Experiments to Evaluate Probabilistic Models for Automatic Stemmer Generation and Query Word Translation

Giorgio M. Di Nunzio, Nicola Ferro, Massimo Melucci, and Nicola Orio

Department of Information Engineering,
University of Padova,
Via Gradenigo, 6/a – 35031 Padova – Italy
{dinunzio,nf76,melo,orio}@dei.unipd.it

Abstract. The paper describes statistical methods and experiments for stemming and for the translation of query words used in the monolingual and bilingual tracks in CLEF 2003. While there is still room for improvement in the method proposed for the bilingual track, the approach adopted for the monolingual track makes it possible to generate stemmers which learn directly how to stem the words in a document from a training word list extracted from the document collection, with no need for language-dependent knowledge. The experiments suggest that statistical approaches to stemming are as effective as classical algorithms which encapsulate predefined linguistic rules.

1 Introduction

The Information Management Systems (IMS)[1] research group of the Department of Information Engineering of the University of Padova participated in the CLEF monolingual track in 2002 for the first time; on that occasion experiments on graph-based stemming algorithms for Italian were specifically carried out. The graph-based stemmer generator proposed achieved a retrieval effectiveness comparable to that of Porter's stemmers.

The spectrum of languages covered has been increased to five and the theoretical framework underlying the stemmer generator has been redefined this year. Although our approach has shifted from a graph-based to a probabilistic framework, the characterizing notion of mutual reinforcement between stems and derivation has been preserved. A new approach to stemmer generation based on Hidden Markov Models has also been experimented and has achieved good results.

2 Monolingual Track

Our approach to monolingual retrieval is focused on the development of stemming algorithms for five languages – i.e., Dutch, French, German, Italian, and

[1] http://www.dei.unipd.it/~ims

Spanish – that have been used as a testbed. Our aim has been to develop algorithms which do not exploit any linguistic knowledge about the morphology of a given language and the rules to form word derivations. To this end, the assumption was that the parameters of a statistical model can be inferred from the set of words of a given language and that they can be applied for stemming words. Two different approaches have been tested. A probabilistic framework for the notion of mutual reinforcement between stems and derivation is presented in Section 2.1, whereas a framework based on Hidden Markov Models (HMMs) is presented in Section 2.2.

2.1 A Probabilistic Framework for Stemmer Generation

The *Stemming Program for Language Independent Tasks* (SPLIT) is a language independent stemming procedure which was originally proposed in [1, 2] to automatically generate stemmers for some European languages. SPLIT is based on a suffix stripping paradigm and assumes that the stem of a word can be found by splitting the word and holding the first part as a candidate stem. Therefore, the problem of word stemming is reduced to the problem of finding the right split for the word. A probabilistic framework for SPLIT has been designed and experimented and the following results have been obtained:

– a technique based on the concept of *mutual reinforcement* between stems and derivations to estimate the probabilities of the framework;
– an algorithm that implements this technique and that generates a stemmer for a given language.

Probabilistic Framework. Given a finite collection W of words, a word $w \in W$ of length n can be split into $n-1$ possible parts so that no empty substring is generated. Therefore each split is associated with a pair of substrings called *prefix* and *suffix* respectively. The concatenation of these substrings form the word w. In our probabilistic framework it is assumed that the prefix-suffix pairs are not equiprobable, but that the concatenation of a *stem* with a *derivation* is a more probable event than the concatenation of two generic prefixes and suffixes.

Thus a maximum likelihood criterion to identify the most probable pair of sub–strings, i.e. the stem and derivation, can be employed as shown in Figure 1. Let U be the set of substrings generated after splitting every word $w \in W$ of a given language into all the possible prefix-suffix pairs. If $x \in U$ and $y \in U$ are the prefix and the suffix of the word w respectively, then $w = xy$. Let $\Omega(w) = \{(x, y) \in U \times U : w = xy\}$ be the set of all the prefix-suffix pairs which form a given word w. Thus the pair $(x, y)^*$ which is the most probable split of a given word w can be found as follows:

$$(x, y)^* = \arg\max_{\Omega(w)} \Pr(x, y) \ . \tag{1}$$

Mutual Reinforcement Model. A mutual reinforcement relationship exists among substrings of a language and can be stated as follows:

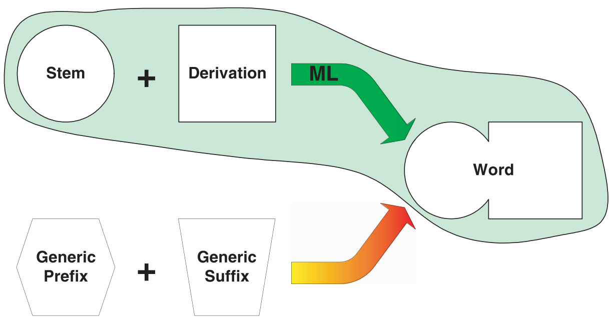

Fig. 1. Choice of the (stem, derivation) pair of a word within the probabilistic framework

> Stems are prefixes that are more likely to be completed by derivations, and derivations are suffixes that are more likely to complete stems.

Figure 2 gives an intuitive view of the mutual reinforcement relationship where, for the sake of clarity, only a small subset of prefixes and suffixes are reported. On the left of the figure a community of stems and derivations is shown. In the example shown `comput` is probably a stem because it refers to substrings which are in turn used to create other words: `ation` and `er` are referred to by `comput` and are used to create `compilation` or `reader`. On the right of the figure, a community of generic suffixes is depicted for prefix `c`; the prefix `c` is not likely to be a stem, because it forms words with suffixes, e.g. `omputer` or `ompilation`, that are not suffixes of any other words. This sort of *coupled frequent usage* allows

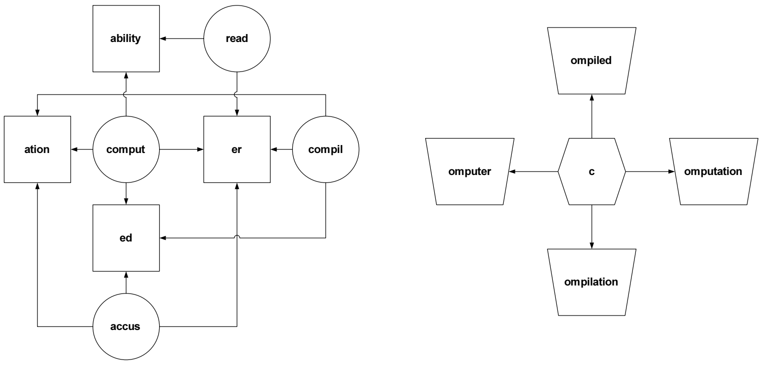

Fig. 2. Example of mutual reinforcement relationship

us to estimate the prefix and suffix probability distribution which is necessary to compute the probability distribution of the pairs so as to apply the probabilistic framework.

The mutual reinforcement relationship can be expressed as follows:

$$\begin{aligned}\Pr(x) &= \sum_{y\in Y}\Pr(x,y) = \sum_{y\in Y}\Pr(x \mid y)\Pr(y)\ ,\\ \Pr(y) &= \sum_{x\in X}\Pr(x,y) = \sum_{x\in X}\Pr(y \mid x)\Pr(x)\ ,\end{aligned} \tag{2}$$

where:

- $\Pr(x)$ is the probability that x is a good prefix, that is a prefix candidate to be a stem. Similarly $\Pr(y)$ is the probability that y is a good suffix, that is a suffix candidate to be a derivation;
- $\Pr(y \mid x)$ is the conditional probability that a word ends with the suffix y given that it begins with the prefix x;
- $\Pr(x \mid y)$ is the conditional probability that a word begins with the prefix x given that it ends with the suffix y;
- $X \subseteq U$ is the set of all the prefixes and $Y \subseteq U$ is the set of all the suffixes.

The SPLIT Algorithm. Figure 3 shows the architecture of the SPLIT algorithm described below:

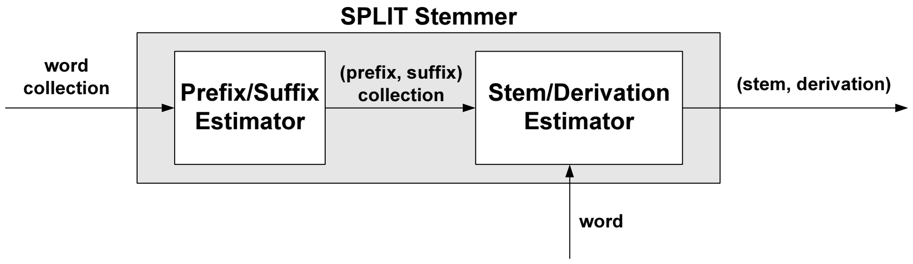

Fig. 3. Architecture of the SPLIT algorithm

- *Prefix/Suffix Estimation* is a global step which tries to infer some knowledge about the language by estimating the distribution of prefixes and suffixes, according to (2). It is a global step because it concerns the whole set U and not a word in particular. This step uses the following estimations of the probabilities:
  - $\Pr(x \mid y) = \frac{1}{|P(y)|}$, where $P(y) = \{x \in U : \exists w \in W, w = xy\}$ and $|P(y)|$ is the number of words which end with suffix y;
  - $\Pr(y \mid x) = \frac{1}{|S(x)|}$, where $S(x) = \{y \in U : \exists w \in W, w = xy\}$ and $|S(x)|$ is the number of words which begin with prefix x.

The algorithm iteratively computes:

$$\Pr^{(t)}(x) = \sum_{y \in Y} \Pr(x \mid y) \Pr^{(t-1)}(y) \ ,$$

$$\Pr^{(t)}(y) = \sum_{x \in X} \Pr(y \mid x) \Pr^{(t)}(x) \ ,$$

for $t = 0, 1, 2, \ldots$, where $\Pr^{(0)}(y)$ is a vector of uniform probabilities.

– *Stem/Derivation Estimation* is a local step, which tries to distinguish among all the pairs which lead to the same word, according to (1). It is a local step, because it concerns a particular word. Equation (1) is solved by considering two different cases:
  1. $\Pr(x, y) = \Pr(x) \Pr(y \mid x)$,
  2. $\Pr(x, y) = \Pr(x) \Pr(y)$.

  Case 1 takes into account the possible stochastic dependence between x and y. On the other hand, case 2 considers x and y as independent events, because $\Pr(x)$ and $\Pr(y)$ have absorbed some knowledge about the morphology of the language through their estimation by (2), which already took into account the dependence between x and y.

  In addition a little linguistic knowledge is injected by inserting a heuristic rule which forces the length of the prefix to be at least α characters and the length of the suffix to be at most β characters.

  The CLEF 2002 collection has been used to train the algorithm and to set appropriate values for all the parameters. Once training was finished, the following parameters were chosen because they gave the best performances in each language:

  - Dutch: case 2 with $\alpha = 4$, $\beta = 4$;
  - French: case 1 with $\alpha = 1$, $\beta = 3$;
  - German: case 2 with $\alpha = 4$, $\beta = 4$;
  - Italian: case 1 with $\alpha = 1$, $\beta = 3$;
  - Spanish: case 2 with $\alpha = 3$, $\beta = 3$.

2.2 An Approach Based on Hidden Markov Models for Stemmer Generation

Another statistical approach to stemming based on Hidden Markov Models (HMM) [3] has been experimented. HMMs are finite-state automata where transitions between states are ruled by probability functions. 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. For each state the parameters that completely define an HMM are the probabilities of being the initial and the final state, the transition probabilities to any other state, and the probability that a given symbol is emitted.

HMMs as Word Generators. HMMs are particularly useful to model processes that are generally unknown but that 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 an HMM. The HMM starts in an initial state and performs a sequence of transitions between states by emitting a new letter at each transition until it stops at a final state. In general, several 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 normally addressed as *decoding*, for which an efficient algorithm exists: the Viterbi decoding.

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

- an initial state belongs only to the stem-set, i.e. a word always starts with a stem;
- the 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;
- a final state belongs to both sets, i.e. a stem can have a number of different derivations, but it may also have no suffix.

A general HMM topology that fulfills these conditions is depicted in Figure 4. Once a complete HMM is available for a given language, stemming can

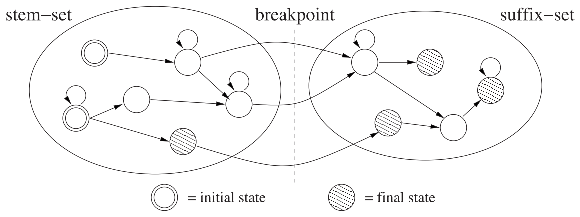

Fig. 4. HMM topology with the stem-set and the suffix-set highlighted

be straightforwardly carried out 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 *breakpoint*. If there is no breakpoint then the word has no suffix, otherwise the sequence of letters observed before the breakpoint is taken as the stem and the one observed after is taken as the suffix.

Training the HMM. The proposed topology defines the number of states, the labels indicating the sets to which the states 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 Baum-Welch expectation-maximization (EM) algorithm. As in the case of SPLIT, 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 performing an unsupervised training of the HMM using only a sample of the words of the considered language. The training set can be built at random from documents that are available at indexing time. It can be noted that an unsupervised training does not guarantee that the breakpoint of the most probable path has a direct relationship with the stem and the suffix of a given word. In order to create such a relationship, the injection of some more knowledge about the general rules for word inflection is proposed. Thus it has been reasonably assumed that the number of different suffixes for each language 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 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 5, where the maximum length of a suffix is set to four letters, and the minimum length of a stem is set to three letters.

After the redefinition of the suffix-set topology, the HMM can be trained by performing the EM algorithm using a training set of words. Given the previous assumption, it is likely that a letter sequence that corresponds to a suffix will be frequently present in the training set. For this reason, the EM algorithm will give a high probability that the letters of frequent suffixes are emitted to the states in the suffix-set. For example the unique state of a suffix-set chain will emit the last letter of each word with the highest probability, the states in a two-state suffix-set chain will respectively emit the most frequent couple of final letters of each word, and so on. Once the model has been trained the path that terminates with the most frequent sequence is expected to have a high probability.

The STON Algorithm. An algorithm called STON has been developed to test the methodology and the changes in retrieval effectiveness depending on some of its parameters. STON needs an off-line training, while stemming can be performed on-line for any new word. Once training has ended, STON receives as input a sequence of letters corresponding to a word and gives as output the position of the breakpoint. Hence, STON performs in two steps:

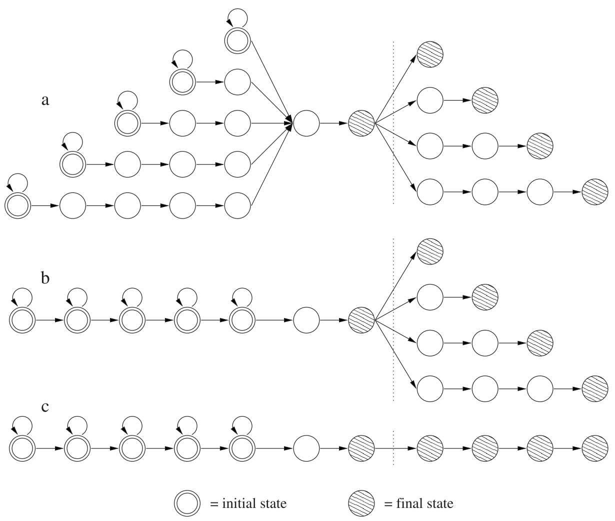

Fig. 5. Three topologies of the HMM that have been tested

- *Training/off-line*: STON computes through the EM algorithm:

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

  given a set of words $w \in W_L$ taken from a collection of documents written in a language L, and given an HMM with parameters λ which define the number of states and the set of allowable transitions. This step needs to be performed only once for each language, possibly using a sample of the words of a document collection.
- *Stemming/on-line*: STON computes the most probable path q across the states corresponding to w_L by using the Viterbi decoding:

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

  given a word w_L written in language L and a trained model λ_L, Decoding can be carried out also for words that were not part of the training set. Once the most probable path q^* is computed, the position of the breakpoint, and hence the length of the stem, can be computed by a simple inspection of the path, that is considering when the path enters the suffix-set.

3 Cross-Language Retrieval Using Mutual Reinforcement

Our approach to bilingual retrieval aimed at testing whether the notion of mutual reinforcement relationship which has been successfully applied to stemming can effectively work also in the context of keyword translation based on machine readable dictionaries. It is assumed that any description of the context of a translation is absent in the dictionary. This way a dictionary is simply a list of records and a record relates a source word to the list of possible target words which are translations of the source.

One problem of dictionaries is the lack of entries for many words in the document collection. The problem of missing translations is especially acute whenever small or simple dictionaries are employed. If no translations are available for a source word, a solution is to translate the words which are most closely related to the source word. The set of words which are most related to another is called context.

A context of a word can be built using collocates. A collocate of a word is one that frequently occurs just before or just after the word in the documents of the collections. If collocates are available, and a source word cannot be translated, the translations of the collocates of the source word can be found. This way a source word might not be directly translated, but could be connected to the target language contexts which translate the context of the source word.

Thus translation is performed between source word contexts and target word contexts rather than between single words. As context translations are uncertain events, they have been modelled using a probability space.

Let Y be the set of target words, X be the set of source words, and $D \subseteq X \times Y$ be the dictionary. D is also the universe of the elementary events of the probabilistic model. Let us define $D'(r) = \{(x, y) \in D \mid x = r\}$ and $D''(t) = \{(x, y) \in D \mid y = t\}$ as the subset of translations of r to any target word and the subset of translations for which t is a target word respectively. The contexts have been defined as subsets $X(r) \subseteq X$ or $Y(t) \subseteq Y$ depending on whether they are of a source word or of a target word respectively.

A target context $Y(t)$ is a translation of the source context $X(r)$ if there is $(x, y) \in D$ such that $x \in X(r)$ and $y \in Y(t)$. Of course, there may be $0, 1, \ldots$ pairs (x, y) such that $x \in X(r)$ and $y \in Y(t)$ and then $Y(t)$ is a translation of the source context $X(r)$ to different degrees. Intuitively, the degree to which $Y(t)$ is a translation of $X(r)$ is directly proportional to the size of the set $D(r, t) = \{(x, y) \in D, x \in X(r), y \in Y(t)\}$ which is the set of translations found between the words of the context of r and those of context of t.

Using this probabilistic model the best target context, i.e. the most probable target context which is a translation of the source context can be found. The selection of the best translation can be formalized by $\Pr(Y(t) \text{ translates } X(r))$ and is approximated by the conditional probability:

$$\Pr(Y(t) \text{ translates } X(r)) \approx \Pr(Y(t) \mid X(r)).$$

Given that the probability that $Y(t)$ translates $X(r)$ is directly related to $|D(r, t)|$, the estimation formulas

$$\Pr(Y(t) \mid X(r)) = \frac{|D(r,t)|}{|D'(r)|} \qquad \Pr(X(r) \mid Y(t)) = \frac{|D(r,t)|}{|D''(t)|}$$

can be defined. At search time, all the target contexts $Y(t)$ corresponding to each possible translation t of r are considered for each source query word r. As there may be several candidate translation contexts $Y(t)$, a criterion to choose which translation context should be used is necessary.

The idea is that the best translations of a source context are the ones which are a translation of other source contexts and that can be conversely translated back to one of the source contexts. This mutual reinforcement relationship between contexts states that the best target translations of a source context are translated by the best source contexts, and viceversa. Thus the following mutual definition is considered:

$$\Pr(Y(t)) = \sum_{X(r)} \Pr(Y(t) \mid X(r)) \Pr(X(r)) \ ,$$

$$\Pr(X(r)) = \sum_{Y(t)} \Pr(X(r) \mid Y(t)) \Pr(Y(t)) \ ,$$

where $\Pr(Y(t))$ is the probability that $Y(t)$ is a translation of a source context, $\Pr(X(r))$ is the probability that $X(r)$ is a translation of a target context. The most probable $Y(t)$ has been taken as a translation of each query word b such that a translation of a word in $X(r)$ occurs in $Y(t)$.

4 Experiments

The aim of the experiments for the monolingual track was to compare the retrieval effectiveness of the language independent stemmers, illustrated in the previous sections, 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 approaches generate 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: Dutch, French, German, Italian and Spanish. For each track four different stemming algorithms were tested:

- **No Stem**: no stemming algorithm was applied;
- **Porter**: the stemming algorithms freely available at the Snowball Web site edited by Martin Porter for different languages have been used;
- **SPLIT**: the stemming algorithm based on the notion of mutual reinforcement has been used;
- **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/`

Table 1. Relevant retrieved document number (recall) for 2003 Topics

Algorithm	**Relevant Retrieved (Recall %)**				
	Dutch	**French**	**German**	**Italian**	**Spanish**
No Stem	1,419 (89.98)	869 (91.86)	1,330 (72.88)	488 (60.32)	2,084 (88.01)
SPLIT	1,420 (90.04)	886 (93.66)	1,376 (75.40)	497 (61.43)	2,122 (89.61)
STON	1,386 (87.33)	891 (94.19)	1,384 (75.84)	503 (62.18)	2,148 (90.71)
Porter	1,416 (89.79)	911 (96.30)	1,434 (78.58)	492 (60.82)	2,202 (92.99)
Total Relevant Docs	1,577	946	1,825	809	2,368

have been used. These stop-lists are cross-linked by the CLEF consortium for the participants of the CLEF campaigns. The details of the retrieval system used for the experiments can be found in [4, 5].

For each language, the results over all the queries of the test collection have been summarized. Table 1 compares the number of relevant retrieved documents and the recall for the different algorithms under examination. Table 2 reports, for each language, the average precision attained by the system with the considered stemming algorithms. Table 3 reports the exact R-precision attained by the system, for each language. The exact R-precision is the precision after R documents have been retrieved, where R is the number of relevant documents for the topic.

In general stemming improves the recall. The Dutch language represents an exception to this note, since both the STON and the Porter stemmer retrieve less relevant documents than without any stemmer.

Note that for French, German, Italian and Spanish stemming positively affects the precision, thus improving the overall performance of the system, since the recall has also improved. Dutch stemming does not degrade the overall per-

Table 2. Average precision for 2003 Topics

Algorithm	**Average Precision (%)**				
	Dutch	**French**	**German**	**Italian**	**Spanish**
No Stem	42.11	42.86	34.92	34.76	39.27
SPLIT	42.84	45.60	37.11	38.17	38.25
STON	42.57	45.67	36.68	34.66	40.56
Porter	43.49	45.87	37.88	35.53	43.42

Table 3. Exact R-precision for 2003 Topics

Algorithm	**Exact R-Precision (%)**				
	Dutch	**French**	**German**	**Italian**	**Spanish**
No Stem	40.51	39.45	36.59	36.32	40.26
SPLIT	41.54	43.22	37.80	38.39	39.85
STON	39.66	42.20	37.53	33.26	39.90
Porter	40.55	41.68	38.73	34.79	42.70

formances of the system. Furthermore, when the stemming algorithms positively affect the performances, SPLIT and STON perform as effectively as Porter's stemmer.

Thus these figures gives a positive answer to both hypotheses H' and H'' since stemming does not hurt and sometimes improves the performance of an information retrieval system (IRS). 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.

However the degree to which the observed differences are significant has to be measured using statistical testing methods. To test the hypotheses more soundly, the runs have been compared using the following measures: number of relevant retrieved documents (labelled Rel. Retr.), average precision (labelled Avg. Prec) and exact R-precision (labelled Exact R-Prec.), which are the same measures used previously. Furthermore the runs have also been compared using precision after 10, 20 and 30 retrieved documents (labelled, respectively, P @ 10 docs, P @ 20 docs and P @ 30 docs). These latter measures correspond to performance assessing from a more user-oriented point of view than a system-oriented one. If stemming is applied in an interactive context, such as that of a search engine or of a digital library, the ranking used to display the results to the user acquires great importance: in fact, it would more interesting to know if the user finds the relevant document after 10 or 20 retrieved documents instead of knowing if successful retrieval is reached after 50% retrieved documents.

Table 4 allows us to answer question H'. As far as the Dutch language is concerned, with our approach stemming does not exhibit significant differences with respect to the case of no stemming. For French, stemming 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 German, stemming exhibits an impact on the performances for all the considered measures with the exception of the exact R-precision. In the case of Italian, there is no clear indication whether the null hypothesis should be rejected or not. Finally for Spanish, stemming clearly influences the performances in terms of number of relevant retrieved documents and average precision; for the other measures there is no strong evidence for accepting the null hypothesis.

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: stemming for the Dutch language is little effective.

Table 5 allows us to answer hypothesis H'' for the SPLIT algorithm. The results show that in general the hypothesis that SPLIT is as effective as Porter's algorithm cannot be rejected. However, there is some exception to this observation: with French and German, there are some significant differences in terms of number of relevant retrieved documents; for Spanish Porter's stemmer significantly performs better than SPLIT in terms of average precision, P @ 20 docs and P @ 30 docs.

Table 4. Comparison of No Stem and Porter runs for different measures

Measure		Dutch	French	German	Italian	Spanish
Rel. Retr.	No Stem > Porter	6	1	4	3	6
	No Stem = Porter	43	41	33	41	28
	No Stem < Porter	7	10	19	7	23
	Signed Rank Test (p–value)	83.94%	0.49%	0.09%	43.16%	0.06%
Avg. Prec.	No Stem > Porter	26	23	20	18	22
	No Stem = Porter	4	6	1	11	1
	No Stem < Porter	26	23	35	22	34
	Signed Rank Test (p–value)	73.61%	39.72%	0.89%	53.64%	1.07%
Exact R-Prec.	No Stem > Porter	10	12	16	16	12
	No Stem = Porter	34	28	18	25	22
	No Stem < Porter	12	12	22	10	23
	Signed Rank Test (p–value)	79.51%	52.96%	16.17%	34.73%	6.66%
P @ 10 docs	No Stem > Porter	12	11	11	13	12
	No Stem = Porter	34	26	23	28	23
	No Stem < Porter	10	15	22	10	22
	Signed Rank Test (p–value)	89.60%	33.16%	4.96%	53.16%	31.06%
P @ 20 docs	No Stem > Porter	7	14	8	9	14
	No Stem = Porter	33	25	22	29	18
	No Stem < Porter	16	13	26	13	25
	Signed Rank Test (p–value)	10.49%	71.81%	2.59%	51.26%	17.72%
P @ 30 docs	No Stem > Porter	13	12	9	12	9
	No Stem = Porter	28	25	24	30	27
	No Stem < Porter	15	15	23	9	21
	Signed Rank Test (p–value)	25.34%	12.63%	1.21%	62.63%	6.10%

Table 6 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, German is an exception with significant differences between STON and Porter's stemmers in terms of number of relevant retrieved documents, where Porter's algorithm performed better than STON. It is worth noting that the reliability of the statistical test might be affected by the presence of a high number of tied values for some measures, such as for example the number of relevant retrieved documents. In particular the omission of tied observations which is performed by both the sign test and the signed rank test introduces bias toward the rejection of the null hypothesis, as reported by [6].

Bilingual Experiments. Free dictionaries available on the Web at `http://www.travlang.com/Ergane` and `http://www.freedict.com/` have been used. Source words have been stemmed and the dictionaries have been merged after stemming. Porter's stemmers have been used because they are considered to be standard algorithms. Stemming increased the number of translations but reduced the number of entries. The five most frequent collocates of each keyword were computed to create word contexts for each language collection. Table 7

Table 5. Comparison of SPLIT and Porter runs for different measures

		Dutch	**French**	**German**	**Italian**	**Spanish**
Rel. Retr.	SPLIT > Porter	6	1	2	6	11
	SPLIT = Porter	44	39	38	43	27
	SPLIT < Porter	6	12	16	2	19
	Signed Rank Test (p–value)	96.97%	0.24%	0.11%	46.09%	5.88%
Avg. Prec.	SPLIT > Porter	24	22	22	25	19
	SPLIT = Porter	3	9	3	12	1
	SPLIT < Porter	29	21	31	14	37
	Signed Rank Test (p–value)	71.99%	72.17%	25.53%	5.59%	0.43%
Exact R-Prec.	SPLIT > Porter	10	13	16	15	14
	SPLIT = Porter	29	31	19	28	21
	SPLIT < Porter	17	8	21	8	22
	Signed Rank Test (p–value)	82.88%	56.62%	41.96%	3.86%	12.95%
P @ 10 docs	SPLIT > Porter	12	10	13	12	11
	SPLIT = Porter	34	29	29	31	25
	SPLIT < Porter	10	13	14	8	21
	Signed Rank Test (p–value)	70.47%	62.52%	47.61%	13.35%	14.61%
P @ 20 docs	SPLIT > Porter	5	11	13	14	9
	SPLIT = Porter	34	31	22	33	20
	SPLIT < Porter	17	10	21	4	28
	Signed Rank Test (p–value)	8.44%	98.61%	2.28%	2.25%	0.07%
P @ 30 docs	SPLIT > Porter	10	6	13	16	9
	SPLIT = Porter	33	31	24	28	19
	SPLIT < Porter	13	15	19	7	29
	Signed Rank Test (p–value)	24.69%	11.72%	2.96%	30.73%	0.02%

summarizes the size and the coverage after stemming and merging the two dictionaries.

The experimental results were rather disappointing. Porter's stemming algorithm used on a source language was rather aggressive and was thus detrimental to retrieval performance because many translations were erroneously mixed. Furthermore, the procedure to generate the contexts was not very effective because many contexts contained unrelated words. The average precision was between 15% and 20%. However, we believe that the approach will stimulate further research.

5 Conclusions and Future Work

This year the IMS research group has carried out many experiments and developed methodologies for both automatic stemmer generation and query translation. The first methodology has been tested in the monolingual track of CLEF, while the second has been tested in the bilingual track. The idea underlying both automatic stemmer generation and query translation has been the use of diverse

Table 6. Comparison of STON and Porter runs for different measures

		Dutch	**French**	**German**	**Italian**	**Spanish**
Rel. Retr.	STON > Porter	3	5	3	6	9
	STON = Porter	45	43	37	40	30
	STON < Porter	8	4	16	5	18
	Signed Rank Test (p–value)	10.16%	100.00%	0.92%	46.48%	31.42%
Avg. Prec.	STON > Porter	24	21	22	21	28
	STON = Porter	7	10	4	11	1
	STON < Porter	25	21	30	19	28
	Signed Rank Test (p–value)	62.95%	98.50%	12.38%	74.70%	23.37%
Exact R-Prec.	STON > Porter	13	15	16	9	20
	STON = Porter	28	28	21	26	16
	STON < Porter	15	9	19	16	21
	Signed Rank Test (p–value)	34.46%	75.33%	39.89%	57.20%	25.14%
P @ 10 docs	STON > Porter	7	10	8	8	13
	STON = Porter	38	28	36	27	30
	STON < Porter	11	14	12	16	14
	Signed Rank Test (p–value)	19.56%	51.90%	60.06%	23.70%	54.73%
P @ 20 docs	STON > Porter	12	10	7	16	18
	STON = Porter	32	28	30	23	21
	STON < Porter	12	14	19	12	18
	Signed Rank Test (p–value)	96.57%	43.17%	7.28%	66.77%	57.61%
P @ 30 docs	STON > Porter	9	9	13	14	16
	STON = Porter	30	29	27	25	23
	STONh < Porter	17	14	16	12	18
	Signed Rank Test (p–value)	6.29%	22.31%	30.37%	75.07%	25.14%

Table 7. A summary of the dictionaries employed

Language	No. of entries	Av. No. of translations
German	67889	3.69
French	29819	2.97
Spanish	15697	3.52
Italian	8958	3.93

probabilistic models. Whereas the probabilistic models for stemmer generation have confirmed the positive results observed last year, those employed for query translation need further experiments and refinement.

Acknowledgments

The work reported in this paper has been conducted in the context of a joined program between the Italian National Research Council (CNR) and the Ministry of Education (MIUR), under the law 449/97-99.

References

1. Agosti, M., Bacchin, M. Ferro, N., Melucci, M.: Improving the automatic retrieval of text documents. In Peters, C., Braschler, M., Gonzalo, J., Kluck, M., eds.: Proceedings of Cross Language Evaluation Forum 2002. LNCS 2785, Springer-Verlag (2003) 279–290
2. Bacchin, M., Ferro, N., Melucci, M.: The effectiveness of a graph-based algorithm for stemming. In: Proceedings of the Internation Conference on Asian Digital Libraries, Singapore (2002) 117–128
3. Rabiner, L., Juang, B.: Fundamentals of speech recognition. Prentice Hall, Englewood Cliffs, NJ (1993)
4. Di Nunzio, G.: The CLEF2003 lexer. (`http://www.dei.unipid.it/~dinunzio/CLEF2003Lexer.pdf`)
5. Di Nunzio, G., Ferro, N., Melucci, M., Orio, N.: The University of Padova at CLEF 2003: Experiments to evaluate probabilistic models for automatic stemmer generation and query word translation. In Peters, C., Borri, F., eds.: Working Notes for the CLEF 2003 Workshop. (2003) 211–223
6. Gibbons, J.D.: Nonparametric Statistical Inference. 2nd edn. Marcel Dekker, Inc., New York, USA (1985)