Expanding Queries using Stems and Symbols

Michela Bacchin [*] Massimo Melucci

Department of Information Engineering
University of Padova
Italy

Abstract

This paper describes the experiments conducted in the ad-hoc retrieval task of the Genomic track at TREC 2004. Different query expansion techniques based on the addition of keyword stems and of genomic product symbols selected by relevance feedback were studied. Stemming was tested using a mutual reinforcement process for building a domain-specific stemmer. Relevance feedback was tested using a technique exploiting associations between symbols and related keywords.

1 Introduction

The Information Management Systems (IMS) research group of the Department of Information Engineering of the University of Padova participated in the ad-hoc retrieval task of the Genomic track. The aim of this first participation in the Genomic track was to explore this new domain-specific context using different query expansion techniques. On the one hand, a probabilistic stemming algorithm was used to expand queries by adding morphologically related words; on the other, a relevance feedback technique was designed to expand queries by selecting symbols added to queries together with their semantically related keywords and symbols – symbols are strings which describe genomic or proteic products.

In general, the employment of stemming is based on the basic idea that words which are similar in morphology are also similar in meaning. Hence the use of stemming algorithm allows to expand the query words with all their variants. Past experiments showed that, firstly, stemming could not improve retrieval performance because of the morphology of the documents language, and, secondly, that stemming is as more effective as the morphology of the language is more complex. In the context of documents about genomics, the dictionary, i.e. the set of keywords extracted from the documents, includes several words which are not present in a generic English dictionary, because they

[*] Contact author: `michela.bacchin@dei.unipd.it`

are composed by two or more domain-specific morphemes. Hence the number of variants seems to be greater than in a generic English collection, making the specific language morphology more complex. The hypothesis that the use of stemming in the Genomic track test collection could be more valuable than in a generic English collection has been investigated.

Symbols are strings of alphabetical, numerical and special characters naming genomic,proteic or biologic products. The language used in the test documents is characterized by the occurrence of several symbols. Symbols precisely and succintly describe document content without using keywords. Despite their precision, symbols can be affected by ambiguity – a genomic product might be identified by more than one symbol – thus making the problem of recall more difficult. It is well known in the IR literature that query expansion is an effective means to improve recall without decreasing precision in contexts where synonymy prevents the retrieval of relevant documents. At TREC our interest was in the investigation as to whether a relevance feedback-driven query expansion technique based on symbol detection can be effective.

Several experiments were conducted before and also after the release of the official runs. Hence it was decided to report in this paper all the significant experiments conducted even if only two runs was submitted for official TREC evaluation. The paper is organized as follows: In the next section, some issues of stemming in the context of genomic information retrieval are illustrated. In Section 3 the probabilistic model for stemmer generation is surveyed, whereas in Section 4 a relevance feedback technique based on the extraction and processing of symbols is presented. Section 5 describes the experimental setting and results.

2 Stemming in a Specific Domain Context

In the last years our research group proposed and developed a probabilistic model for stemmer generation [4]. The proposed stemming algorithm infers the word formation rules directly from the words of the corpus of documents and uses no prior linguistic knowledge on the language. Several experiments were carried out to evaluate if this probabilistic approach could be applied for building stemming algorithms for different languages and if the probabilistic stemming algorithm could be as effective as the ones manually built by linguistic experts.

The results observed were quite successfully with all the tested European languages, such as Italian, Spanish, French, German and English [1, 2, 3, 6]. For all the test languages the retrieval performances of the probabilistic stemming algorithm were similar to the ones obtained by linguistic stemming algorithms manually built. It was observed that for languages with a complex morphology, such as the Italian language, stemming could be useful for improving the retrieval performance of systems, especially if a researcher is interested in evaluating the precision after a few retrieved documents.

In TREC 2004 the interest was concentrated in evaluating if stemming could be valuable for improving retrieval performance in the context of documents and queries about Genomics. In this paper, it has been made the hypothesis that the morphology of this domain-specific language could be more complex than the one of the English common language. The dictionary of the MEDLINE collection is different from the one extracted from a collection which uses a common English language, such as the Wall Street Journal articles. There are many word compounds and words composed by two or more domain-specific morphemes. This fact could make the morphology of the MEDLINE language more complex than the common English morphology which is on the contrary quite simple and gives out a few variants for each word. The presence in the document collection of many variants decreases the probability of retrieving all the relevant documents because in every relevant document could be present a different variant which represents the same concept relevant to the one expressed in the query.

The complexity of the language of the Genomics track test collection would make the implementation of a stemmer difficult and subject to errors, and its effectiveness is not guaranteed even if it is built after an intellectual labor conducted by experts of the domain. Moreover, the language of the domain might evolve rapidly and the stemmer should be kept up-to-date. The research reported in this paper investigated whether an automatic and language-independent procedure to generate a stemmer for this domain could be a useful and effective exercise. In the next section, the design of such a procedure is illustrated.

3 A Probabilistic Model for Stemmer Generation

The probabilistic model proposed in [4] computes a list of pairs (word, stem) starting from the set of words extracted from a corpus of documents. It is based on a suffix stripping paradigm in which each word is split into a pair of substrings, called prefix and suffix, and considers the prefix as the stem. The model considers that words are the outcome of a generative process performed by a hypothetical machine that takes the set of all the possible prefixes and suffixes as input and produces words as output according to some type of linguistic knowledge and not at random. Because of this, the probability of generating a pair is not uniform – since the machine exploits some kind of linguistic knowledge, the probability that a stem is correctly concatenated with a derivation is higher than the probability that a generic prefix is concatenated with a generic suffix. Stemming can be seen as the inverse of this generative process: given a word, a stemmer has to guess the prefix and the suffix in order to form the most probable pair that the machine has chosen to generate the word. As the machine pools together its knowledge of the language, the most probable pair is formed by the stem and the derivation of the word.

3.1 The Algorithm

Given a finite collection W of words, let U be the set of N sub-strings generated after splitting each word $z \in W$ into all possible positions, except for those which generates empty sub-strings. If x, y are the prefix and the suffix of word z, respectively, then $z = xy$ and there are $n-1$ possible positions to which z is split, if $|z| = n$. Let us define the universe of the elementary random events as follows. Let $\Omega = \{(x, y) \in U \times U : \exists z \in W, z = xy\}$ be the set of all pairs (prefix, suffix) which can form any word and let $\Omega(z) = \{(x, y) \in \Omega : xy = z\}$ be the set of all of the pairs (prefix, suffix) leading to the same word z.

The stemmer has to infer the most probable pair of prefixes and suffixes $(x, y)^* = \omega^*$ chosen by the machine to generate the given word, computing the expression:

$$\begin{aligned}
\omega^* &= \arg \max_{\omega \in \Omega(z)} \Pr(\omega \mid z) && (1) \\
&= \arg \max_{\omega \in \Omega(z)} \frac{\Pr(z \mid \omega) \Pr(\omega)}{\Pr(z)} && (2) \\
&= \arg \max_{\omega \in \Omega(z)} \Pr(\omega) && (3) \\
&= \arg \max_{i=1,\ldots,n-1} \Pr(x_i, y_i) && (4)
\end{aligned}$$

where (2) is obtained applying the Bayes' rule, (3) is obtained observing that $\Pr(z \mid \omega) = 1$, since $\omega \in \Omega(z)$ yields to z only, and $\Pr(z)$ is the same for all ω and so it does not influence the maximization, and (4) is obtained because $\Omega(z) = \cup_{i=1}^{n-1} \{\omega_i\}$ and $\omega_i = (x_i, y_i)$.

Finally, the relationship which views that probability as the combination of one marginal probability and one conditional probability was exploited to compute $Pr(x_i, y_i)$, and hence:

$$\omega^* = \arg \max_{\omega \in \Omega(z)} \Pr(x) \Pr(y \mid x)$$

3.2 The Mutual Reinforcement in Stemming

To estimate the probability distribution of the pairs (prefix, suffix) $\Pr(x_i, y_i)$ which is necessary to find the most probable pair, the following notion of probabilistic mutual reinforcement in stemming was introduced:

> Stems are prefixes which have a high probability of being completed by derivations; derivations, in turn, are suffixes which have a high probability of completing stems.

If a collection of words is observed, a prefix is completed by diverse suffixes, and a suffix completes diverse prefixes. The mutual reinforcement relationship emphasizes that stems are more likely to be completed by derivations; derivations in turn are more likely to complete stems.

Let us formalize the notion of mutual reinforcement in stemming just introduced. It is a fact that:

$$\Pr(x_i) = \sum_{j=1}^{N} \Pr(x_i, y_j)$$

$$\Pr(y_j) = \sum_{i=1}^{N} \Pr(x_i, y_j)$$

and

$$\Pr(x_i, y_j) = \Pr(y_j \mid x_i) \Pr(x_i)$$
$$\Pr(x_i, y_j) = \Pr(x_i \mid y_j) \Pr(y_j)$$

Thus the mutual reinforcement relationship between the stems and the derivations can be written as:

$$\begin{aligned} \Pr(x_i) &= \sum_{j=1}^{N} \Pr(x_i \mid y_j) \Pr(y_j) \quad i = 1, \ldots, N \\ \Pr(y_j) &= \sum_{i=1}^{N} \Pr(y_j \mid x_i) \Pr(x_i) \quad j = 1, \ldots, N \end{aligned} \tag{5}$$

Using a more compact notation,

$$\mathbf{p} = [\Pr(x_1) \cdots \Pr(x_N)]'$$

as the vector of the prefix probabilities, and

$$\mathbf{s} = [\Pr(y_1) \cdots \Pr(y_N)]'$$

as the vector of the suffix probabilities. Moreover, let $\mathbf{A} = [a_{sr}]$ be the $N \times N$ matrix such that $a_{sr} = \Pr(x_r \mid y_s)$, and let $\mathbf{B} = [b_{rs}]$ be the $N \times N$ matrix such that $b_{rs} = \Pr(y_s \mid x_r)$. Therefore,

$$\mathbf{p} = \mathbf{A}'\mathbf{s}$$

and

$$\mathbf{s} = \mathbf{B}'\mathbf{p}$$

After substituting,

$$\mathbf{p} = \mathbf{A}'\mathbf{B}'\mathbf{p}$$

and

$$\mathbf{s} = \mathbf{B}'\mathbf{A}'\mathbf{s}$$

and then $\mathbf{p}$ is the eigenvector of $\mathbf{C} = (\mathbf{BA})'$ associated to unity eigenvalue, and $\mathbf{s}$ is the eigenvector of $\mathbf{D} = (\mathbf{AB})'$ associated to unity eigenvalue. Hence, the probability $\Pr(x_i)$ can estimated by the component of the eigenvector $\mathbf{p}$ associated with the prefix x_i, and the probability $\Pr(y_i)$ by the component of the eigenvector $\mathbf{s}$ associated with the suffix y_i. To compute the eigenvectors an iterative algorithm can be applied; the details of our iterative algorithm can be found in [4].

4 Query Expansion using Symbols

The symbols, i.e. strings of alphabetical, numerical and special characters used to name genomic or proteic products, could be useful to make retrieval results more precise. Actually, symbols encapsulate much information about what the end user aims at searching since these strings are often acronyms or nicknames of longer descriptions of the products about which the documents are relevant or not.

Despite their potential usefulness in improving retrieval performance, symbols are also ambiguous since a genomic or proteic product can be labelled using more than one symbol in different research papers of the database. As a consequence, a query can label a product using a symbol which mismatches the ones used in relevant documents.

As it is well known in the IR literature, query expansion helps to address the problem of word ambiguity. Therefore query expansion could be applied to symbols as it was done for keywords. However, it is necessary to add semantics to symbols so that they can be employed in a query expansion technique. If semantics is added to symbols, these can be associated one to each other and these associations can be exploited to expand a query including symbols.

To add semantics to a symbol, it could be possible to exploit the information given by ontologies or similar databases. However, ontologies or similar databases often cover a small portion of a document collection. Alternatively, it could be possible to exploit the information encoded into the documents content and compute the similarities among all the symbols in the collection. However, such an approach poses significant problems since a vector-based representation of the symbols and a similarity function should be defined.

In order to search for a source of evidence which adds semantics to symbols, the following hypothesis was drawn: Symbols occur in the document texts closely to keywords which give a semantics to the symbols. This closeness can occur if, for instance, a symbol is introduced just before or after the words giving its definition or function. Another hypothesis was drawn: Symbols occur closely to keywords to which related symbols occur as well. The latter is the association between symbols being searched. The algorithm to discover the associations among symbols has been based on the notion of mutual reinforcement which has been used to design the stemming algorithm presented and tested in this paper. The notion of mutual reinforcement for query expansion can be stated as follows:

> A symbol used to expand the query co-occurs frequently with and closely to keywords used to expand the query, and a keyword used to expand the query co-occurs frequently with and closely to the symbols used to expand the query.

4.1 Symbol Recognition

To recognize symbols, the following regular expressions were defined:

```
[A-Z][A-Z0-9]*[-/][a-zA-Z0-9]+
[a-zA-Z][A-Z0-9]*'('[a-zA-Z0-9]+')'[a-zA-Z0-9]*
[a-zA-Z][A-Z0-9][a-zA-Z0-9]+
```

These expressions defined a symbol as a string containing uppercase letters, digits or some special characters. Some normalization was done to make special characters little influential. The characters `-()/` were removed from the strings matching one of the first two regular expressions.

4.2 The Blind Relevance Feedback Algorithm

Given a collection of documents D and a full-text query q, a list of documents was retrieved and the m top-ranked ones were selected to start up the query expansion algorithm. Let M be set of these selected documents. The assumption of implicit relevance feedback was made, i.e. the list of document contains many relevant items and the synonyms occur in the retrieved relevant items. From M the algorithm extracted all the tokens labelling symbols or keywords – the distinction between symbols and keywords was made using the regular expressions above introduced.

Query expansion can be computed choosing to expand the query only with tokens labelled as symbols, or with those labelled as keywords. Once the token type had been chosen, the k top-ranked tokens were extracted from M. The tokens extracted from the top-ranked documents were ranked by a TF·IDF measure, in which TF represents the token frequency in M, while IDF represents the inverse document frequency computed considering all the documents in the collection D. This way, query q was expanded by adding the tokens which were the most specific ones for the top-ranked documents M.

4.3 The Exploitation of the Mutual Reinforcement

The algorithm explained above used the TF·IDF measure to select the k top-ranked tokens to be added to the query. In order to investigate if the notion of mutual reinforcement could be valuable also in the query expansion context, the algorithm was modified to compute also a different token ranking based on the mutual reinforcement relationship between symbols and keywords. The algorithm built a graph whose nodes were symbols and the keywords occurring closely to those symbols. If a symbol occurred within a 10-word text window, then all the window words were extracted and associated to that symbol as co-occurring words. Hence an edge was added from the symbols and the nodes which represented the keywords associated. Then an algorithm exploiting the mutual reinforcement relationship between symbols (hubs) and keywords (authorities) was performed to associate a weight to each symbol and keyword. These weights represented a measure of the degree to which each token (symbol or keyword) is associated to the symbols of the list of top-ranked documents. A symbol (keyword) with the highest score is the "best" one. The hypothesis was that two synonymous symbols tend to be associated to the same keywords, and that the synonyms being found can be effectively used to improve the performance if the keywords come out of relevant documents. The query is then expanded using the top-ranked symbols or keywords. Distinct experiments were carried out both using the best symbols and the best keywords.

5 Experimental Setting

System. To carry out the experiments for the ad-hoc task of the genomic track, a PC equipped with RedHat Linux, a 800 MHz Intel Pentium CPU and 512MB RAM was used. The MySQL AB relational database systems was employed for storing, indexing and searching documents, and a suite of tools were developed in Java and C++ for pre- and post-processing. MySQL is one of the most popular open source database server. It has full-text indexing and searching capabilities, based on a space-vector model [8].

Indexing. Before storing the documents into the document table in MySQL a conversion was necessary. Because of the limit of 4GB for each table in MySQL, all the sections of a generic document could not be stored, and `(PMI, TI, AB)`, i.e., Document-Id, Title and Abstract were only stored. The original 9GB document collection was reduced to a 3GB MySQL document table. All the 599 stop-words have been removed before storing the documents. The MySQL full-text capabilities were exploited to build an index consisting of both Title and Abstract fields.

Searching. To retrieve the relevant documents a database application was developed to establish a connection to the MySQL database, send the queries and produce a list of relevant documents in a format compatible with `trec_eval` [5]. The queries have been built using the MySQL full-text extension to standard SQL:

```
SELECT pmid, match(TI,AB) AGAINST(Query_Text) AS score
FROM Table_name WHERE
match(TI,AB) AGAINST(Query_Text) LIMIT 1000
```

To build the query string *Query_Text*, all the words except stop-words appearing in `Title` and `Need` sections of the topics provided by NIST were used.

6 Experimental Results

Several runs were performed to test, on the one hand, the effectiveness of query expansion by stemming and, on the other hand, the effectiveness of query expansion algorithm presented in Section 4. Each run was characterized by the use of a different algorithm and the run labels and a brief description are reported in Table 1.

Run Labels	Description
`PDnoStem`	No stemming nor query expansion
`PDTDmp4`	Stemming using the probabilistic algorithm
`PDporter`	Stemming using Porter's algorithm for English
`PDnm`X`k`Y`f`F`r`Z`y`W	Query expansion using `m` $= X$, `k` $= Y$, `f` $= F$, `r` $= Z$, `y` $= W$ as parameter values
Parameters	**Description**
`m`	number of top-ranked documents: $\lvert M\rvert$
`k`	number of top-ranked tokens to be added to q
`f`	token type choice: 1=symbols, 2=keywords
`r`	number of iterations of mutual reinforcing computation
`y`	number of tokens extracted to build the graph

Table 1: Run labels and descriptions.

6.1 Experimental Results about Stemming

Since MySQL does not use any stemming algorithm, stemming was a pre-process and a different document table was stored into the database for each different stemming algorithm applied to the original documents. As reported in Table 2, the size of the table which store the original documents was obviously much greater than the size of the stemmed tables. In particular, the table which stored the documents stemmed by the probabilistic stemmer is about 75% of the original one thus confirming that stemming could be useful to decrease table and index sizes. The same stemming pre-process was also performed for

Stemming algorithm	Size in MB	Unique terms
`PDnoStem`	3,330	706,523
`PDTNmp4`	2,527	324,234
`PDporter`	2,824	577,553

Table 2: Size of tables in the database.

the topics, so three set of 50 queries were created – one set for each stemming algorithm applied. Once the topics and the documents were stemmed by the same stemming algorithm, the searches were carried out using the database application described in the previous Section.

Building the stemming algorithm. Starting from the complete MEDLINE collection, a set of 706,523 words with frequency greater than one were selected out of the 1,200,090 unique words extracted from the collection. The probabilistic stemmer generation was performed from this restricted, yet large set of words thus obtaining a list of pairs (word, stem).

By default, MySQL does not index words which have a length minor than four, hence the stemming algorithm was modified in order to stem only words whose length was at least five – a rule such that the stem length has to be at least four was added. Moreover, several words which should not have been stemmed because they were already roots were observed in the past experiments. At TREC-13 the stemming algorithm was modified in a way that algorithm does not split the words which have a high probability of being root forms – the complete words is treated as a stem. As result, 113,212, out of 706,523 words has been not stemmed because 89,394 presented a word length minor than five, and 23,818 were already stems with high probability.

Results. Table 3 reports the traditional effectiveness measures: the number of relevant documents which had been retrieved (Rel-Retr), Average Precision (A-P), R-Precision (R-P) and the Precision computed at 5 (P@5) and 10 (P@10) document cut off values. The `PDTNmp4` run was one of the two submitted official runs. The results seem confirming the intuition that stemming could be useful

RunID	Rel-Retr	A-P	R-P	P@5	P@10
`PDnoStem`	2985	0.2015	0.2384	0.4880	0.4300
`PDTNmp4`	3102	0.2074	0.2476	0.5120	**0.4560**
`PDporter`	**3248**	0.2024	0.2457	0.5160	0.4320

Table 3: Some effectiveness measures for the experiments done.

in this domain-specific context, even if the improvement is little for A-P and R-P. Yet the precision computed at the 5-document cut off value shows a greater improvement for the runs which stemming was applied in. It is remarkable that `PDporter` reports the largest improvement of recall, whereas `PDTNmp4` is comparable in terms of precision.

6.2 Experimental Results about Relevance Feedback using Symbols

The experiments were designed in a way that it could be possible to isolate each single algorithm feature, which could influence the performances of the query expansion algorithm proposed in this paper. First of all, a set of runs were carried out with different quantities of top-ranked documents used as the source from which relevance information is extracted – `m` is the number of top ranked documents from which the token to be added to the query are extracted – and with different quantities of top-ranked tokens added to the initial query – `k` is the maximum number of tokens to be added to the query. It should be recalled that the extracted tokens are ranked by TF·IDF (see Section 4.2).

For each value of `m,k` two runs were performed: one run was performed by adding symbols to queries, one run was performed by adding keywords. In this first set of runs, the mutual reinforcement among symbols and keywords was not used. A run was also done using the query expansion feature of the embedded full-text engine of MySQL with the aim of comparing our algorithm, which distinguishes symbols from keywords, to another blind relevance feedback-based query expansion algorithm which does not distinguish symbols from keywords.

To implement implicit relevant feedback, it was decided to use only the top 2 and 5 top-ranked document, i.e. `m` $= 2$ and `m` $= 5$ was used, and to extract the top `k` $= 5$ and `k` $= 20$ tokens from M.

Table 4 reports the usual effectiveness figures for the first set of runs.

RunID	Rel-Retr	A-P	R-P	P@5	P@10
`PDnoStem`	2985	0.2015	0.2384	0.4880	0.4300
`PDnm2k5f1`	3014	0.1967	0.2331	0.4480	0.4000
`PDnm2k5f2`	3002	0.1902	0.2304	0.4600	0.4060
`PDnm5k20f1`	2968	0.1904	0.2224	0.4600	0.4100
`PDnm5k20f2`	**3035**	0.1918	0.2263	**0.5040**	0.4400
`PDmysqlqe`	2745	0.2086	0.2397	0.4840	0.4300

Table 4: Effectiveness measures for the query expansions.

Quite surprisingly, a constant decrease of all the effectiveness figures was observed. In effect, the precision computed at five documents showed a decrease of the baseline performance. The most probable reason is that the baseline could not provide enough relevant documents within the top `m`. Also the MySQL query expansion run labelled with `PDmysqlqe` did not obtain an effectiveness improvement.

An exception was the run labelled with `PDnm5k20f2` which reported an increase of the total number of retrieved relevant documents and of the precision at five documents retrieved.

To test if this average decrease could be imputable to the lack of enough relevant documents among the top `m` retrieved, a set of runs was performed to test *explicit* relevant feedback. The top ranked `m` relevant documents retrieved among the top `j` retrieved documents were identified. The symbols and the keywords were extracted from this subset of documents to be added to the initial query. Table 5 reports the figures for the second set of runs. The runs based on

RunID	Rel-Retr	A-P	R-P	P@5	P@10
`PDnoStem`	2985	0.2015	0.2384	0.4880	0.4300
`PDnm5k20f1j25`	2992	0.2014	0.2277	0.5440	0.4500
`PDnm5k20f2j25`	3028	0.2082	0.2295	0.5640	0.4640
`PDnm5k100f1j1000`	3003	0.2104	0.2417	0.6000	0.4920
`PDnm5k100f2j1000`	3038	0.2130	0.2405	0.6520	0.5220

Table 5: Effectiveness measures for explicit relevance feedback.

a known set of relevant documents seem to perform better than the same runs based on blind relevance feedback. In particular, the improvement of the precision at five documents is statistically significant with a p-value < 0.002 for the Wilcoxon test and for all the runs. Hence, it seems that the explicit knowledge of some relevant documents is a necessary condition to significantly improve the performance of the relevance feedback technique based on the distinction between symbols and keywords.

In order to test if the mutual reinforcement could be useful to improve the ranking of the top-ranked tokens used to expand the query, a third set of runs was performed. These runs were labelled using $\mathbf{r} = R$ as the number of iterations performed by the algorithm implementing the mutual reinforcement between symbols and keywords. As reported in Table 6 the mutual reinforcement in this context is not clearly useful since precision does not improve consistently. Yet the total number of retrieved relevant documents increases as reported by `PDnm5k100f1j1000r10y10` and `PDnm5k20f1j25r10y10` which are two runs that employ symbols.

RunID	Rel-Retr	A-P	R-P	P@5	P@10
`PDnoStem`	2985	0.2015	0.2384	0.4880	0.4300
`PDnm5k100f2j1000r10y10`	2986	0.2015	0.2384	0.4880	0.4300
`PDnm5k100f1j1000r10y10`	**3134**	0.2125	0.2388	0.5680	0.4720
`PDnm5k20f1j25r10y10`	**3120**	0.2063	0.2316	0.5160	0.4400

Table 6: Effectiveness measures for explicit relevance feedback and mutual reinforcement.

However, `j` = 1000 is a rather unrealistic assumption in real operational settings and the improvement reported by the runs based on such a high value should be considered carefully. On the contrary, it is worth noting that an appreciable improvement can be observed if `j` = 25 and mutual reinforcement is performed using `r` = 10.

7 Conclusions and Future Work

Our interest has been in the investigation of different query expansion algorithms. First stemming was studied to test if it could be useful in this domain-specific context, and the results were quite positive because all the runs which employ the stemming process obtained performances which were superior to the baseline.

Then a set of relevance feedback techniques were tested on the basis of the idea that symbols could be crucial for this domain specific context. The results have been rather inconclusive – the query expansion algorithm seems to really improve the performances of the systems only if the start up document set consisted of relevant documents and keywords are the tokens used to expand the query. If the query expansion algorithm were based on blind relevance feedback or symbols are used instead, performance decreases. This might be due to the fact that in the first `k` documents there are several not relevant documents or symbols do not sufficiently discriminate relevant documents from non-relevant documents.

However, there are some positive signals which suggest to continue the study of the symbols role in this domain-specific context. An analysis is planned to compare the symbols statistical distribution among relevant and non-relevant documents. Other activities are also planned to improve and adapt the probabilistic stemmer model to this domain – it may be hypothesised that if the model could better infer the word formation rules specific for this context, it could improve the performances of the retrieval system.

References

[1] M. Agosti, M. Bacchin, N. Ferro, and M. Melucci. Improving the Automatic Retrieval of Text Documents. In C. Peters, M. Braschler, J.Gonzalo, M. Kluck (eds.) *Advances in Cross-Language Information Retrieval, Third Workshop of the Cross-Language Evaluation Forum, CLEF 2002. Rome, Italy, September 19-20, 2002. Revised Papers*, pages 279–290. Lecture Notes in Computer Science (LNCS) 2785, Springer-Verlag, Germany, 2003.

[2] M. Bacchin, N. Ferro, and M. Melucci. The Effectiveness of a Graph-based Algorithm for Stemming. In E. P. Lim, S. Foo, C. S. G. Khoo, H. Chen, E. A. Fox, S. R. Urs, and C. Thanos (eds.) *Digital Libraries: People, Knowledge, and Technology. Proceedings of 5th International Conference on Asian Digital Libraries (ICADL 2002)*, Singapore, December 11-14, pages 117–128. Lecture Notes in Computer Science (LNCS) 2555, Springer-Verlag, Germany, 2002.

[3] M. Bacchin. A Language-Independent Stemming Algorithm. *Ph.D. Thesis*, Department of Information Engineering, University of Padua, Italy, 2002.

[4] M. Bacchin and N. Ferro and M. Melucci. A Probabilistic Model for Stemmer Generation. *Information Processing & Management*, 41(1): pages 121–137, Elsevier, 2005.

[5] C. Buckley *et al.* The `trec_eval` Evaluation Package. `ftp://ftp.cs.cornell.edu/pub/smart/`, 2004. Visited on October, 2004.

[6] G. Di Nunzio, N. Ferro, M. Melucci, and N. Orio. Experiments to Evaluate Probabilistic Models for Automatic Stemmer Generation and Query Word Translation. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck (eds.) *Evaluation of Cross-Language Information Retrieval Systems, Fourth Workshop of the Cross-Language Evaluation Forum, CLEF 2003.* Trondheim, Norway, August 21-22, 2003. Revised Papers. Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, 2004.

[7] D. Harman. How Effective is Suffixing? *Journal of the American Society for Information Science*, 42(1):7–15, Wiley, 1991.

[8] MySQL AB. MySQL Homepage. `http://www.mysql.com`. Visited on October 2004.

[9] M.F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980. Reprinted in K. Sparck Jones, and P. Willet, *Readings in Information Retrieval*, Morgan Kaufmann, 1997,

[10] The English Stoplist of the SMART System. `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`. Visited on October, 2004.