-frequency* finds high yield queries. Note the differences in scales on the graphs; longer queries (eg., length 10) are much more likely to return no documents at all, and so can be costly. b) *Odds-ratio* is consistently the most precise in finding Slovenian documents. Precision increases with the number of query terms, though many query-methods are able to find very few total target-language documents when using long queries.

Len	Performance measure no. target documents after	Method TF	 PTF	 OR	 PO	 UN
1	1000 queries	88	249	1652+	427	364
	2500 queries	118	529	—	640+	786+
	5000 queries	501	—	—	—	—
	1000 retrieved doc	93	261	636	461	382
	3000 retrieved doc	266	573	1652+ (2402)	640+ (1391)	786+ (2013)
	4000 retrieved doc	502	—	—	—	—
3	1000 queries	182	388	1768+	94	35
	2500 queries	1377	426+	—	110	51
	5000 queries	2094+	—	—	125	64
	15 000 queries	—	—	—	176	100
	25 000 queries	—	—	—	—	132
	100 000 queries	—	—	—	—	246+
	1000 retrieved doc	178	646	835	94	246+ (374)
	3000 retrieved doc	996	1629+ (2540)	1768+ (2149)	—	—
	5000 retrieved doc	1860	—	—	125	—
5	1000 queries	530	143	1167	11	10
	2500 queries	1586	349	1221	17	11
	5000 queries	2524	673	1282	21	12
	15 000 queries	4066	1699	1465	33	15
	25 000 queries	5916	—	1572	46	22
	100 000 queries	—	—	1889	73+	30
	1000 retrieved doc	721	799	844	73+ (79)	36+ (39)
	3000 retrieved doc	2310	1881+ (2431)	1913+ (2348)	—	—
	5000 retrieved doc	3915	—	—	—	—
	7000 retrieved doc	5551	—	—	—	—
7	1000 queries	0	10	1384	4	2
	2500 queries	0	55	1523+	6	3
	5000 queries	0	167	—	7	3
	15 000 queries	5	289+	—	10	5
	25 000 queries	356	—	—	11	5
	100 000 queries	363	—	—	20+	9
	1000 retrieved doc	417+ (536)	289+ (352)	831	20+ (20)	10+ (10)
	3000 retrieved doc	—	—	1523 (1823)	—	—
	5000 retrieved doc	—	—	—	—	—
10	1000 queries	0	10	16	0	1+
	2500 queries	0	25	243	1	—
	5000 queries	0	45	362	2	—
	15 000 queries	0	102	436	3	—
	25 000 queries	0	—	557	3	–
	100 000 queries	6	—	—	4+	–
	1000 retrieved doc	11+ (157)	102+ (115)	598+ (755)	4+ (4)	1+ (2)
	3000 retrieved doc	—	—	—	—	—
	5000 retrieved doc	—	—	—	—	—

Table 2. Performance of different term selection methods for each query length varied from 1 to 10. Sign "+" next to the number indicates that there were more target documents, but the experiments stopped before reaching the indicated number of retrieved documents or queries.

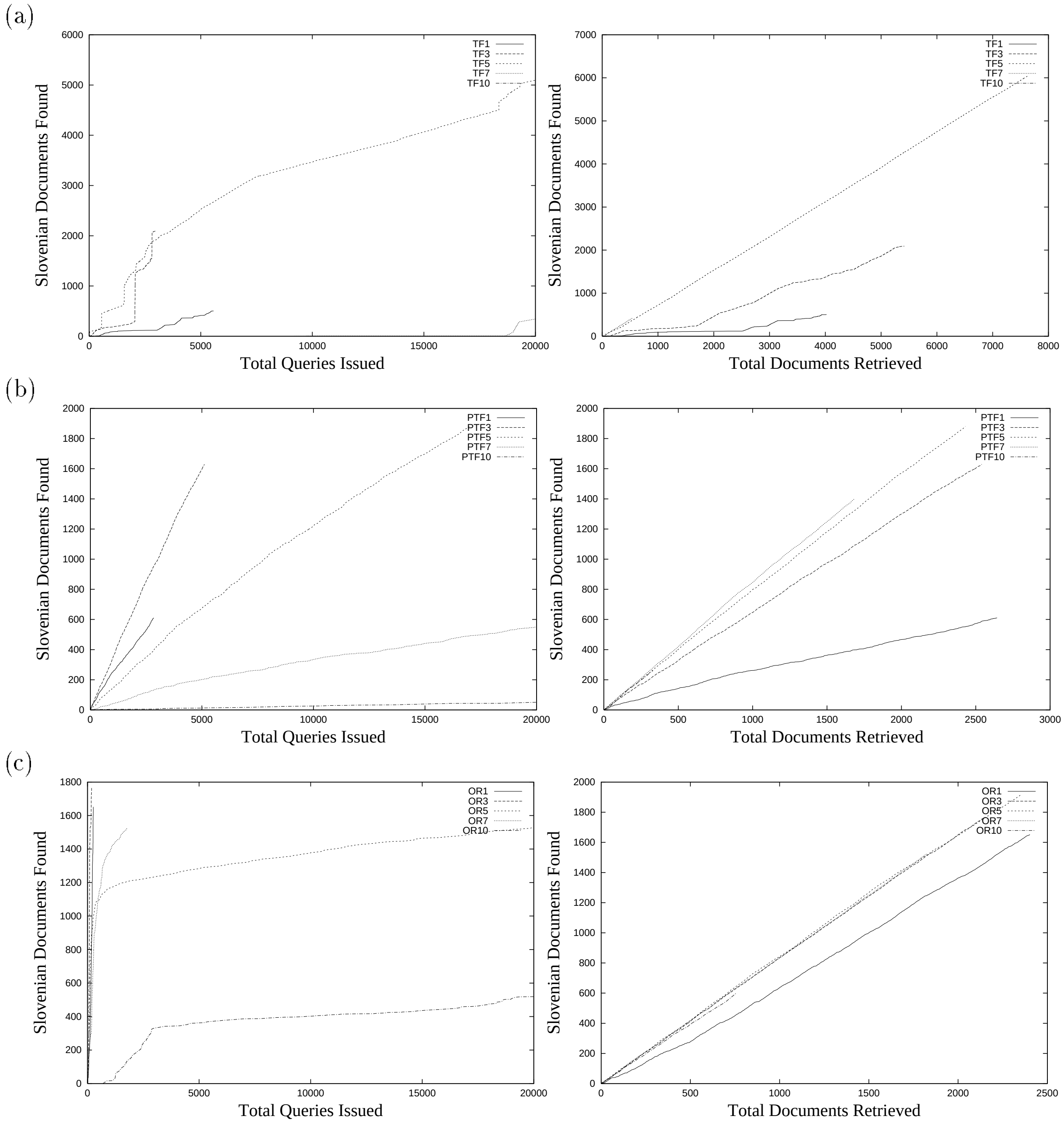


Figure 3. Comparison of performance for the same term selection methods using different lengths. (a) For *term-frequency*, the best performance is achieved with length 4 (when 4 terms are included and 4 terms are excluded). (b) For *probabilistic term-frequency* and (c) *odds-ratio* length 3 gives the best results.

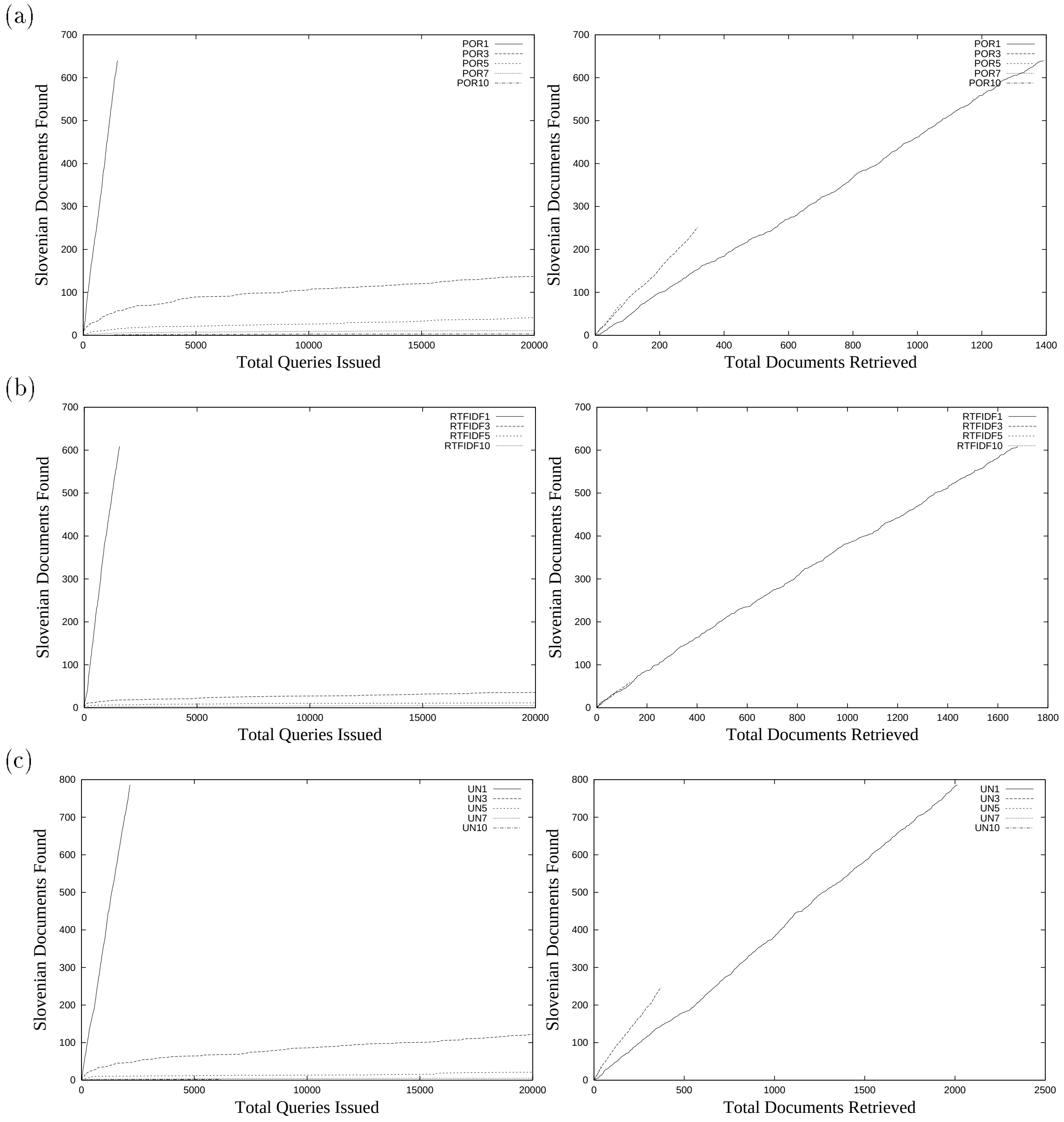


Figure 4. Comparison of performance for the same term selection methods using different lengths. (a) For *probabilistic odds-ratio*, (b) *rtfidf* and (c) *uniform* using more than 1 include and 1 exclude term gives a very small number of the target language documents while using a very high number of queries.

5. Learning query parameters

As described in section 4.1, different methods excel with different query lengths. For *term-frequency*, for example, the best performance is achieved with length 4 (when 4 terms are included and 4 terms are excluded). While the previous experiments permit us to see how different methods and query lengths perform in isolation, it is still possible that the best overall querying strategy would use one method and length initially, then change the ideal method and length as more and more documents in the target language are retrieved. As corpus construction proceeds, our system may explore different parts of Web/feature space 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 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.

5.1 Learning Overview

Since our queries can be described by four parameters: Inclusion Term-Selection Method, Exclusion Term-Selection Method, Inclusion Length, and Exclusion Length. Our learning methods should learn the ideal parameters for a given target concept and adapt to changing environments. Since we believe that the target concept is shifting (we always want previously unseen documents) 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 (when the documents it finds have been exhausted), 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 (*term-frequency*, *probabilistic term-frequency*, *odds-ratio*, and *probabilistic odds-ratio*) 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 (according to the multinomial distribution) 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 (described in Table **??**).

Our general goal is to find a good querying mechanism in the shortest time possible. In this way, there is a trade-off between exploration and exploitation. A method which quickly finds a reasonable querying mechanism can then exploit that mechanism. However, an algorithm which spends more time searching for a very strong querying mechanism may do better over the long term. Depending on whether our goal is to acquire as many documents as possible, a fixed number, or documents with sufficient vocabulary coverage, the ideal measurement statistic can vary.

5.2 Learning Methods

We performed experiments varying the time horizon used in learning: 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 and useful in generating the next query. At the same time, the aggregated knowledge from past queries may prove invaluable for learning about the task as a whole.

Here we describe the implementation of each method for on-line learning (Blum, 1996) of query-generation methods. In the unparameterized-length setting the same algorithm is used for learning query lengths.

5.2.1 Memory-Less Learning (ML)

This was designed to permit a successful querying method to continue as long as it was finding positive documents. The algorithm is as follows:

- Initialize: set uniform distribution over methods m in $D(m)$

- while(1)
  1. Select a query-generation method m_i according to $D(m)$
  2. Generate a query with m_i; fetch a document d
  3. Update scores (repeat the previous method if successful, if not switch to some other method)
     - if d is in the target language
       $D(m_i) = 1, \forall m_j, j \neq i, D(m_j) = 0$
     - else { *d is not in the target language or no new documents were found*
       $D(m_i) = 0, \forall m_j, j \neq i, D(m_j) = \frac{1}{|M|-1}$ }
  4. Normalize scores to give probability distribution over methods, store in $D(m)$

Note that in step 3, the most recent method has probability 1 of being selected on success, otherwise it has a zero probability of being selected, with a uniform probability distribution over the remaining methods.

5.2.2 Long-Term Memory Learning (LT)

This method estimates each method's future probability of success based equally on all past performance. We used two kinds of updating rules: additive update (LTA) outlined below and multiplicative (LTM), Winnow-like update using $\beta = 0.5$. In Winnow-like updating, for the successful query-generation method m (either length or term-selection method) we use $Score(m)$ *= $\frac{1}{\beta}$, and for an unsuccessful length or scoring method we use $Score(m)$ *= β.

- Initialize:
  - assign each query-generation method m a positive score of 1, corresponding to a prior of one target language document fetched; store scores in $Pos(m)$.
  - assign each query-generation method m a negative score of 1, corresponding to one non-target language document fetched; store scores in $Neg(m)$.
  - $Score(m) = \frac{Pos(m)}{Pos(m)+Neg(m)}$
  - Normalize scores to give probability distribution of methods, store in $D(m)$
- while(1)
  - Select a query-generation method m_i according to $D(m)$
  - Generate a query with m_i; fetch a document d
  - Update scores
    * if d is in the target language
      $Pos(m_i)$ += 1
    * else *d is not in the target language or no new documents were found*
      $Neg(m_i)$ += 1
  - Recalculate $Score(m_i) = \frac{Pos(m_i)}{Pos(m_i)+Neg(m_i)}$
  - Normalize scores to give probability distribution over methods, store in $D(m)$

5.2.3 Fading Memory Learning (FM)

This algorithm bases some of the current performance on the past, but gradually reduces the impact of learning experiences further in the past. The algorithm is as follows:

- Initialize: assign each query-generation method m a score of 1, corresponding to a prior of one target language document fetched; store scores in $S(m)$. Normalize scores to give uniform probability distribution $D(m)$ over methods.
- while(1)
  - Select a query-generation method m_i according to $D(m)$
  - Generate a query with m_i;fetch a document d
  - Update:
    * decay scores: $\forall m$, $D(m)$ *= α
    * update scores:
      1. if d is in the target language
         $S(m_i)$ += 1

2. else { d is not in the target language or no new documents were found }
$\forall m \neq m_i : S(m) \mathrel{+}= \frac{1}{(|M|-1)}$

* renormalize: normalize scores in $S(m)$ and store as probability distribution in $D(m)$

For all experiments we set $\alpha = 0.9$. Note that in step 2, we increase the scores of uninvolved methods in the event of failure.

5.3 Results

5.3.1 Fixed-length Queries

In this experiment, to reduce the number of parameters to be learned, we fixed the query length for both inclusion and exclusion terms. This is symbolized in graph titles by "IF" (assume methods are **I**ndependent and take **F**ixed length). We ran experiments for each length separately. We found that the best performance in terms of the number of target-language documents found was achieved by lengths 3 and higher, but in terms of the number of queries used, lengths 1 to 3 were the best, closely followed by length 5, while higher lengths used many more queries for the same number of the target-language documents. From this we can conclude that the best performance is achieved when using length 3—5 (see Figure 5). Shorter queries are more successful in getting documents but less accurate than the longer queries.

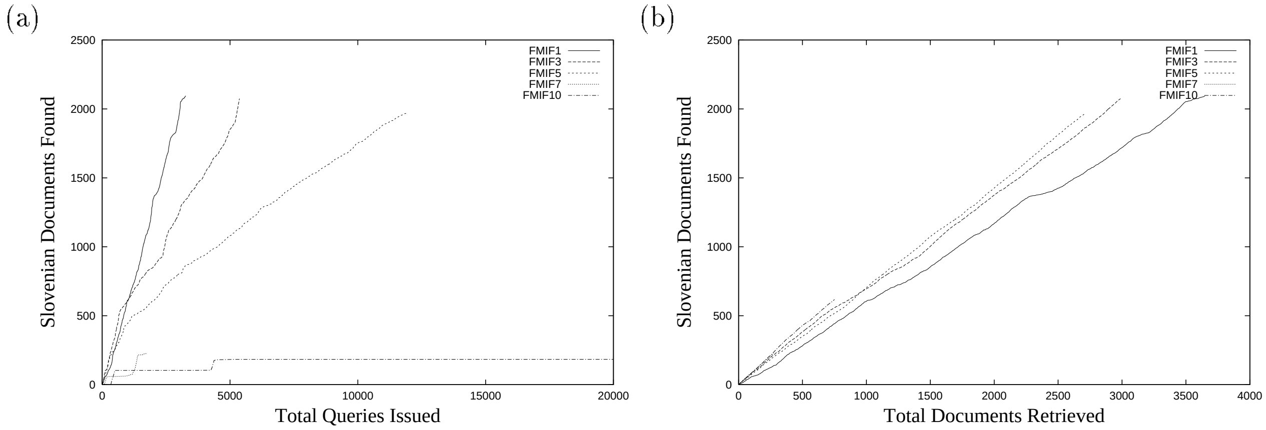


Figure 5. Fading Memory (FM) with fixed Query Length (IF) for query-lengths 1, 3, 5, 7 and 10 on Slovenian. The query-selection method is probabilistically learned from a decaying record of past successes and failures. (a) With shorter queries, we make less overall queries, as most return one or more documents. (b) With longer queries, we may make more hits on the search engine, but will find target-language documents a high proportion of the time, when we find any at all.

5.3.2 Preferred Parameter Values

Our hypothesis here is that the learning methods will prefer the better-performing parameter values (length and term selection method) from the earlier experiments (see Table **??**). In order to test this, we recorded probabilities of different parameter values at each iteration for different learning methods. The results in Table 5.3.2 show that *odds-ratio* (OR) was preferred by all the learning methods for selecting include-terms and that the number of include terms was typically lower than the number of exclude terms (1—7 vs. 4—10). For selecting exclude terms, *probabilistic odds-ratio* was preferred in most cases, and *Memory-Less* approach didn't make difference in performance of the available term-selection methods. More precisely, `LTA` preferred *odds-ratio* to select 2 to 5 include terms and *probabilistic odds-ratio* to select 6 to 10 exclude terms. `FM` preferred *odds-ratio* to select 1 to 4 include terms and *odds-ratio* to select any number of exclude terms. `ML` preferred *odds-ratio* to select 1 to 7 include terms and any of the term-selection methods to select 6 to 8 exclude terms.

Figure 6 shows how the probabilities of different parameter values change as learning progresses on example of the Long-Term Memory method `LTA`. It can be seen that very early in the process of learning, the most

Learning method	Parameter values ordered by their performance inside different learning methods			
	include terms	exclude terms	include method	exclude method
LTA	$2 \leq$ Len ≤ 5	$6 \leq$ Len ≤ 10	$OR > PTF > TF > PO$	$PO > OR > PTF > TF$
LTM	$1 \leq$ Len ≤ 5	$4 \leq$ Len ≤ 10	$OR > PTF > TF = PO$	$PO > OR > PTF = TF$
FM	Len ≤ 4	all equal	$OR > PTF = TF > PO$	$PO = OR > PTF = TF$
ML	$1 \leq$ Len ≤ 7	$6 \leq$ Len ≤ 8	$OR > PTF = TF > PO$	all equal

Table 3. Comparison of parameter values preferred by different learning methods. All learning methods focussed on *odds-ratio* as a inclusion term selection method, showing that they are able to find the best-performing method.

successful length for the include part of the query is around 3 and that *odds-ratio* is the most successful for choosing the inclusion terms. For the other learning methods there is no such clear convergence either in the length nor in the query method.

5.3.3 Comparison of Learning Methods

We compared different learning methods: Memory-Less `ML`, Long-Term Memory `LTA`, `LTM` and Fading-Memory `FM` described in Section 5.2. For methods involving rewards based on success (`LTA` and `FM`) we also tried a variant penalizing more for returning no document than for returning a negative document (negative documents give us some information to update our language model, while no document means wasted query). However, we didn't observe substantial differences in their performance.

Our next hypothesis is that different learning methods will differ in their performance, since they use different time horizons in the on-line learning process. The performance is measured the same way as in experiments with fixed query parameters described in Section 4, as the number of the target concept documents depending on the number of retrieved documents and on the number of queries issued, as a way of capturing accuracy and efficiency.

When comparing different learning methods, Long-Term Memory and Fading Memory learning perform better than Memory-Less (Figure 7). Long-Term Memory learning dominates both Fading Memory and Memory-Less in terms of the number of documents retrieved, as well as queries issued. The difference is more evident when retrieving 1000—2000 documents and issuing 1000—4000 queries. However, all the 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.

6. Generalizations

In order to examine how our experiments generalize to other conditions, we examined the effects of varying the initialization conditions to a variety of documents, and to a handful of keywords. We also varied the target language from Slovenian to Croatian, Czech and Tagalog.

6.1 Generalizations of initial conditions

6.1.1 Using Different Initial Documents

Since all the query generation methods described in this paper derive the query terms from documents already found, it is important to consider the effect of varying the initial document used. We used three different initial positive documents in Slovenian. The properties of these initial documents are shown in Table 4.

The results of running fixed query parameters experiments using *odds-ratio* with length 3 under different initial conditions are given in Part a) of Figure 8. The other methods and lengths show similar behavior. It can be seen that the method performs comparably for all initial conditions, getting about the same proportion of target documents. At the time the results diverged, over 1000 positive documents had been

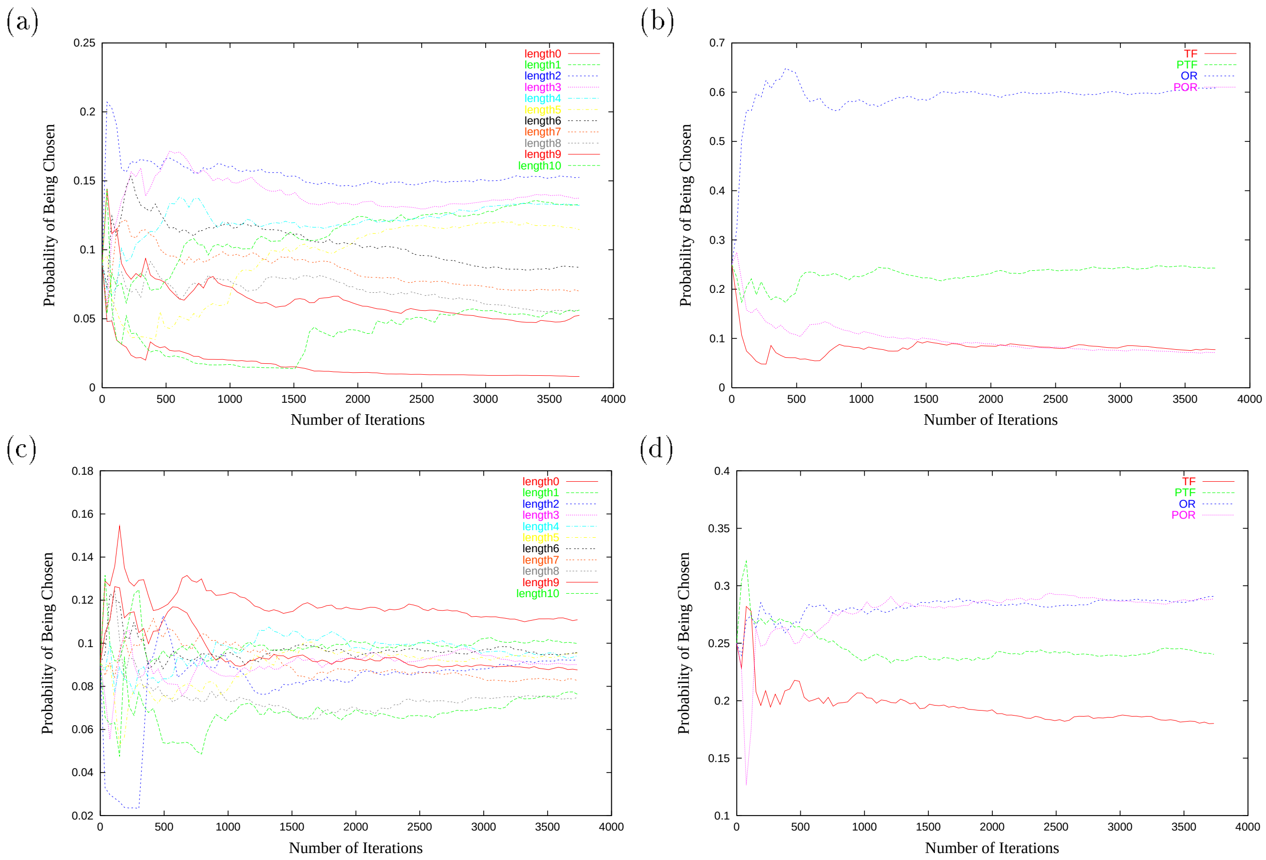


Figure 6. Probabilities of the four parameters when using Long-Term Memory with Multiplicative update rules for learning length and method. These probabilities reflect the relative performance of different lengths and methods we observed in experiments where no learning was involved. (a) For include length, 3 and 4 dominate. , namely after 2000 updates the highest probability have lengths 3 and 4, followed by 2, 5 and 6. Using 0 or more than 7 include terms was not successful. (b) For include method *odds-ratio* has the highest probability. then *probabilistic term-frequency*, *term-frequency* and *probabilistic odds-ratio*. (c) Exclude length 0 followed by other lengths lower than 7 dominate after 2000 updates. (d) For exclude method *odds-ratio* has the highest probability followed by *probabilistic odds-ratio*.

found. It is interesting to note that "Formal" and "News" initial documents needed less queries to retrieve the first 200 documents, while the "Informal" initial document retrieved over 1000 target documents only after issuing about 700 queries. Experiments comparing learning starting with different initial documents and starting with the user-provided keywords show that all sets of words led to similar results. Part b) of Figure 8 shows the results for Long-Term Memory approach, the other approaches exhibit similar behavior.

6.1.2 Initializing with User-supplied Keywords

It may not always be easy to find entire documents in a language to supply to our system. As an alternative to starting with a positive document in the target language, we asked three native speakers of Slovenian to supply us with words. We asked three questions of each native speaker, to elicit initial words of varying types. We asked for ten *common* words, to obtain words a Slovenian may be first likely to think of. We

	Type	Topic	Length	Vocabulary Size
1	Formal	Horticulture	568	354
2	News	Politics	716	446
3	Informal	Personal Web-Page	90	74

Table 4. Description of the 3 different initial documents used in experiments

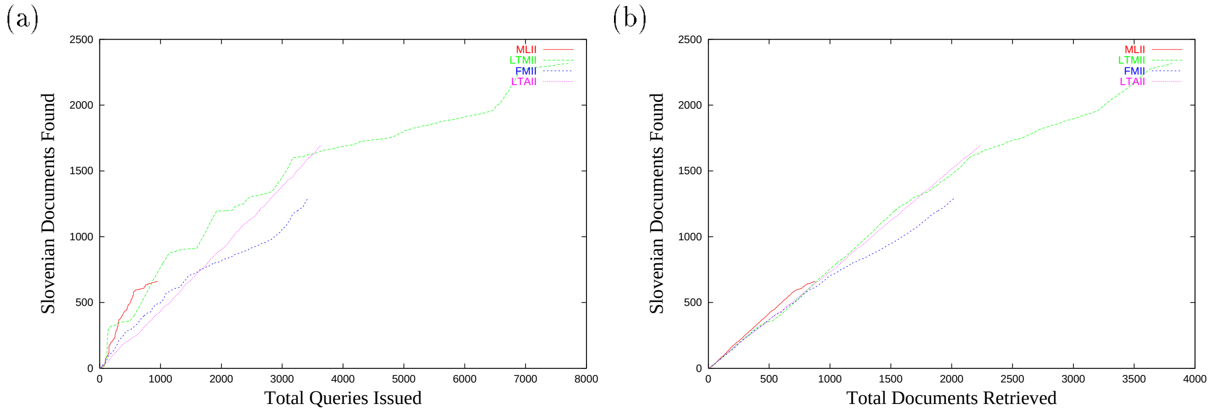


Figure 7. Comparison of learning methods on Slovenian language assuming independance between the methods, as well as between the lengths (hence `II` in the name of the methods on the graph). (a) In terms of queries Long-Term Memory using Winnow-like update `LTMII` performs the best. (b) In terms of retrieved documents, after 700—1000 documents LTMII is again the best.

then asked for ten *unique*ly Slovenian words. Croatian and Czech, for example, share many strings and words in common with Slovenian. A native speaker of Slovenian may know words which are shared with other languages. They may, however, not know that some are shared with other languages. Finally we asked for ten words *useful* for finding texts in Slovenian on the web.

The words we obtained from native Slovenian speakers shown in Table 5 were used as mock initial documents, and 10 English stop-words as the initial negative document. Figure 9 shows experimental results using fixed query parameters set to *odds-ratio* with length 3 and learning query parameters using Long-Term Memory with multiplicative update rule. All sets of words led to similar results, moreover the results are similar to the results of starting with initial documents shown Figure 8.

6.2 Other Languages

Systematic experiments with Slovenian by themselves do not provide any evidence that our techniques can be generalized to other languages. To test the hypothesis that our approach will indeed perform well on many languages, we repeat some of the experiments for Tagalog, Croatian and Czech, which can all be considered minority languages on the Web. Moreover, Slovenian, Croatian and Czech are all Slavic languages and share many words.

As shown in Table 6 and Figure 10, *odds-ratio* found more target documents than both *term-frequency* and *probabilistic term-frequency* for the same number of total documents examined, for all of Slovenian, Croatian, Czech and Tagalog. This suggests that the superiority of this query-generation mechanism generalizes across languages. Table 7 and Figure 10, however, show that the number of queries is dependent on the target language and method. This may reflect the number of web-pages which are confusable between Slovenian, Czech and Croatian when queries are made with frequent-words, whereas Tagalog's frequent words are more unique on the Web.

6.3 Learning Parameters for Other Languages

In order to test the generalization power of our learning methods for different target-languages, we performed experiments on two other natural languages, Croatian and Tagalog, also representatives 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. On Tagalog, as in Slovenian, the Winnow-like update rule gives the best performance while on Croatian, its performance is the worst and the additive update rule is the best performing. In Figure 11 we show the number of documents retrieved for experiments using Croatian and Tagalog.

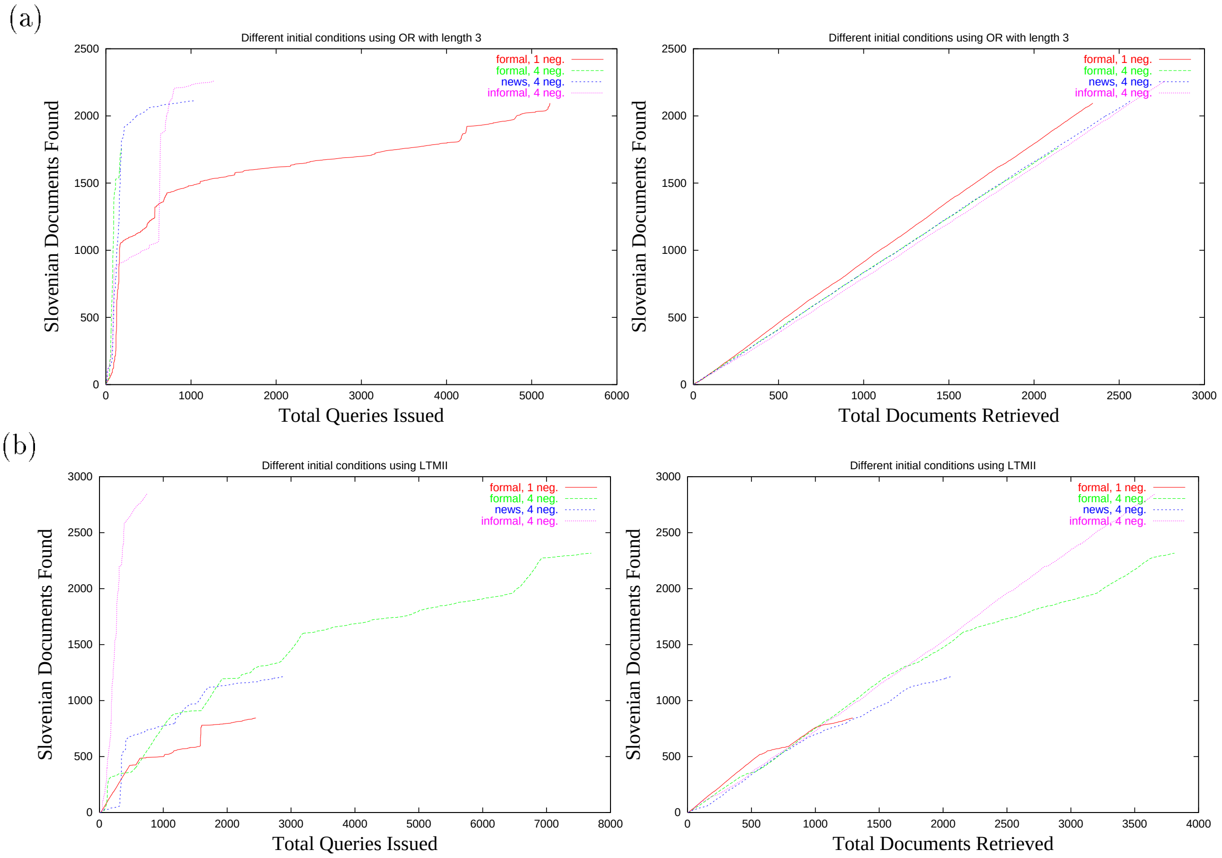


Figure 8. Comparison of four differnt initialization conditions for collecting documents in Slovenian. In three cases, the same four documents, one each of Czech, Croatian, English, and Serbian, were used as initial negative documents. A single different initial positive document was used in each case: formal text, informal text or news text. In the fourth experiment (`formal, 1 neg.`), the formal Slovenian document was used as the initial positive document, and a single English document was used as the initial negative document. (a) Under all four initial conditions *odds-ratio* with length 3 performs with similar results in the number of documents retrieved. Differences emerge long after the initial document is dwarfed by the presence of over 1000 other documents. However, initial conditions influence the number of queries needed for the same amount of target documents. (b) When learning query parameters, the Long-Term Memory approach using a multiplicative update rule performs similarly in the number of documents retrieved for all conditions. Differences emerge long after the initial document is dwarfed by the presence of over 600 other documents. In the number of queries issued, the "informal" initial document outperforms the others by retrieving almost 3000 target documents after 900 queries. Using only English as negative initial document results in higher number of queries needed for the same number of target documents.

Speaker 1	
common	da, pa, in, je, bi, si, bo, a, se, ki
unique	jaz, hisa, ogenj, dez, gozd, hlod, zlikrofi, struklji, okno, cevelj, najin, vajin, njun
useful	splet, stran, podjetje, vsebina, vsak, ker, miza, hisa, kazalo, povezava
Speaker 2	
common	janez, delo, hribi, omara, promet, sonce, papir, stena, ker, hrana
unique	hrepenenje, karkoli, strani, enajst, zoprno, trkanje, uporabljati, splet, kozolec, navodilo
useful	izmenjava, zoprno, karkoli, cvetje, velikanski, zmeda, gostilne, prehajanje, obsijal, gozdar
Speaker 3	
common	miza, govoriti, danes, sola, racun, pivo, kosilo, zoga, avto, smucke
unique	skatla, potica, brisaca, vrtec, menjalnica, lesnik, morje, zamuda, pepelnik, bolezen
useful	kakor, bil, je, ker, lahko, brez, s, z, ne, tudi

Table 5. Words supplied by native Slovenian speakers for Use in Automatic Query Construction. Speaker 1 provided function words as *common*, while Speaker 2 and Speaker 3 provided common nouns and names such as Slovenian for "John" (janez), "work" (delo), "table" (miza), and "school" (sola). For *unique* words, all speakers provided the names of traditional Slovenian foods and artifacts. Additionally Speaker 2 also provided uniquely Slovenian words that are not likely to appear in other languages. For *useful* words, Speaker 1 provided internet and computer-related terms, such as Slovenian for "Web" (splet), "page" (stran), "content" (vsebina) and "directory" (kazalo), while Speaker 2 provided words useful for finding different topics on the Web, such as Slovenian for "exchange" (izmenjava), "flowers" (cvetje), and "restaurants" (gostilne). Speaker 3 provided function words as *useful*. All sets of words led to similar results, moreover the results are similar to the results of starting with an initial document instead. Regardless of the starting conditions, CorpusBuilder generated queries which brought in Slovenian documents 80% of the time, when OR-3 was used as the query-generation method. Note that we substituted non-ASCII characters in Slovenian (č, š, ž) with ASCII characters (c, s, z), which are commonly also used also in Web documents.

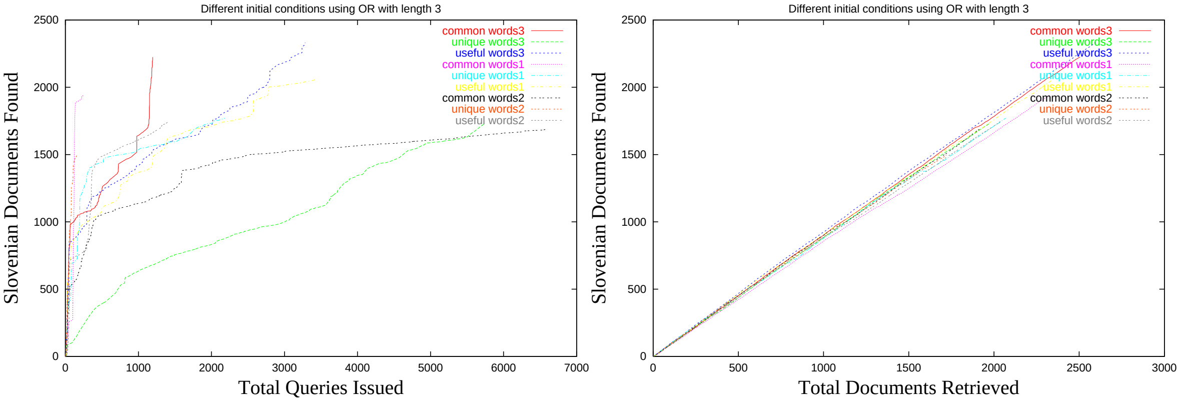
Method	Number of Target Docs at Total 1000 Docs Retrieved			
	Slovenian	Croatian	Czech	Tagalog
TF-3	178	39	385	440
PTF-3	646	410	451	359
OR-3	**835**	**677**	**743**	**664**

Table 6. *Odds-ratio* found more target documents than both *term-frequency* and *probabilistic term-frequency* for the same number of total documents examined, for all of Slovenian, Croatian, Czech and Tagalog.

Method	Language	Docs	Typical Query
TF-3	Slovenian	182	+in +v +za -html -a -re
PTF-3	Slovenian	342	+v +bolj +podrocju -carapama -den -jest
OR-3	Slovenian	**1409**	+torej +bomo +nisem -attila -laszlo -kerdes
TF-3	Croatian	61	+i +u +za -de -o -a
PTF-3	Croatian	231	+su +u +osobito -vozil -iso -pripad
OR-3	Croatian	93	+allah +takodjer +uzviseni -ief -mgtf -summary
TF-3	Czech	286	+na +v +pro -re -the -a
PTF-3	Czech	198	+v +na +s -zakroky -za -ga
OR-3	Czech	**680**	+vldy +mt +tden -found -davky -dite
TF-3	Tagalog	**793**	+sa +ng +ang -the -to -a
PTF-3	Tagalog	262	+unang +ang +dati -obey -optik -beef
OR-3	Tagalog	236	+niya +walang +siya -i -za -server

Table 7. Number of target documents found at 1000 queries. *odds-ratio* found more documents than both *term-frequency* and *probabilistic term-frequency* for the same number of queries, for Slovenian and Czech. Typical queries shown are those that most commonly found positive documents in our experiments. For PTF-3 every query was unique; a randomly selected one is shown.

(a)



(b)

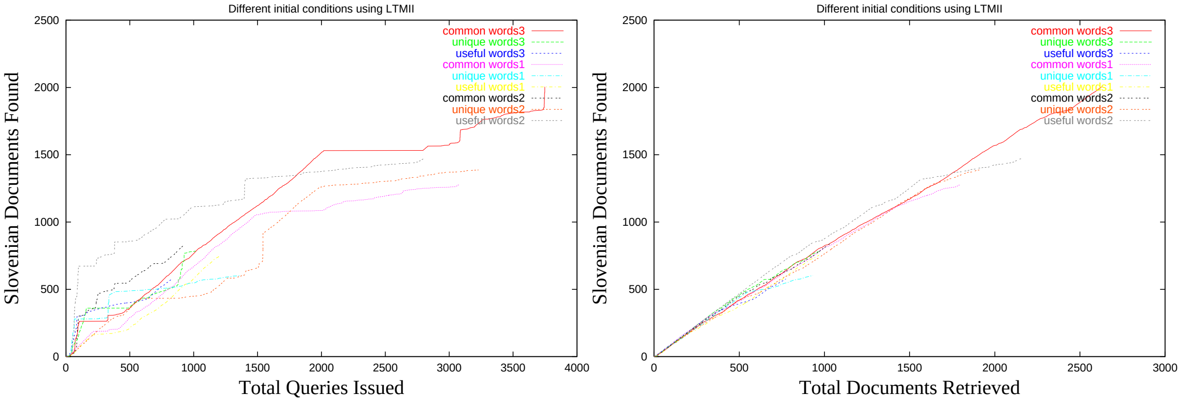


Figure 9. Using 10 keywords supplied by a native Slovenian speaker as initial positive information, and 10 English stop-words as initial negative information performs similarly regardless the set of keywords used, measured in the number of documents retrieved. There are some differences between the keyword lists in the number of issued queries for retrieving the first several hundred documents. (a) When using fixed query parameters, *odds-ratio* of length 3, the first list of common words outperforms the other keyword lists by requiring less queries for the first 200 documents. (b) When learning query parameters, all the keyword lists perform similarly. The second list of useful words outperforms the other keyword lists by requiring less queries for the first 600 documents.

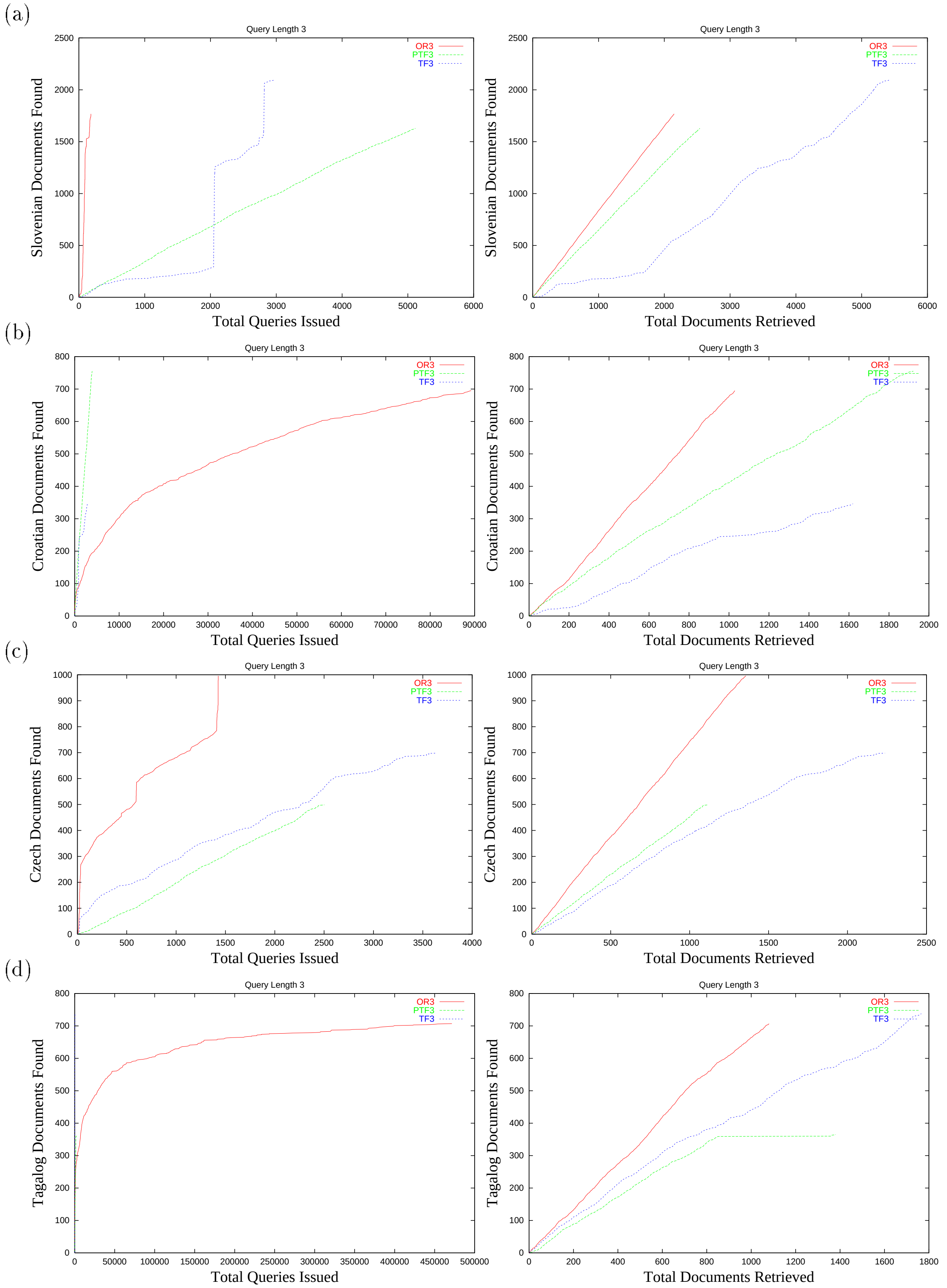


Figure 10. Comparison of different methods with length 3 over four different languages. For all four languages, *odds-ratio* performs the best in the number of the target documents after the same number of examined documents. For instance, after examining 1000 documents, (a) for Slovenian about 900 documents are in Slovenian and for the other three languages (b), (c), (d) about 700 documents are in the target language. There is a difference between the tested languages in the number of queries issued. (a) For Slovenian and (b) Czech *odds-ratio* outperforms the other methods by needing much less queries for the same number of target documents, while (b) for Croatian and (d) for Tagalog *odds-ratio* is outperformed by the other two methods. Note that in general for Croatian and Tagalog require many more queries to find documents than do Slovenian and Czech, as these languages are much more rare on the web.

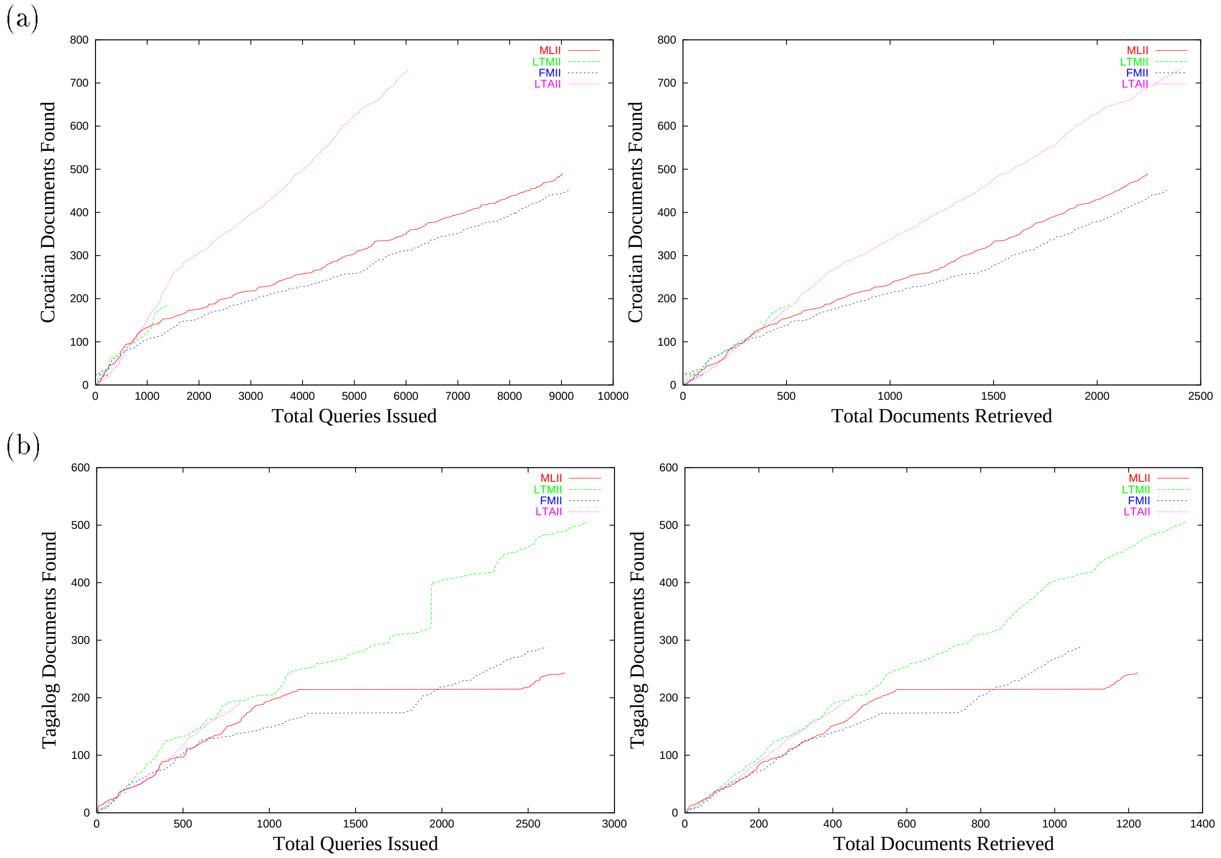


Figure 11. Comparison of learning methods on (a) Croatian and (b) Tagalog. For both languages, Long-Term Memory methods either using aditive (LTAII) or multiplicative (LTMII) update rule, perform the best.

7. Discussion

Our approach performs well at collecting documents in a minority language starting from a few words or documents but it does require a language filter for that minority language. There are filters available for quite a few language but this is potentially a limitation of our approach. In earlier work (Ghani & Jones, 2000), we experimented with constructing a filter on-the-fly, starting from the initial document and bootstrapping, and our experiments with Tagalog yielded encouraging results. We plan to experiment with this idea further and use different methods of building language filters with small amounts of labeled data.

Currently, we evaluate the corpus collected through our approach by sampling the corpus and verifying the decisions made by the language filter. Although it gives us information about the precision of our classifier and thus the corpus, it does not give any indication of the level of coverage obtained by our system. Since the number of total web pages in the minority languages we have worked on so far is not available, we can use approximation techniques for evaluation such as using the intersection of Web pages found by using each of the three documents as seeds for our experiments. We can also measure the rate at which we find new Slovenian documents as our experiments progress and a decreasing rate would give us a bound on the number of documents we can find using our methods. Callan et al. (**?**) and Ghani and Jones (Ghani & Jones, 2000) use various measures like percent vocabulary coverage and ctf to evaluate the coverage of their language models. Although our task is not to construct a language model for Slovenian and we do not have the "true" model to compare against, we can still use these measures and calculate of rate at which we add new vocabulary words and the distribution of the words we have in our vocabulary. Since we are sampling in the space of Slovenian words, convergence of the sampled distribution would indicate a reasonable coverage.

We utilize the resources of a search engine and focus on the task of generating queries that are highly accurate and cover different parts of the language space we are dealing with. There are several other ways that we could collect corpora from the Web and we discuss the merits and drawbacks of some of the other approaches below.

Using a dictionary of the minority language and manually selecting function words to query and then iterating a few times to eliminate function words that are shared with other languages is one approach that would work well but would require a lot of domain knowledge in the form of knowledge about function words for the particular language. Our approach learns from examples in that it only requires a handful of documents and requires very little domain knowledge.

We could also use a spider that crawls all pages under the domain names of the countries where that language is spoken. For example, in the case of Slovenian, we could spider all web pages in the .si domain. This would not result in good coverage or precision since not all the pages under the .si domain are in Slovenian and there are Web pages outside the .si domain that are in Slovenian. In our experiments, we found approximately 31,000 Web pages under the .si domain, 24% of them being classified as being not in Slovenian by our classifier. Similarly, out of the 30,000 Slovenian documents we found, 22% of them were not under the .si domain.

Another way of collecting a Slovenian corpus would be to crawl all the pages starting from the Regional → Countries → Slovenia category in Yahoo (or the category of the country where the language is used) . This again would not be accurate because Yahoo does not have an exhaustive list of Slovenian Web sites in general and of Web pages in Slovenian in particular.

One promising alternative we compared our approach with was using the built-in functionality of looking up "similar" web-pages that some search engines provide. The details of the comparison are as follows.

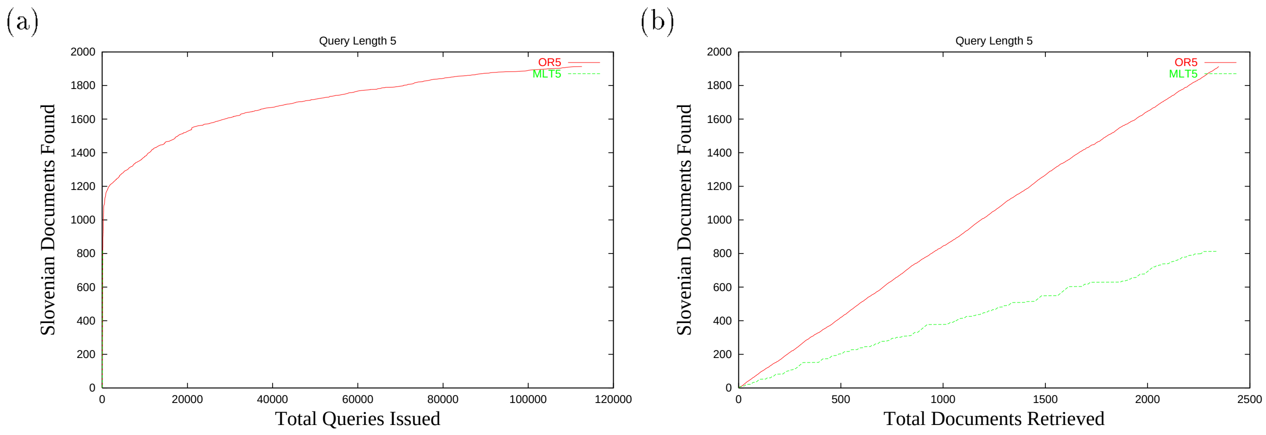


Figure 12. "More Like This" compared to Odds-Ratio (a) in the number of queries issued and (b) in the number of documents retrieved.

7.1 "Related Pages"

Search engines provide functionality to allow users to find similar pages, based on both content and hyperlink information (Dean & Henzinger, 1999). We hypothesize that our system can outperform the "Related Pages" feature and in order to test our hypothesis, we ran an experiment using AltaVista's "Related Pages" function. We start with the same initial documents as in the systematic Slovenian experiments described earlier, and use CorpusBuilder query generation, setting include and exclude lengths to 5 and the term selection method to *odds-ratio*. Then for each AltaVista hit classified as relevant by our language filter, we query AltaVista using *like:URL* to find documents "like" the one we retrieved then continue to repeat this process with all of the relevant ones, finding more "like" them each time.

The algorithm is as follows:

1. MoreLikeThisList = ()
2. PositiveDocList = InitialSlovenianDoc
3. While(1)
   - while we have no documents in the MoreLikeThisList
     – use the CorpusBuilder query generation method `generate_query(odds-ratio, 5, odds-ratio, 5)` to find AltaVista hits, add URLS of documents classified as positive to the MoreLikeThisList
   - while we have URLs in the MoreLikeThisList
     (a) take the first URL from the MoreLikeThisList
     (b) query AltaVista with "like:URL"
     (c) add URLS of documents classified as positive to the end of the MoreLikeThisList

We used `generate_query(odds-ratio, 5, odds-ratio, 5)` as this method had been performing well in other experiments. We find that *odds-ratio* outperforms MoreLikeThis, as can be seen in Figure 12. MoreLikeThis finds a document in the target language around 33% of the time, while using a query based on the odds-ratio term-selection method with 5 terms each for inclusion and exclusion finds a document in the target language around 90% of the time.

7.2 Using Existing Corpus-construction Techniques

Ghani and Jones (Ghani & Jones, 2000) showed that single word queries were sufficient for finding documents in Tagalog, and that selecting the query-words according to their probabilities in the current documents performed the best. It is important to note that their experiments were run on a small corpus[2]

[2]the corpus consisted of 500 Tagalog documents and 15000 documents mostly in English and Brazilian Portuguese

of Tagalog documents and other distractor documents collected from the web and stored on disk. We compared their best-performing methods against other query generation methods and lengths, on the tasks of finding both Tagalog and Slovenian documents on the Web.

Applying single-word *term-frequency* and *probabilistic term-frequency* queries to the Web for Slovenian results in relatively low precision, as shown in Figure 13. Using the *odds-ratio* query generation method described in section 3.3 outperforms the *probabilistic term-frequency* approach with single include and exclude-word queries. Furthermore, using more words in the query (3 for inclusion and 3 for exclusion) performs better than the single word queries previously used.

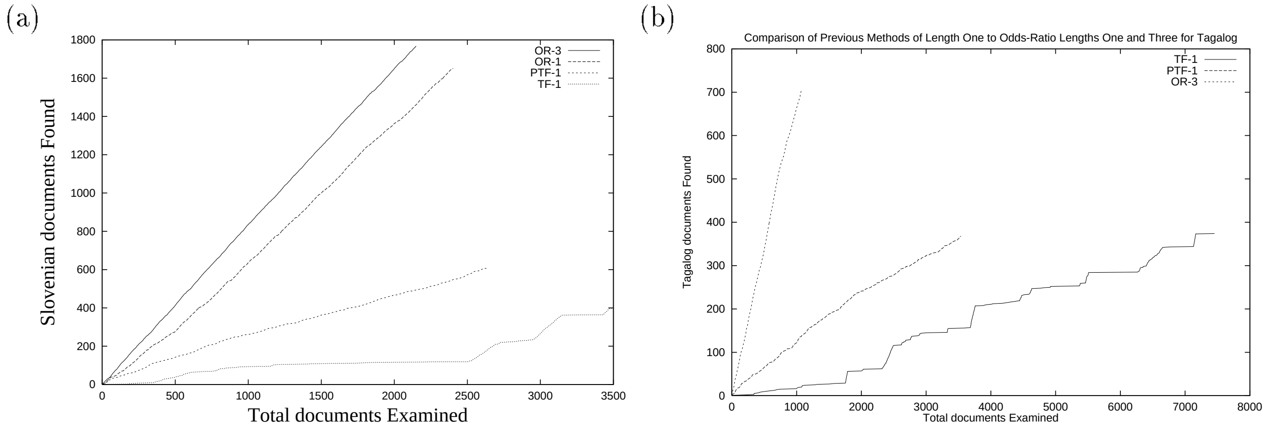


Figure 13. a) On Slovenian *term-frequency* (TF) and *probabilistic term-frequency* (PTF) term selection methods with one inclusion and one exclusion term were outperformed by *odds-ratio* (OR) with one inclusion and one exclusion term, and an even better-performing *odds-ratio* with 3 inclusion and 3 exclusion terms. b) On Tagalog less documents were found overall, but *odds-ratio* for 3 terms still dominated TF-1 and PTF-1.

To establish whether the better performance of *odds-ratio* was due to the choice of language, we performed the same evaluation on Tagalog. Although less documents were found overall, the trend remained the same, with OR-3 finding over 600 documents in Tagalog after examining 1000 documents, while OR-1 had found just over 300, PTF-1 had found 122 and TF-1 had found only 17 documents in Tagalog as shown in Figure 13.

7.3 Query Length Parameterization

Up till now we have been estimating parameters separately for each length and method by assuming that they are independent of each other. We can formalize these assumptions by proposing four different ways of parameterizing query-lengths. These differ according to two kinds of independence assumptions and combinations of these: (1) *Method and Length Independence* which would assume that ideal query-length is independent of the query-generation method used. This assumption means that less parameters need to be tuned. In our experiments we use this assumption of independence; experimentation with the other models is part of future work. With method and length dependence, the ideal query length depends on the method chosen. (2) *Length Independence* which would assume that query lengths are independent of one another. This is the multinomial query-length model, and uses more parameters (one per candidate length). This setting is denoted in our experiments with *II*. Another model requiring less parameters matches our intuition that similar length queries are likely to perform relatedly. To this end we parameterized query lengths as a Gamma distribution, denoted here as *ID*.

We can summarize the use of the independence and dependence models as follows:

	Method-Length Independent	Method-Length Dependent
Lengths Independent (Multinomial Model)	II	DD
Lengths Dependent (Gamma Model)	ID	DD

The abbreviations will be used when describing these combinations and In the experiments in this paper, we only used II and ID. Experimentation with the other models is part of future work. The previous experiments reported results using the II model for the parameters. We ran the same experiments again using the ID model for the learning methods using both variants of Long-Term Memory and Fading Memory. Using ID model actually hurts the performance of the Fading Memory Method and does not result in any improvement for the Long-Term Memory method when compared to the II model. Actually after about 5500 queries, the Long-Term Memory method using additive update rule LTAID catches up with LTMII.

8. Conclusions and Future Work

We presented an approach for automatically collecting web-pages in a minority language and showed that it performs well on several natural languages (Slovenian, Croatian, Czech and Tagalog) and only requires a handful of documents to start with.

We found that our basic *odds-ratio* query construction method outperforms our other methods and also performs better than simply using the existing Related Pages function in AltaVista. *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 in the 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 multiplicative or 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`.

In experiments investigating length parameterization, we found that using a Gamma distribution hurts the performance when Long-Term Memory is used for learning. Further experimental work is needed to draw solid conclusions about the consequences of the length independence assumption used in this work. We plan to look into using more than one document at each iteration so the learning process can be faster although that could come at the expense of precision.

An interesting question is how our results transfer to other target concepts such as collecting documents about some topic or getting documents that match a user profile. We conducted some preliminary experiments for collecting Course web pages starting from a handful of examples and we plan to pursue collecting topic-specific corpora in future work. Using these techniques to augment existing techniques for developing a domain specific search engine is also an interesting future direction.

Our approach for automatic query-generation is useful for any application where we are able to build a very accurate, but expensive classifier. We can then use the search engine as a simpler filtering mechanism, bringing in only those documents likely to satisfy our classifier, thus saving us the expense of running the classifier on all documents retrieved by the crawler. The kinds of high-precision classifiers which would benefit from this approach include systems with parsers, systems which fetch more web-pages to decide on the classification for a given web-page and systems which involve user input thus making our approach applicable in various useful and important areas.

Acknowledgements

The authors are grateful to Avrim Blum and Belinda Thom for valuable discussion and suggestions. The authors also wish to thank to Andrej Bauer, Janez Brank and Marko Grobelnik for contributing Slovenian words.

References

Blum, A. (1996). On-line algorithms in machine learning. *In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl, 1996..*

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.

Brown, P., Pietra, S. D., Pietra, V. D., & Mercer, R. (1993). The mathematics of statistical machine translation. *Computational Linguistics*, *19*.

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval* (pp. 161–175). Las Vegas, NV.

Chen, Z., Meng, X., Zhu, B., & Fowler, R. H. (2000). Websail: From on-line learning to web search. *Proceedings of the 2000 International Conference on Web Information Systems Engineering.*

Dean, J., & Henzinger, M. (1999). Finding related pages in the world wide web. *Proceedings of the Eighth International World Wide Web Conference..*

Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., & Gori, M. (2000). Focused crawling using context graphs. *26th International Conference on Very Large Databases* (pp. 527–534). Cairo, Egypt.

Ghani, R., & Jones, R. (2000). Learning a monolingual language model from a multilingual text database. *Proceedings of the Ninth International Conference on Information and Knowledge Management.*

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.

Golding, A. R., & Roth, D. (1999). A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, *34*, 107–130.

Haines, D., & Croft, B. (1993). Relevance feedback and inference networks. *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Jelinek, F. (1999). *Statistical methods for speech recognition.* MIT Press.

Liberman, M., & Cieri, C. (1998). The creation, distribution and use of linguistic data. *Proceedings of the First International Conference on Language Resources and Evaluation.* Grenada, Spain.

Mladenic, D., & Grobelnik, M. (1999). Feature selection for unbalanced class distribution and naive bayes. *Proceedings of the 16th International Conference on Machine Learning.*

Rennie, J., & McCallum, A. K. (1999). Using reinforcement learning to spider the web efficiently. *Proceedings of the 16th International Conference on Machine Learning.*

Resnik, P. (1999). Mining the web for bilingual text. *Proceedings of 34th Annual Meeting of the Association of Computational Linguistics.* Maryland.

van Noord, G. (1997). Textcat. http://odur.let.rug.nl/ vannoord/TextCat/.

Yang, Y., & Pedersen, J. (1997). A comparative study on feature selection in text categorization. *Proceedings of the 14th International Conference on Machine Learning.*