Building Minority Language Corpora by Learning to Generate Web Search Queries

Rayid Ghani\(^1\), Rosie Jones\(^2\) and Dunja Mladenic\(^3\)
\(^1\)Accenture Technology Labs, Chicago IL, USA
\(^2\)Carnegie Mellon University, Pittsburgh PA, USA
\(^3\)J. Stefan Institute, Ljubljana, Slovenia

Abstract. The web is a 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. The concept used for this paper is that 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. Using \textit{odds-ratio} scores calculated over the documents acquired was one of the most consistently accurate query-generation methods. To reduce the number of estimated parameters, we 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.

Keywords: Web Mining; Online Learning; Query Generation; Corpus Construction

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 linguis-
tic and sociolinguistic studies, as they are readily searchable and statistics can
easily be computed.

Current methods for creating text corpora for specific languages require much
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 web has been gaining popularity as a resource for multilingual content.
Resnik (Resnik, 1999) explored the web as a source for parallel text and auto-
matically 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 find a wide range of documents in the target language,
and that filter out a large proportion of closely related languages. Our hypothesis
is that by selecting appropriate inclusion and exclusion terms from documents
already collected, and by 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
show that 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 “Similar
Documents” option. We explore different methods for selecting query words and
use on-line learning to modify the queries based on feedback by the language
filter. We show that our system, initialized with a single document in the target
concept, 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 minority languages on the web.

2. Related Work

While search-engines are an invaluable means of accessing the web for users,
atomated 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 automatic-
ally augment user queries for specific documents with terms designed to find
document genres, such as home-pages and calls for papers. Rennie and McCal-
lum (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.

In earlier work (Ghani & Jones, 2000), we described an algorithm for building a language-specific corpus from the world-wide web. However, our experiments were limited to a small closed corpus of less than 20,000 documents, vastly limiting the generalization power of the results to the web. We 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 the experiments were run on a small corpus\(^1\) of Tagalog and other distractor documents collected from the web and stored on disk. We compare our earlier best-performing methods against the query generation methods and lengths presented in the current paper to find Tagalog and Slovenian documents on the web and find that applying single-word term-frequency and probabilistic term-frequency queries to the web for Slovenian results in relatively low precision and 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, query-generation methods and the language-filter used. Our approach differs from pseudo-relevance feedback (Robertson & Sparck Jones, 1976), (Rocchio, Jr., 1971) in that retrieved documents are labeled by an automatic language classifier as relevant or irrelevant, and this feedback is used to generate new queries.

3.1. General Algorithm

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

---

\(^1\) The corpus consisted of 500 Tagalog documents and 15000 documents mostly in English and Brazilian Portuguese.
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 a 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. Here we refer to a URL returned by the search engine as a hit. When querying the search engine with a new query, only the first hit is used but the remaining hits are stored to a file for efficiency, so that we avoid re-querying the search engine for the same query. If we have used the same query before, we take the next unseen hit from our cached results. If all hits have been seen, no hit is returned.

The general algorithm is as follows:

1. Initialize frequencies and scores based on relevant and non-relevant documents
2. Generate query terms from relevant and non-relevant documents
3. Retrieve next most relevant document for the query
4. Language Filter, assign document to relevant or non-relevant set
5. Update frequencies and scores 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 experimented with using a single initial document as well as with using only 10 keywords instead of an entire document. Term frequencies for all initial documents in the relevant language and irrelevant languages are calculated separately and used by the query generation methods.

3.3. Query Generation Methods

Given a collection of documents classified into relevant and non-relevant classes, 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 to (but not the same as) the relevant documents (i.e. also relevant) and not similar to the non-relevant documents. We can view this as sampling without replacement from the web. We use query generation and a search engine to sample from this document collection. (Ghani & Jones, 2000) experimented with sampling with and without replacement on a subset of the web and found that both strategies performed similarly for their dataset.

We construct queries using 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 online 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. 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 $k$ terms 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 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 \( \log \frac{\text{total number of documents}}{\text{number of documents containing the term}} \). 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

\[
\log \left( \frac{P(w|\text{relevant doc}) \cdot (1 - P(w|\text{nonrelevant doc}))}{P(w|\text{nonrelevant doc}) \cdot (1 - P(w|\text{relevant doc}))} \right)
\]

- *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 the words in the vocabulary (UN). Scoring terms according to their frequency (TF) has been 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 & Croft, 1993) show that rtfidf is a good scoring mechanism for information retrieval. (Mladenic & 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). Sample queries generated using several of these techniques for a variety of languages can be seen in Table 7.

The query generated at each step may be a novel query, or one that we have issued previously, 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 can only change 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 from the empty query. 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

We conducted exhaustive experiments comparing the performance of all the term-selection methods (described in section 3.3) 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). The higher the value of these two metrics, the better we judge our methods to be. 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 different query-generation methods is given in Table 2, while detailed graphs comparing query-lengths, query methods, 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 (between lengths 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. As observed from this table, longer queries do not perform well in terms of the number of relevant documents returned per query. Longer queries are much more likely to return no documents at all. Even if all the words contained in the query are from the target concept, they are unlikely to co-occur in the same document.
<table>
<thead>
<tr>
<th>Query Length</th>
<th>Methods ordered by their performance wrt PosDocs measure after retrieving 3000 docs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>68.8%(OR)&gt;46%(PO)&gt;39%(UN)&gt;19.1%(PTF)&gt; 8.9%(TF)</td>
</tr>
<tr>
<td>3</td>
<td>82.3%(OR)&gt;65.8%(UN)<em>&gt;64.1%(PTF)&gt;33.2%(TF)&gt;9.4%(PO)</em></td>
</tr>
<tr>
<td>5</td>
<td>92.4%(UN)<em>&gt;92.3%(PO)</em>&gt;81.5%(OR) &gt;77.4%(PTF)&gt; 77%(TF)</td>
</tr>
<tr>
<td>10</td>
<td>100%(PO)<em>&gt;88.7%(PTF) &gt;79.2%(OR)&gt;50%(UN)</em>&gt;7%(TF)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Query Length</th>
<th>Methods ordered by their performance wrt PosQueries measure after issuing 1000 queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6(OR)&gt;0.4(PO)&gt;0.4(UN)&gt;0.3(PTF)&gt;0.1(UN)</td>
</tr>
<tr>
<td>3</td>
<td>1.8(OR)&gt;0.4(PTF)&gt;0.2(PTF)&gt;0.1(PO)&gt;0.04(UN)</td>
</tr>
<tr>
<td>5</td>
<td>1.2(OR)&gt;0.53(PTF)&gt;0.14(P'TF)&gt;0.01(PO)&gt;0.01(UN)</td>
</tr>
<tr>
<td>10</td>
<td>0.02(OR)&gt;0.01(PTF)&gt;0.0(TF)&gt;0.0(PO)&gt;0.001(UN)</td>
</tr>
</tbody>
</table>

Table 2. Comparison of different term selection methods for query length varied from 1 to 10.

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 is given in Table 2. We report results after retrieving 3000 documents and for 1000 issued queries, unless the result is marked by *, where the values are given 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 are very precise (high percentage of documents returned in the target class) but most of the times do not return any documents (especially for longer queries).

We found that each method performs best within 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, tfidf and uniform term selection methods.
Fig. 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) Odd-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 (e.g., length 10) are much more likely to return no documents at all, and so can be costly. (b) Odd-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.
The figure shows the performance of different query reformulation techniques.

(a) For a single query and a single term, the performance is represented with different query term lengths.
(b) For multiple queries and a single term, the performance is represented with different query term lengths.
(c) For a single query and multiple terms, the performance is represented with different query term lengths.

Each graph shows the number of documents found versus the number of queries issued.

- **TF1** (Term Frequency 1)
- **TF3**
- **TF5**
- **TF7**
- **TF10**

**Slovenian Documents Found**

**Total Queries Issued**

**Total Documents Retrieved**

- **PTF1** (Probabilistic Term Frequency 1)
- **PTF3**
- **PTF5**
- **PTF7**
- **PTF10**

**Slovenian Documents Found**

**Total Queries Issued**

**Total Documents Retrieved**

- **OR1** (Relevance of Terms)
- **OR3**
- **OR5**
- **OR7**
- **OR10**

**Slovenian Documents Found**

**Total Queries Issued**

**Total Documents Retrieved**
Higher number of queries exceed term indices by a very small number of the larger language documents that use a very high and (c) is not found more than 1 inside and 1 outside.

Figure 4. Comparison of performance for the same term motion methods using different threads.

(a) RFTIDF1

(b) RFTIDF3

(c) RFTIDF5

(d) RFTIDF10

(e) UN1

(f) UN3

(g) UN5

(h) UN7

(i) UN10

Building Minority Language Corpora by Learning to Categorize Web Search Queries
3.2 Learning Methods

can vary
documented with different vocabulary coverage, the ideal measurement statistic
whether our goal is to achieve as many documents as possible, a fixed number, or
yearly short-term performance may do better on the short term. Depending on
the mechanism. However, in our evaluation which stops more than searching for a
method which quickly finds a reasonable shortest mechanism can then explore
method. In this way, there is a trade-off between expectation and exploration.
Our general goal is to find a good alternative mechanism in the shorter time

during evaluation of any learning (described in Table 2).

- Y-Recall-Intersection in our learning evaluation since it performed poorly in
  learning evaluation. These results are due to the problem discussed in our
  proposed algorithms, some fundamentals, and learning algorithms. Those of directly

are incorporated some fundamentals in our learning algorithms. Instead of directly

so far, these results have been experienced, our system may explore different
the short-term models should learn the short-term models or a gain

5.1 Learning Overview

also report experimental results,
time. We present a family of learning algorithms in the next section and
methods, as do some of the earlier models. In the latter part of the earlier models, the

discussed in this section. The short-term models, some of them, may explore different
components can be described by four parameters: Information, Term-Specific

5. Learning Query Parameters

as discussed in section 5.4, different methods excel with different query lengths.


\[(\text{score}(m)) = \begin{cases} \frac{\sum_{n \in D} \text{score}(m)}{|D|} & \text{if } m \neq \emptyset \\ 0 & \text{if } m = \emptyset \end{cases} \]


\[
\text{Next, in step } 2, \text{ the most recent method has probability } 1 \text{ of being selected.}
\]

\[
\text{and for an unsuccessful English or scoring method we use } \text{score}(m) = \text{score}(m) \text{ and for an unsuccessful German or scoring method we use } \text{score}(m) = \text{score}(m) \text{ and for an unsuccessful German or scoring method we use } \text{score}(m) = \text{score}(m).
\]

\[
\text{For the success and unsuccessful methods, we use different update mechanisms for the long-term memory.}
\]

\[
\text{The method estimates each method's probability of success based on the past performance of that particular memory method.}
\]

\[
\text{This was designed to perform a successful clustering method to continue as long as possible.}
\]

Building Dictionary Language Corpus, by learning to create Web Search Queries.
than the larger queries.

5.3. Results

In the experiments, to reduce the number of parameters to be learned, we fixed

\[ \text{a = 0.9} \]

" virtual parameter (the other parameters are independent and are fixed ahead).

For all experiments we set \( \alpha = 0.9 \) and \( \beta = 0.5 \) and include the scores for that experiment, and set \( \gamma = 0.5 \) and include the scores for that experiment. The scores, however, are not included in the next section.

The method used in the context of the experiment is a form of the Generalized Exponential Model (GEM) method. This method is used to predict the probability of a query being relevant to a document. The GEM method is based on the assumption that the relevance of a query is independent of the relevance of other queries.

The formula for the GEM method is:

\[
\text{GEM}(q) = \frac{1}{1 + \exp(-\theta q)}
\]

where \( \theta \) is a parameter that controls the sparsity of the model.

The results are shown in the following table:

<table>
<thead>
<tr>
<th>Query</th>
<th>Relevance Score</th>
<th>GEM Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>q2</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>q3</td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

As can be seen, the GEM method outperforms the baseline method in terms of the Kendall Tau correlation.
sense. For the other learning methods there is no such clear con-
trast between the test results. Inclusion and exclusion methods, which are based on the number of documents and terms, seem to be the best option. The number of documents and terms is generally lower than the number of terms in the first position. The results in Table 3 show that this holds true for all learning methods. The results in Table 4 show that this holds true for all learning methods.

Table 3. Comparison of parameter values of different learning methods. The parameter values were ordered by their performance.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Number of Terms</th>
<th>Parameter Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inclusion</td>
<td>6 - 8</td>
<td>A.I.A - A.I.A - A.O</td>
</tr>
<tr>
<td>Exclusion</td>
<td>6 - 8</td>
<td>A.I.A - A.I.A - A.O</td>
</tr>
</tbody>
</table>

We hypothesized that the on-the-fly learning learning best matches the performance of the other methods. In this case, the number of documents and terms is generally lower than the number of terms in the first position. The results in Table 4 show that this holds true for all learning methods. The results in Table 4 show that this holds true for all learning methods.

Table 4. Comparison of parameter values of different learning methods. The parameter values were ordered by their performance.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Number of Terms</th>
<th>Parameter Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inclusion</td>
<td>6 - 8</td>
<td>A.I.A - A.I.A - A.O</td>
</tr>
<tr>
<td>Exclusion</td>
<td>6 - 8</td>
<td>A.I.A - A.I.A - A.O</td>
</tr>
</tbody>
</table>
We compared different learning methods in terms of their performance. Our results indicate that different learning methods yield different outcomes. In Section 2.3, we present a comparative analysis of various learning methods.

![Comparison of Learning Methods](image-url)

In Figure 1, we compare the performance of different learning methods across several metrics. The x-axis represents the number of iterations, while the y-axis shows the probability of being chosen. Each line represents a different learning method, with varying colors for easy identification. The plots illustrate how each method performs under different conditions.

Figure 1 (a) shows the comparison for a specific dataset, while Figure 1 (b) displays the results for another dataset. These figures help us understand the effectiveness of each method in terms of accuracy and efficiency.

In summary, our experiments highlight the importance of choosing the right learning method for specific tasks. Further research is needed to explore the optimal conditions for each method.
with 


The results of running these experiments on different initial conditions are shown in Table 4. The properties of these initial documents are shown in Table 5. Slovenian Documents Found


6.2 Generalizations of Initial Conditions

In order to examine how our experiments generalize to other conditions, we examined the effects of varying the initial conditions by a range of doc-gen combinations of parameters (i.e., the number of unique words in the document).


Table 4: Description of the 3 different initial documents used in experiments - vocabulary

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Vocabulary Size</th>
<th>Total Documents Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>33</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>50</td>
<td>51</td>
</tr>
</tbody>
</table>

(a) In terms of number of documents, after 3000 documents LTM is again the best. (b) In terms of number of documents, after 3000 documents LTM is again the best. (c) The number of documents LTM is again the best. (d) The number of documents LTM is again the best.
Different initial conditions using LTMII
Other Languages

shown in Figure 8.

The words are ordered from native speakers' native languages to English stop-words in the initial language. We asked for the words needed for the first lexicalization. The words were selected for their frequency of occurrence in common with other lexicalizations. The words are similar to the words used in the current context. They are comparable, not known that some are shared with other languages.

Table 2: Words supplied by native speakers for two languages at the initial adjective level. The words are common to English stop-words in the initial language. We asked for the words needed for the first lexicalization.
In order to test the generalization power of our learning methods for different languages, we compared the results of a model trained on the web. The

6.3 Learning Parameters for Other Languages

words are more unique on the web than their counterparts in the TREC dataset. This is because the number of unique words is dependent on the size of the dataset, and the number of unique words in the TREC dataset is much larger than the number of unique words on the web. Further, the number of unique words on the web is much larger than the number of unique words in the TREC dataset.

Difficult

Chen et al.
Documents from Slovenian and Croatian are much more rare on the Web.

In our experiments, we tested three different methods for determining the number of relevant documents: (a) for each query and (b) for each query relevance function. The other


differences between the three methods and for the three relevance functions.

Fig. 10. Comparison of different methods with length 3 over four different languages (a) Slovenian, (b) Croatian, (c) Czech, and (d) Tagalog.
We show the number of documents retained for experiments using Cramér and
the update rule gives the best performance when on Cramér. The performance
of the long-term memory methods (GLP) in the W2M-520-52 output.

Table 2. A number of larger documents from 1000 queries, after 1-100 documents

<table>
<thead>
<tr>
<th>Method</th>
<th>LEX-M夸 LenJay does</th>
<th>1-100 queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>OH-3</td>
<td>Czech</td>
<td>680</td>
</tr>
<tr>
<td>OH-3</td>
<td>English</td>
<td>198</td>
</tr>
<tr>
<td>OH-3</td>
<td>Spanish</td>
<td>266</td>
</tr>
<tr>
<td>OH-3</td>
<td>Portuguese</td>
<td>16</td>
</tr>
<tr>
<td>OH-3</td>
<td>German</td>
<td>32</td>
</tr>
<tr>
<td>OH-3</td>
<td>Turkish</td>
<td>18</td>
</tr>
<tr>
<td>OH-3</td>
<td>Arabic</td>
<td>18</td>
</tr>
<tr>
<td>OH-3</td>
<td>Japanese</td>
<td>18</td>
</tr>
<tr>
<td>OH-3</td>
<td>Chinese</td>
<td>18</td>
</tr>
<tr>
<td>OH-3</td>
<td>Russian</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 6. Chisquare-based more important documents from both Cramér and
proposals.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of queries</th>
<th>Total 1000 documents retained</th>
</tr>
</thead>
<tbody>
<tr>
<td>OH-3</td>
<td>464</td>
<td>464</td>
</tr>
<tr>
<td>OH-3</td>
<td>477</td>
<td>477</td>
</tr>
<tr>
<td>OH-3</td>
<td>677</td>
<td>677</td>
</tr>
<tr>
<td>OH-3</td>
<td>835</td>
<td>835</td>
</tr>
<tr>
<td>OH-3</td>
<td>941</td>
<td>941</td>
</tr>
</tbody>
</table>

H. Chalm et al.
Figure 11. Comparison of learning methods on (a) Croatian and (b) Tagalog. For both languages, Long-Term Memory networks either use single (LTMI) or multiplicative (LTMII) update.
Another way of collecting a Schwarzschild core at the

not under the "Compton

imbry", 1 of the 3 000 Schwarzschild documents we found, 22% of them were

many 24% of them being classified as pure not in Schwarzschild by our classifier.

implied, we found approximately 1 000 pages under the "ligt-". In our experiments, we found approximately 1 000 pages outside the "Compton

are we pages outside the "Compton

of their core as shown in the good coverage of the pages under the "Compton

of their core as shown in the good coverage of the pages under the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton

are we pages outside the "Compton
7.2 Using Existing Corpus Construction Techniques

When building a model, the goal is to create a model that can accurately predict the sentiment of a document. Existing corpus construction techniques can be used to achieve this goal. These techniques involve using a pre-existing corpus of documents to train a model. The corpus is typically a large collection of text documents that have been labeled with sentiment scores. The model is then trained on this corpus to learn the relationships between the features of the documents and the sentiment scores.

The primary advantage of using existing corpus construction techniques is that they can be used to create a model that is more accurate and robust than a model that is trained on a smaller, less diverse corpus. This is because the model is trained on a larger and more diverse set of data, which helps it to better generalize to new, unseen data.

One popular technique for using existing corpus construction techniques is to use a pre-existing corpus to train a model and then fine-tune the model on a smaller, more specific corpus. This approach allows the model to learn from the larger corpus while still being able to adapt to the specific characteristics of the smaller corpus.

Another technique is to use a pre-existing corpus to train a model and then use transfer learning to adapt the model to a new task. This approach allows the model to leverage the knowledge it has learned from the pre-existing corpus, while still being able to adapt to the specific characteristics of the new task.

Overall, using existing corpus construction techniques can be a powerful way to create a model that is more accurate and robust than a model that is trained on a smaller, less diverse corpus. However, it is important to carefully evaluate the performance of the model on the target corpus to ensure that it is still able to accurately predict sentiment scores.
Figure 12. (a) Comparison between P-TF (P) and P-TF-1 (PTF) in terms of Precision-Recall, and (b) Comparison between P-OE (P) and P-OE-1 (OE) in terms of Precision-Recall.

Figure 13. Comparison with Previous Work for Tagalog documents Found.

Figure 14. Comparison with Previous Work for Slovenian Documents Found.

It is important to note that the performance of the models depends on the nature of the documents and the tasks of linking, retrieval, and ranking.

Over 600 documents in Tagalog were found overall; the trend remained the same with OR-3 filtering documents. While the second experiment was performed on Tagalog, the Slovenian Documents Found were due to the trend where the better performance of the models depends on the nature of the documents and the tasks of linking, retrieval, and ranking.
8. Conclusions and Future Work

We present an approach for automatically collecting web pages in a minority language and show that it performs well on several minority languages. Since this approach can be applied to any language, it opens up new avenues for research in the area of cross-lingual information retrieval. The approach is based on the idea that web pages in a minority language can be translated into a more common language and then processed using standard techniques.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>English</th>
<th>French</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>Indepedent</td>
<td>Indepedent</td>
<td>Indepedent</td>
</tr>
<tr>
<td>II</td>
<td>Dependent</td>
<td>Dependent</td>
<td>Dependent</td>
</tr>
</tbody>
</table>

In conclusion, our approach shows promise for automatically collecting web pages in minority languages. Further research is needed to improve the accuracy and coverage of the collected pages. Future work could also focus on developing more efficient methods for translating web pages from one language to another.

7.4 Query Length Paramaterization

Documents in Figure 13, for example, have found only 17
documents of Query 300 PPT-1 had found 20 and T-1 had found only 17
building blocks. However, these blocks turn out to be indepen-
Acknowledgements

Our approach for automatic query formulation is useful for any application for pattern recognition or developing a domain-specific search engine, as also our experiments have shown. In this work, we have experimented to augment existing domain-specific corpora in a different way. In order to continue the research and to use the one document at each experiment in this work, we need to perform experiments for other larger corpora. Our experiments on a single domain show that the engine performance using the experimental setup is not sufficient to provide a full control of the correlation, and other experiments are needed to further condense the content. The performance of the experiments was measured using the Leven-Farne algorithm, which partitions a successful experiment with higher precision. The experiments with a successful experiment using the Leven-Farne algorithm in this work are therefore a different explanation of the performance of the experiments, which have succeeded in the previous experiments. This was expected, given our experiments on a single domain. However, we found that our basic approach greatly contributes to the performance of the experiments, and we plan to further experiment with these experiments.

