The University of Padua at CLEF 2004: Experiments on Statistical Approaches to Compensate for Limited Linguistic Resources

G.M. Di Nunzio N. Ferro N. Orio

University of Padua – Department of Information Engineering
{dinunzio,nf76,orio}@dei.unipd.it

1 Introduction

This year 2004 is the third participation of the Information Management Systems (IMS) research group of the Department of Information Engineering of the University of Padua to the CLEF campaign. The participation to the previous CLEF 2002 and 2003 campaigns gave us the opportunity to enforce and enrich our statistical approach to stemming, and gave us the possibility to underline the evidence that our language independent stemmer generator could reach the performance of the state-of-the-art language dependent Porter's stemmers.

This year we participated again in the mono-lingual track using a Hidden Markov Model approach for the stemmer generation. It was our main purpose to engage the problem of managing two complete unknown and for us non-understandable languages like Finnish and Russian. Given the experience of the previous year, we participated also in the bi-lingual (German-to-French) track to apply a new solution for the problem of query expansion/translation from a language to another, especially when linguistic resources are low.

2 Monolingual Track Experiments

The main goal of monolingual experiments has been the development of methodologies and techniques that do not require, or minimize, the human labor when applying information retrieval (IR) techniques to new languages. To this end, languages as Finnish and Russian are particularly suitable because they are very different from languages known by the components of our research group – i.e. Italian, French, and obviously English. French can be considered as a reference language for comparing the system performances.

We focus our attention on the development and test of stemming algorithms, which is the component of an information retrieval system (IRS) that is more related to the structure of a given language. With the goal of minimizing manual work, we carried on also for this evaluation campaign the development of a set of stemmers based on Hidden Markov Models (HMMs). According to our approach, which is explained in the next sections, HMMs do not require any previous knowledge about the morphology of the language to be stemmed and can be trained simply using a set of words automatically extracted from the test collection.

2.1 Hidden Markov Models to Generate Words

Hidden Markov Models (HMMs) are finite-state automata where transitions between states are ruled by probability functions [5]. At each transition, the new state emits a symbol with a given probability. HMMs are called *hidden* because states cannot be directly observed, what is observed are only the symbols they emit. The parameters that completely define an HMM are, for each

state: the probabilities of being the initial and the final state, the transition probabilities to any other state, and the probability of emitting a given symbol.

HMMs are particularly suitable to model processes that are unknown but can be observed through a sequence of symbols. For instance, the sequence of letters that forms a word in a given language can be considered as a sequence of symbols emitted by a HMM. The HMM starts in an initial state and performs a sequence of transitions between states emitting a new letter at each transition, until it stops in a final state. In general, more state sequences, or *paths*, can correspond to a single word. It is possible to compute the probability of each path, and hence to compute the most probable path corresponding to a word. This problem is addressed as *decoding* and normally solved using the Viterbi algorithm.

In order to apply HMMs to the stemming problem, a sequence of letters that forms a word can be considered the result of a concatenation of two subsequences: a prefix and a suffix. A way to model this process is through a HMM where states are divided in two disjoint sets: states in the *stem-set* generate the first part of the word and states in the *suffix-set* generate the last part, if the word has a suffix. For many Indo-european languages, there are some assumptions that can be made on the model:

- Initial states belong only to the stem-set; i.e. a word always starts with a stem.
- Transitions from states of the suffix-set to states of the stem-set have always a null probability; i.e. a word can be only a concatenation of a stem and a suffix.
- Final states belong to both sets; i.e. a stem can have a number of different derivations, but it may also have no suffix.

As it can be seen, these assumptions are quite general. We supposed that they could apply, at least to a certain extent, also for Finnish and Russian, while we know that are valid for French. A general HMM topology that fulfills these conditions is depicted in Figure 1.

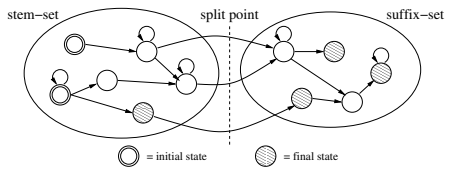

Figure 1: HMM topology with highlighted the stem-set and the suffix-set.

Once a complete HMM is available for a given language, stemming can be carried out straightforwardly considering a word as a sequence of symbols emitted by the HMM. As a first step, the most probable path that corresponds to the observed word is computed using decoding. Then the analysis of this path highlights the transition from a state of the stem-set to a state of the suffix-set, we call this transition the *split-point*. If there is no split-point then the word has no suffix, otherwise the sequence of letters observed before the split-point is taken as the stem and the one observed after is taken as the suffix.

2.2 Training the HMM

The proposed topology defines the number of states, their labels indicating the sets to which they belong, the initial and final states, and the allowable transitions. Yet all the probability functions that constitute the HMM parameters need to be computed. The computation of these parameters is normally achieved through *training*, which is based on the Expectation-Maximization (EM) [5] algorithm. It is important to stress that our goal is to develop fully automatic stemmers that do not require previous manual work. This means that we consider that neither a formalization of morphological rules nor a training set of manually stemmed words are available.

We propose to perform an unsupervised training of the HMM using only a sample of the words of the considered language. The training set can be taken at random by documents that are available at indexing time. It can be noted that an unsupervised training does not guarantee that the split-point of the most probable path has a direct relationship with the stem and the suffix of a given word. With the aim of creating such a relationship, we propose to inject some more knowledge about the general rules for the creations of word inflections. Thus, we make the reasonable assumption that, for each language, the number of different suffixes is limited compared to the number of different stems. Suffixes are a set of letter sequences that can be modeled by chains of states of the HMM. This assumption suggests a particular topology for the states in the suffix-set, which can be made by a number of state chains with different lengths, where: transitions from the stem-set are allowed only to the first state of each chain; the transition from one state to the next one has probability one; each chain terminates with a final state. The maximum length of state chains gives the maximum length of a possible suffix. Analogously, also the stem-set topology can be modeled by a number of state chains, with the difference that a state can have non-zero self-transition probability. The minimum length of a chain gives the minimum length of a stem. Some examples of topologies for the suffix-set are depicted in Figure 2, where the maximum length of a suffix is set to four letters, and the minimum length of a stem is set to three letters.

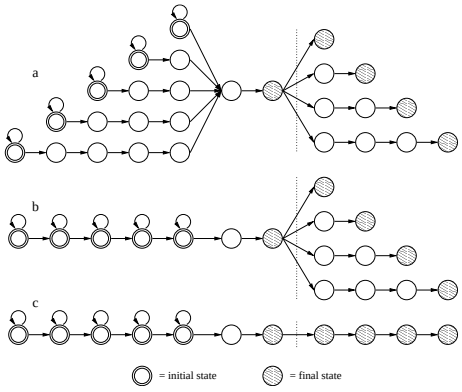

Figure 2: Three topologies of the HMM that have been tested.

After the redefinition of the suffix-set topology, the HMM can be trained using the EM algorithm on a set of words. Given the previous assumption, it is likely that a sequence of letters that corresponds to a suffix will be frequently present in the training set. For this reason, the EM algorithm will give to the states in the suffix-set a higher probability of emitting the letters of frequent suffixes. For example, considering the suffix-set chains, the state in the one-state chain will emit with the highest probability the last letter of each word, the states in the two-states chain will respectively emit the most frequent couple of ending letters of each word, and so on. Once the model has been trained, for each word the corresponding path that terminates with the most frequent sequence of letters it is expected to have an high probability of being selected as the most probable path, giving a correct stemming of the word.

2.2.1 The STON Algorithm

We developed an algorithm, named STON, to test the methodology and the changes on retrieval effectiveness depending on some of its parameters. STON receives as input a sequence of letters corresponding to a word and gives as output the position of the split-point. As explained in the previous sections, STON needs to be trained off-line using a subset of the words of the collection. Stemming can then be carried out on-line for any new word, also for the ones that are not present in the training set. The algorithm can be divided in three main steps:

1. *Training/off-line*: given a set of words $w \in W_L$, taken from a collection of documents written in a language L, and a HMM with parameters λ which define the number of states and the set

of allowable transitions, STON computes through the Expectation-Maximization algorithm:

$$\lambda_L^* = \arg\max_{\lambda} \prod_{w \in W_L} Pr(w \mid \lambda)$$

This step needs to be carried out only once for each language, eventually using a sample of the words of a document collection.

2. *Decoding/on-line*: given a word w_L, written in language L, and a trained model λ_L, STON computes the most probable path q across the states corresponding to w_L by Viterbi decoding:

$$q^* = \arg\max_{q} Pr(q \mid w_L, \lambda_L)$$

3. *Stemming/on-line*: once the most probable path q^* for word w_L is computed using model λ_L, the split-point can be computed by a simple inspection of the path, that is when the state sequence enters the suffix-set.

3 Bilingual Track Experiments

Bilingual experiments have been carried out with the main goal of testing the effectiveness of an IRS when advanced tools for query or document translations are not exploited. This situation applies each time an existing IRS has to be extended to a set of new languages, when the cost cannot be afforded for reliable translators for each language couple – or for each language from/to a single language (e.g. English). Moreover, it may the case that advanced translators are completely unavailable from/to languages that are spoken by a reduced amount of the world population – or that are spoken in countries where the economic and technological growth are still slow. Yet we argue that an IRS should have reasonable performances also when linguistic resources are minimal.

In our approach we considered only simple word-by-word translations, such as the ones provided by most of the free translation services on the Web, as tools for the bilingual experiments in the evaluation campaign. As it is well known, word-by-word translations have a number of drawbacks, mainly due to the absence of a context for word disambiguation and for dealing with synonyms and antonyms. Clearly these drawbacks can have a negative impact on the performances of a bilingual IRS. In order to obtain a base translator, we used the Web translation service offered by Google[1], using the translation of single words to not take advantage from the possible use of linguistic cues by the Google translator. It can be noted that this choice gave an additional constraint, that is there was no control on the size of the vocabulary.

The methodology that we propose to partially overcome the problems arising from a simple word-to-word translator is based on the use of two collections, the one in the language of the topic and the one in the language of the relevant documents. The source collection is used to expand the query terms that are to be translated. The methodology is presented in the following sections.

3.1 Almost Comparable Corpora

There has been extensive research on the combination of documents collections written in different languages for bilingual information retrieval. Usually the assumption is that two collections of documents, written in two different languages, allow the coupling of documents. The two collections are normally referred as *parallel corpora* [4] when they can be exactly aligned. This may be the case of transcriptions of legal documents for bilingual countries such as Canada, or when one collection is the translation (made by human experts) of the other one. The two collections are normally referred as *comparable corpora* [6] when documents are independently written in the two languages, but it is possible to couple a subset of the documents, which have the same topic. The coupling can be done using external information such as metadata. An example of this situation is

[1] The service is reachable from the Google homepage, at `http://www.google.com/`

a bi- or multi-lingual newspaper, where news articles that are potentially of interest for the whole population are independently written in all of the languages, while more local news are reported in articles written in only one language; the GIRT collection, which is available for this evaluation campaign, can be considered an example of comparable corpora.

Unfortunately, parallel and/or comparable corpora are not available for all language couples. Yet there are a number of collections written in different languages that, though not really comparable, have documents that share similar subjects. As an example, we can considered the collections of news articles used for the CLEF campaign. Newspapers in Europe are independently written, yet there are a number of events, for instance of political, social, or economical nature, that are of interest for most of the European citizens. These events are likely to be the subject of a set of news articles in all of the different newspapers. Clearly, the number of news articles that each newspaper dedicates to a given subject may dramatically vary, depending on political choices, on the locality of the event, and so on. Also the time span in which a given subject is treated can be different. In any case, it is likely that important events give rise to *threads* of news articles, and that threads in different languages can be coupled. We refer to this situation as *almost comparable corpora*.

3.2 Automatic Thread Identification

We propose to use news threads of almost comparable corpora for improving the performances of a bilingual IRS. In particular, if we assume that a topic has been the subject of threads in both the source and the target languages, the automatic identification of a thread in the former can help retrieving relevant documents in the latter.

The first step in our methodology hence regards the automatic identification of news threads in the source language. To this end, we propose to apply *hierarchical clustering* [3] to the documents retrieved by querying a monolingual IRS using the topic of interest. In order to reduce computational load, clustering is applied only to the first K retrieved documents, that is the ones which are potentially more relevant to the topic. The distance measure used to highlight clusters is based on the classic $tf \times idf$ weighting scheme and computed using the cosine of the angle between retrieved documents. The *inverse document frequency idf* used to calculate the distance between documents is not the same of the original *idf* computed in the first retrieval. Actually, the new inverse document frequency is computed on the first K documents chosen for the query expansion. With this approach we try to discover those words less frequent in the first K documents that may give an added value to the ones of the original query.

The clustering step gives a partition of the set of retrieved documents. We made the assumption that clusters are strictly correlated to news threads. The choice of the most relevant threads for the topic can be based on different strategies. For instance, threads with highest average rank can be chosen, as well as threads that contain the documents with the top 5 or top 10 documents. In our experiments, we chose to select only one thread, the one that contained the document with the highest rank, and to stop the clustering step when the number of clusters is 10 or the present distance between documents belonging to different clusters is less than 0.9. This choice allows us to get a good agglomeration of documents without forcing documents or clusters that are distant, in the sense of the cosine angle, one from the other to merge together. On the other hand it is possible that the cluster that contains the document with the highest rank is made up of only that document.

3.3 Query Expansion and Translation based on Thread Identification

Once that the potentially most relevant thread in the source language has been highlighted, it can be used to retrieve potentially relevant documents in the target language. We are still investigating the possibility of directly coupling threads in the source and the target languages. For the aims of the present evaluation campaign, our interest was more on the use of thread identification to partially overcome the drawbacks of simple word-by-word translators.

To this end we applied query expansion techniques to the topic in the source language, by enriching the bag of words of the topic with a set of words taken from the highlighted thread. In particular, we chose to add words that were more discriminative of the thread, that is the ones with high average tf inside the thread and high idf inside the set of retrieved documents. It is likely that this additional set of words gives a more complete description of the topic of interest, and also that it contains synonyms of the words in the topic that may allow for a more effective translation.

This expanded set of words can be translated word-by-word, possibly applying stemming, obtaining a set of tokens to be used by an IRS in the target language. The resulting ranked list of documents can be taken as the final result of a run on a given topic. Moreover, the same principle of thread identification can be applied to the target language, in order to rerank the documents according to their belonging to potentially relevant threads. After that threads are highlighted in both source and target language, a coupling between threads might help refining the results of the run. These steps have not been investigated in detail yet, and they are left as future work.

3.4 The Thread-based Algorithm

According to the proposed methodology, we developed an algorithm that have been implemented in our IRS. The main step can be described as follows.

1. The topic in the source language is used to query an IRS in the same language. A ranked list of potentially relevant documents is obtained. At this step stemming can be carried out on the source language to possibly improve the performance of the IRS.

2. The first K documents, where K is an integer that has been tested within the range $50 \leq K \leq 100$, are used as the initial set of singletons for hierarchical clustering. The distance between pairs of documents is computed as the cosine of the angle between documents.

3. The merging of clusters stops when the number of created clusters is 10 or when the present distance between documents is 0.9. The cluster that contains the top rank document is taken as the news thread that is more likely relevant to the topic in the source language.

4. The H words that are good candidates to improve query translation are used to expand the initial query in the source language, obtaining an extended set of tokens. A maximum of 10 words for the title of the query and 100 words for the description of the query have been used to expand queries during tests. The algorithm for choosing these words is described in the following section.

5. All the tokens in the extended set are translated singularly, using an on-line translation Web service, obtaining an extended set of translated tokens. These tokens are added to the word-by-word translation of the topic in the source language. We used the Google translation Web service.

6. All of the tokens in the target language are used to query an IRS in the same target language. Also at this step, stemming can be carried out in order to improve the system performances.

Finding Good Candidate Words of a Cluster

Once the news *thread* has been found in the first K retrieved documents, how do we choose the words that are going to expand the original query? We propose the following approach: we use a (tf $\times$ idf)-like weighting scheme to weight terms of a cluster. In particular, we consider the documents of the cluster as boolean vectors, so that we do not make an explicit use of term frequency. For each word w we count how many documents contain w and name it *cluster frequency* CF, and we use it in a (tf $\times$ idf)-like formula.

The following steps describe the algorithm used to choose the terms of the cluster. It is important to say that we repeat the algorithm both for the words of the title of a document and

the words of the body of a document. In this way, we can add specific title terms to the title of the query, and specific body terms to the description or narrative of a query.

1. Get the cluster that contains the first ranked document in the original retrieval; let N the number of documents of the cluster, and n be the number of documents in which the word w appears.
2. Let the *Cluster Frequency* CF of w be equal to n, and let *relative Cluster Frequency* rCF be n/N.
3. Calculate the weight of a each word as the product $(1 + log(CF)) \times (-log(rCF))$.
4. At this point, cluster words according to their weights, that is to say create sets of words with the same weights and order these sets by decreasing order.
5. Add the first set of terms to the query; continue to add set of words until the number of words that has been added exceed a predefined threshold.

A threshold of 10 words was used for the words of the query title and a threshold of 100 was used for the words of the query description in our tests.

4 Experiments

We run a set of experiments in both the monolingual and bilingual tracks using a prototype system that has been developed in our research group. A description of the prototype and experimental results are presented in the following sections.

4.1 Experimental Prototype System

We developed an experimental information retrieval system, called IRON (Information Retrieval ON), which has been used for the first time at the CLEF 2002 evaluation campaign; IRON has been completely re-engineered for CLEF 2003 and it has been further expanded for CLEF 2004 to support different character sets and to include the proposed approach for news thread identification.

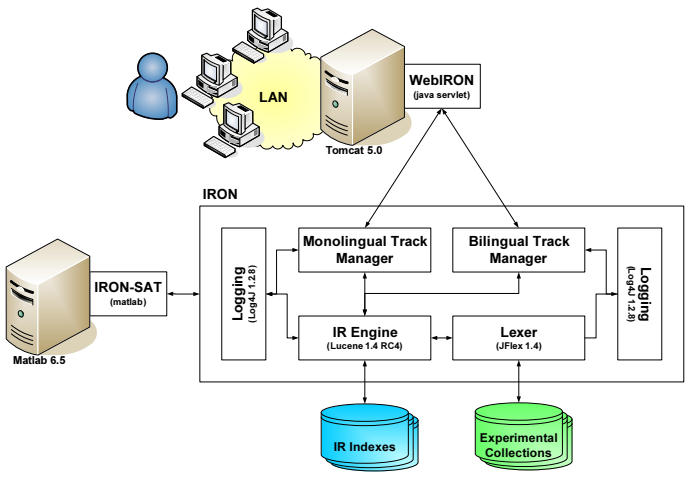

Figure 3: Architecture of IRON.

Figure 3 shows the architecture of our prototype IR system: IRON is a java multi-threaded program, which provides IR functionalities and enables concurrent indexing and searching of

document collections for both monolingual and bilingual tracks. It provides a modular environment suitable for testing the performances of different IR components allowing us to easily plug-in at runtime the components under examinations, such as lexical analyzers (lexers) or stemmers.

IRON is made up of the following components:

- **Lexer**: implements an efficient lexer using JFlex 1.4[2], a lexer generator for Java, written in Java. The current lexer is able to process any CLEF collection in a transparent way with respect to the user. The fundamental changes made to the last year lexer has been maintained and a couple of important refinements have been made.
  - Including the new collections for Finnish and Russian was a quite easy task given the modular design of our lexer. It was just a matter of writing four or five states for the automata and add them to the main lexer in a fashion presented in [1].
  - We had to manage the problem of choosing the right encoding for every collection, given the Russian collection was formatted using the UTF-8 encoding. When a file is passed to the lexer for processing, we check the encoding of the file: if it is UTF-8 encoded, we open it using the `InputStreamReader` Java class with the appropriate encoding.
  - Moreover, when the lexer is run on different platforms, the problem of platform encoding arises. In the same way as we did for the Russian collection, we had to specify for every file passed to the lexer (which was not encoded in UTF-8) that it had to be opened using the ISO8859-1 encoding.
  - All the documents of the collection are translated into UNICODE encoding, so that all the following processing is uniformly based on UNICODE.
  - Since some SGML tags may be corrupted, the lexer is now able to adapt quickly to anomalous situations like missing or corrupted SGML tags, misplaced tags, etc. Furthermore, the lexer is engineered in such a way that if a file contains documents of different collections and languages it is properly processed in a transparent way.
- **IR engine**: is built on top of the Lucene 1.4 RC4[3] library, which is a high-performance text search engine library written entirely in Java. Lucene implements the vector space model, and a (tf $\times$ idf)–based weighting scheme. Some parts of the Lucene library were completely rewritten, that is we wrote a set of parallel classes without modifying the original source code of Lucene, so that IRON remain compatible with the official Jakarta distribution of Lucene. In particular we modified those parts of Lucene concerning the text processing, such as tokenization, stop words elimination, stemming, and the query construction. Furthermore we adapted Lucene to our logging infrastructure, described below;
- **Monolingual Track Manager**: drives the underlying IR engine and provides high-level indexing and searching functionalities in order to carry out the monolingual track. It provides an high-level Application Program Interface (API) that allows us to easily plug together the different components of an IRS. This API can be further used to create a front-end application to IRON: for example we can develop a command-line or a Graphical User Interface (GUI) for a stand-alone application or a Web based User Interface (UI);
- **Bilingual Track Manager**: drives the underlying IR engine and provides high-level indexing and searching functionalities in order to carry out the bilingual track. Furthermore it implements the clustering algorithm described in Section 3. As the Monolingual Track Manager, also the Bilingual Track Manager provides an high-level API that can be used to develop different kinds of UI for IRON;
- **Logging**: provides a full-fledged log infrastructure, based on the Log4J 1.2.8[4] java library. Each other component of IRON sends information about its status to the logging infrastructure, thus allowing us to track each step of the experiment. The logging infrastructure

[2] `http://www.jflex.de`
[3] `http://jakarta.apache.org/lucene/docs/index.html`
[4] `http://logging.apache.org/log4j/docs/`

was particularly useful to debug the different components of IRON – for example to solve some encoding issues concerning the lexer – and to analyse and fine tune the different algorithms used by IRON – for example to analyse the behaviour of the clustering algorithm. Furthermore the logging infrastructure stores information about the execution time and the executed method; this information could be further analysed with a profiling tool to tune the performances of the system. This last feature has not been used this year.

IRON is partnered with two other tools:

- **WebIRON**: is a Java Servlet based Web interface. WebIRON is based on the Tomcat 5.0[5] Web server, making IRON a Web application. It provides a set of wizards which help the user to set all the parameters and choose the IR components, which are needed in order to conduct a run or, more generally, an IR experiment.
- **IRON-SAT** (IRON - Statistical Analysis Tool): is a Matlab program that interacts with IRON in order to carry the statistical analysis of the experimental results. IRON-SAT parses the trec_eval files and stores the parsed information into a data structure suitable for the following processing. It is designed so that new statistical tests or new data processing can be easily added to the existing code in a modular way. The statistical analysis is performed using the Statistics Toolbox 4.0 provided by Matlab 6.5[6]. The statistical tests reported in the following sections are all automatically produced by IRON-SAT.

4.2 Monolingual Experiments

The aim of the experiments for the monolingual track was to compare the retrieval effectiveness of the language independent stemmer, illustrated Section 2, with that of an algorithm based on a-priori linguistic knowledge – we have chosen the widely used Porter's stemmers. The hypothesis was that the proposed probabilistic approach generates stemmers that perform as effectively as Porter's stemmers. To evaluate stemming algorithms, the performances of different IR systems have been compared by changing only the stemming algorithms for different runs, all other things being equal. Our aim was to test the following hypotheses:

H': stemming does not hurt and can enhance the effectiveness of retrieval,

H'': the proposed statistical stemmers perform as effectively as Porter's ones.

Experiments were conducted for the following languages: Finnish, French, and Russian. For each track the following stemming algorithms were tested:

- **No Stem**: no stemming algorithm was applied;
- **Porter**: the stemming algorithms freely available at the Snowball[7] Web site edited by Martin Porter for different languages have been used. Russian is an exception, because the java implementation for the Russian stemmer seems to not properly process Unicode strings and so we were unable to produce runs with this stemmer;
- **STON**: the stemming algorithm based on Hidden Markov Models has been used.

As regards the stop-words used in the experiments, i.e. words which have little semantic meaning, the stop-lists available at `http://www.unine.ch/info/clef/` have been used.

Tables 1(a) and 1(b) report the general figures for 2004 monolingual topics. Both these tables show that stemming improves the performances for all the figures for all the considered languages. Thus these figures give a positive answer to both hypotheses H' and H'' because stemming improves the performance of an IRS. Moreover, the experimental evidence confirms the hypothesis that it is possible to generate stemmers using probabilistic models without or with very little knowledge about the language.

[5] `http://jakarta.apache.org/tomcat/index.html`
[6] `http://www.mathworks.com/`
[7] `http://www.snowball.tartarus.org`

Table 1: General figures for 2004 monolingual topics.

(a) Relevant retrieved document number (recall).

Algorithm	Relevant Retrieved (Recall %)		
	Finnish	French	Russian
No Stem	258 (62.46)	763 (83.38)	82 (66.66)
STON	305 (73.84)	809 (88.41)	94 (76.42)
Porter	346 (83.77)	832 (90.92)	–
Total Relevant Docs	413	915	123

(b) Precision.

Algorithm	Average Precision (%)			Exact R-Precision (%)		
	Finnish	French	Russian	Finnish	French	Russian
No Stem	39.62	38.64	28.40	36.12	38.64	28.45
STON	40.70	41.53	34.06	36.75	39.55	30.56
Porter	46.31	42.53	–	43.71	38.71	–

However the degree to which the observed differences are significant has to be measured using statistical testing methods. We used the Wilcoxon signed ranks test [2], which is a non parametric statistical test for paired samples. The runs have been compared query-by-query using the same figures reported in Table 1 with a significance level $\alpha = 5\%$.

Tables 2 and 3 allow us to answer question H' for both Porter stemmers and STON. For Finnish, the Porter stemmer exhibits an impact on the performances for all the considered measures; STON shows significant differences with respect to the case of no stemming in terms of number of relevant retrieved documents, but not for the other measures. For French, both the Porter stemmer and STON show significant differences with respect to the case of no stemming in terms of number of relevant retrieved documents and average precision. Finally for Russian, STON shows significant differences with respect to the case of no stemming in terms of number of relevant retrieved documents, but not for the other measures. Thus, in general, the hypothesis that stemming influences the performances of an IRS cannot be rejected. The impact of the stemming depends on both the language and the considered measure.

Table 4 allows us to answer to hypothesis H'' for the STON algorithm. The results show that in general the hypothesis that STON is as effective as Porter's algorithm cannot be rejected. However, for Finnish and French there are significant differences between STON and Porter's stemmers in terms of number of relevant retrieved documents, where Porter's algorithm performed better than STON.

4.3 Bilingual Experiments

As soon as we received the results for the CLEF bilingual track, we discovered an anomalous behavior of the system. In particular, some of the translated queries were completely empty. We found the error in the following point: when a word is translated from German to French, an apostrophe may appear. Since IRON reads a query with the `StreamTokenizer` Java class, we discovered that this class contains a method (namely `quoteChar()`) that is used when a quote character is encountered. If a string quote character (in our case the apostrophe) is encountered, then a *quotation* string is recognized, consisting of all characters after the string quote character, up to the next occurrence of that same string quote character, or a line terminator, or end of file. This causes IRON to discard parts of (or even whole) queries and consequently to perform badly (around 13% of average precision). When the error was fixed the general performances of the system improved significantly (up to 23% of average precision).

Table 5 presents the correct results of the bilingual track. As it can be noted, this time the differences between stem and no-stem is very subtle. Also the difference between using the first

Table 2: Comparison of monolingual No Stem and Porter runs for different measures.

Measure		Finnish	French	Russian
Rel. Retr.	No Stem > Porter	0	1	–
	No Stem = Porter	34	33	–
	No Stem < Porter	11	15	–
	Signed Rank Test (p–value)	0.10%	0.08%	–
Avg. Prec.	No Stem > Porter	12	14	–
	No Stem = Porter	10	5	–
	No Stem < Porter	23	30	–
	Signed Rank Test (p–value)	3.06%	0.53%	–
Exact R-Prec.	No Stem > Porter	7	8	–
	No Stem = Porter	22	28	–
	No Stem < Porter	16	13	–
	Signed Rank Test (p–value)	4.97%	49.79%	–

Table 3: Comparison of monolingual No Stem and STON runs for different measures.

Measure		Finnish	French	Russian
Rel. Retr.	No Stem > STON	1	3	0
	No Stem = STON	35	34	25
	No Stem < STON	9	12	9
	Signed Rank Test (p–value)	1.37%	0.54%	0.39%
Avg. Prec.	No Stem > STON	17	14	11
	No Stem = STON	10	5	7
	No Stem < STON	18	30	16
	Signed Rank Test (p–value)	65.83%	2.15%	6.79%
Exact R-Prec.	No Stem > STON	8	8	2
	No Stem = STON	25	25	27
	No Stem < STON	12	16	5
	Signed Rank Test (p–value)	56.28%	20.86%	57.81%

Table 4: Comparison of monolingual STON and Porter runs for different measures.

		Finnish	French	Russian
Rel. Retr.	STON > Porter	2	1	–
	STON = Porter	36	40	–
	STON < Porter	7	8	–
	Signed Rank Test (p–value)	1.95%	2.73%	–
Avg. Prec.	STON > Porter	15	20	–
	STON = Porter	9	5	–
	STON < Porter	21	24	–
	Signed Rank Test (p–value)	11.44%	19.92%	–
Exact R-Prec.	STON > Porter	7	11	–
	STON = Porter	24	31	–
	STON < Porter	14	7	–
	Signed Rank Test (p–value)	10.59%	51.35%	–

Table 5: General figures for bilingual German → French.

Algorithm	Rel. Retr. (Recall %)	Avg. Prec. (%)	Exact R-Prec. (%)
No Stem, first 50 docs	566 (61.85)	22.15	21.38
No Stem, first 100 docs	517 (56.50)	22.25	21.57
STON, first 50 docs	560 (61.20)	22.50	22.36
STON, first 100 docs	486 (53.11)	22.41	21.92
Porter, first 50 docs	580 (63.38)	23.17	23.00
Porter, first 100 docs	487 (53.22)	22.83	22.73
Total Relevant Docs	915	–	–

50 or the first 100 documents is small, although the runs using the first 50 documents attain a better recall than the ones using the first 100 documents.

We performed the same statistical analysis of the previous section, but it does not give any significative difference between stem and no stem, and between using 50 or 100 documents for clustering. There is an exception to this general trend: Porter, first 50 docs, shows a better average precision (p = 3.57%) than No Stem, first 50 docs.

5 Conclusions and future work

The idea of minimizing human labor and computing resources was the main point of the many experiments that were carried out this year by the IMS research group. The automatic stemmer generation using Hidden Markov Models confirmed to be a valid alternative way to language dependent stemmers like Porter's ones. Statistical tests on the mono-lingual track gave evidence to not reject the hypothesis that, in general, STON is as effective as the Porter's algorithm and that stemming does not hurt or even enhances retrieval performances. Automatic identification of news threads together with hierarchical clustering were used for query expansion for the bi-lingual track. A word-by-word translation using Google on-line services was carried out. Although stemming seemed to not affect performance as in the monolingual track, results given by our query expansion/translation approach are encouraging. Further experiments and refinements may bring performances of this approach comparable to state-of-the-art systems.

References

[1] G. M. Di Nunzio, N. Ferro, M. Melucci, and N. Orio. The University of Padova at CLEF 2003: Experiments to Evaluate Probabilistic Models for Automatic Stemmer Generation and Query Word Translation. In C. Peters, editor, *Working Notes for the CLEF 2003 Workshop*, pages 211–223. `http://clef.iei.pi.cnr.it:2002/2003/WN_web/27.pdf`, 2003.

[2] J. D. Gibbons. *Nonparametric Statistical Inference.* Marcel Dekker, Inc., New York, USA, 2nd edition, 1985.

[3] S.C. Johnson Hierarchical Clustering Schemes. *Psychometrika*, 32, pages 241–254, 1967.

[4] J.-Y. Nie, M. Simard, P. Isabelle, and R. Durand Cross-language Information Retrieval Based on Parallel Texts and Automatic Mining of Parallel Texts from the Web. *Proc. of the 22nd ACM SIGIR Conference*, pages 74–81, Berkeley, CA, 1999.

[5] L. Rabiner and B.H. Juang. *Fundamentals of speech recognition.* Prentice Hall, Englewood Cliffs, NJ, 321–389, 1993.

[6] P. Sheridan and J.P. Ballerini Experiments in Multilingual Information Retrieval Using the SPIDER System. *Proc. of the 19th ACM SIGIR Conference*, pages 58–65, Zurich, Switzerland, 1996.