Design, implementation, and evaluation of a methodology for automatic stemmer generation*

Massimo Melucci Nicola Orio

Abstract

This article illustrates a statistical approach based on Hidden Markov Models (HMMs) which faces the problem of generating automatically stemmers. The proposed approach ensures little effort so as to insert new languages on the system even if little linguistic knowledge is available. This is an advantage that becomes crucial especially for Digital Libraries which are often constructed for a particular institution or nation, and can manage a great amount of documents written in more local languages. The evaluation described in the paper shows that the stemmers implemented by means of HMMs are as effective as those based on linguistic rules.

*This is an extended and revised version of Melucci and Orio's paper (2003) – new information and experimental results have been reported in this version.

Contact

Author	Massimo Melucci
Electronic mail	melo@dei.unipd.it
Postal address	Dipartimento di Ingegneria dell'informazione
	via Gradenigo n. 6/a – 35131 – Padova, Italy
Telephone	+39-049-827-7927
Facsimile	+39-049-827-7799

1 Introduction

The aim of a textual information retrieval (IR) system is to retrieve documents relevant to the end user's information need expressed through a textual query. To achieve this goal, the system matches the semantic content of the documents with the query content. In general, matching document content with query content is a difficult task especially for non-textual documents or queries because of the difficulty of assigning a meaning to document or query features. For textual documents and queries the task is made easier than for non-textual ones because meaning can be inferred for terms and the system just matches the inferred meaning of each document term with the meaning of each query term. A simple but effective approach for matching term meanings is based on the comparison of term forms – the hypothesis is that if two terms have similar forms then they might have the same meaning. However two drawbacks occur; two terms with similar forms can have different meanings (polysemous terms) or a meaning can be implemented by terms with different forms (synonymous terms). Different techniques have been proposed to address term ambiguity – many of them rely on complex and effective query modification techniques (Efthimiadis, 1996). Simpler but still effective methods can be applied if words that have similar meaning share the same root, or stem; e.g., for `computer` and `computing` that are derivations to the verb `to compute`.

Stemming is the task of finding the string that represents the stem of a word. Stemming is commonly carried out by IR systems to conflate different word forms together so that they might refer to the same meaning. This way stemming allows the system to retrieve those documents that do not exactly match query words, but that have word stems that match query word

stems. Stemming then boosts retrieval effectiveness if the words with a common stem share a similar meaning. On the other hand, overstemmed words with different meanings can be conflated; e.g., the stem `comp` is shared by `computer` and `company`. From a human-computer interaction point of view the use of a stemmer is intuitive to many users, who can submit the query to the system using a specific word without worrying about variants and inflections. Yet, presenting stems to the user may be disturbing if derivational suffixes are removed, therefore the user should be always provided with whole words, while the system is processing stems.

The design of a stemming algorithm may follow either a linguistic approach using prior knowledge of the morphology of the specific language, or a statistical approach using some methods based on statistical principles, which infer the word formation rules of the language for which the stemmer is developed. It is evident that the effectiveness of the linguistic approach depends on the availability of linguistic resources and know-how. However, a linguistic knowledge-based approach implies manual labor that has to be done by experts in linguistics – as a matter of fact, it is necessary to formalize the word formation rules, which is a difficult task especially for those languages whose morphology is complex.

The linguistic approaches to stemming are made more difficult in a multilingual context than in a monolingual one. Therefore the retrieval of documents written in more languages may be quite mature only if enough linguistic knowledge is available. "The technology has been developed and proved in the context of languages with many speakers, e.g., English, Spanish, French, German, Chinese, and Arabic. One challenge now is developing ways to very quickly (a few weeks) and inexpensively (a few person-weeks) find/create data for languages where resources are minimal today" (Allan

and Croft, 2003). Stemming algorithms based on statistical methods ensure little effort so as to insert new languages on the system even if little linguistic knowledge is available. This is an advantage that becomes crucial especially for IR applications, like the search services of Digital Libraries which are often constructed for a particular institution or nation, and can manage a great amount of documents written in more local languages.

This article illustrates a statistical approach based on Hidden Markov Models (HMMs) which faces the problem of generating automatically stemmers. The generated stemmers detect the most probable stem for any input word in an inflectional language whose words are sequences of symbols drawn from an alphabet. The method is fully automatic because a HMM implements an algorithm that is then used as stemmer. A negligible amount of work is devoted to design a simple graph representing the structure of a finite-state machine that generates words in the language for which the stemmer is needed. The article also presents an exhaustive evaluation of the generated stemmers. By using five mono-lingual collections of different languages, the approach is to compare the stemmers with those commonly used by the operational or experimental IR systems. Statistical testing is applied to the evaluation measures to assess the significance of the differences between different stemmers. The evaluation gives clues about the average performance over a dozen queries and hundreds thousand documents for each collection. An analysis and observation carried out using the queries as statistical units showed the largest differences between stemmers performance thus revealing the factors that affect retrieval performance.

The paper is structured as follows. Section 2 describes the previous contributions to the research on stemming, whereas our contribution is described at an intuitive level in Section 3. HMMs are explained in Section 4

at tutorial level to make the description of the methodology clearer in Section 5. The results reported in Section 7 are of the experiments carried out using standard test collections for five different languages, as illustrated in Section 6. The findings and our reflections are discussed in Section 8.

2 Previous Contributions

The research on automatic tools for reducing a word to its stem dates back to the Sixties when the early stemmers were proposed and experiments were reported to assess their impact on retrieval effectiveness (Frakes, 1992). Since then it has been clear that the degree to which stemming is effective depends on the collection and specifically on the vocabulary used by the IR system. Furthermore the early research showed that the reliability of the results on stemming effectiveness depend on the quality of the experiments. A rigorous analysis requires the use of large test collections and the performance of deep analysis of the retrieval results in terms of statistical significance testing and analysis at query-level (Hull, 1996).

The past literature reported two main approaches to stemming. An approach is strongly based on linguistic sources and methods, like lexicons or rules aiming at detecting word derivations. The other approach makes use of statistical methods, such as information theoretic measures or co-occurrence measures. Linguistic knowledge-based stemming algorithms have been often used to implement systems and to carry out experiments while statistical stemming algorithms have been less frequently tested, maybe because of the little clear advantage in terms of effectiveness. The "S" stemmer, which is a stemmer that reduces plural words to singular words, is an example of well-known algorithm based on linguistic rules, and in particular on a

suffix removal paradigm. The stemmers proposed by Porter (1980) and by Paice (1990) for English have been implemented and are available at (Porter, 2003) and (O'Neill and Paice, 2004). Another famous stemmer is by Lovins (1968).

Despite its theoretical effectiveness, the performance of stemming has often been debated by researchers. Most of the experiments were carried out by measuring retrieval performance in terms of precision and recall, while few of them considered other parameters, like the degree to which two words which share the stem have the same meaning; and the previous research seldom reported a detailed query-by-query evaluation to understand why stemming performs well or badly. An example of detailed analysis is reported by Hull (1996) for English. When stemming has been evaluated using precision and recall some researchers observed that stemming influences little the overall performance of an IR system. This outcome is quite surprising if compared with the reason why stemming is employed in a retrieval setting, but it may be explained by the relatively simple morphology of English which was the language of the early test documents. This suggests that stemming effectiveness may depend on the language and that the design of a linguistic knowledge-based stemmer for a language might be a useless exercise for some languages, while it might be worth for others.

Harman (1991) reported some experiments conducted to assess the retrieval performance of stemming using Porter, Lovins and the "S" stemmer. She pointed out that none of the tested stemmers significantly improves the overall performance of a system which retrieves documents from three test collections: Medlars, CACM, and Cranfield. However the limited effectiveness of stemming was not the only result of Harman's and it would appear discouraging if it were taken without further analysis. Indeed Harman high-

lights that some queries showed an improvement and some others did not. In particular the number of queries which showed improvements in performance after 10 retrieved documents is nearly balanced by the number of queries which showed degradation in performance. If the top 30 retrieved documents are considered instead, the number of queries which showed improvement is higher than the number of queries which showed degradation. This result suggested that stemming should be used depending on the query and on the number of documents examined. To test the first hypothesis term re-weighting and term selection were added to the basic algorithm. However the result was still negative. The main reason why stemming did not help to improve the performances was the noise added by the stemmed words, despite the modifications made to the basic algorithm. Besides the rigorous experimental methodology, Harman's study was limited by the lack of large test collections, which have been made available from the beginning of the Text REtrieval Conference (TREC) program.

More recent studies were carried out with larger English collections and reported for example by Hull (1996). These studies were conducted using the TREC document sets and showed that stemming does not degrade performance across different collections, and that a stemmer for English could be an effective device. However Hull's experiments showed that stemming is little beneficial when (*i*) queries are long and (*ii*) very few documents are examined. As regards the first case, stemming adds as much more noise as longer the query is since each stemmed query word is matched with one or more variants that might have different meanings. As regards the second case, it should be noted that stemming helps to increase the ranks of the documents that match variants but the documents that exactly match the query word are likely to remain at the top ranks even when stemming is

applied. On the contrary stemming can be effective if queries are short and need to be expanded, or when the user wants to retrieve more relevant documents, then additional matches between queries and documents have to be found. As a consequence a stemmer should be used in a more "intelligent" way: if a query is short or the user looks for more documents, stemming should be automatically performed. It is important noting that these results are valid for English only – each language has its own characteristics and stemming is strongly influenced by these characteristics.

Once when other languages came on the scene of IR, some researchers started to apply linguistic knowledge-based stemming which was proved effective for English to Spanish, French, German, Italian, and so on. Examples of experiments are reported by Savoy (1999) for French, Popovic and Willett (1992) for Slovene, Kraaij and Pohlmann (1996) for Dutch, Braschler and Ripplinger (2004) for German, and Bacchin et al. (2002) for Italian. As stemming is strongly influenced by language characteristics, ad hoc stemmers had to be developed for each language – except for the stemmer by Bacchin et al. (2002). What is important with other languages, like Italian, Arabic or French is that they are more complex than English, and their words have much more variants than English words. Therefore the chance that a document is missed because its words do not match a query word is higher. Moreover the impact of effectiveness of stemming is magnified by the size of the query and by the number of the documents wanted by the end user. A glaring example is Arabic which is a highly inflected language because of the use of infixes other than suffixes and prefixes Larkey et al. (2002). Experiments for Arabic IR are also reported for example by Aljlayl and Frieder (2002) and Abu-Salem et al. (1999) who showed that stemming is always beneficial for retrieval effectiveness. Any method like stemming

that aims to expand the query to capture the missed relevant documents has a high probability to succeed and Arabic IR does benefit a lot from the use of a stemmer.

Statistical methods for natural language processing is a well-established field – for a comprehensive overview on this subject, see (Manning and Schütze, 1999). Similarly to other natural language processing methods, stemming has been addressed using statistical approaches to overcome a number of problems of the linguistic knowledge-based approaches: The main problem is the fact that a stemmer is related to the language for which it is developed. Even though most algorithms were based on linguistic sources and methods, the investigation of statistical stemming algorithms is not really recent, on the contrary it was a research carried out since the early Seventies since the cost of the labor work which is necessary to build a stemmer was recognized as a disadvantage. In recognizing the possibility that statistical methods can perform as effectively as linguistic knowledge-based approaches, Hafer and Weiss (1974) proposed stemmers that attempt to determine word and morpheme boundaries using large vocabularies and then decide the prefix which is the most plausible stem. Another example is given by Adamson and Boreham's research Adamson and Boreham (1974); for each word the number of unique digrams is computed and then the similarity between every pair of words is calculated in terms of the number of common digrams. Similarity values are used to cluster together words with a high number of common digrams. The experiments showed that clustered terms tend to be related; as the words that share a common stem are likely to be related, this approach is useful to automatically stem words without using linguistic rules. More recently Hull (1996) showed that linguistic knowledge-based approaches to stemming seems not to increase retrieval

effectiveness compared to those approaches which employ little linguistic knowledge. Similar conclusions were drawn by Savoy (1993) and Krovetz (1993), which used much more linguistic knowledge such as dictionaries and rules. This finding suggests that statistical methods for stemming can be tailored to different languages and produce good results. However a great deal of caution should be paid because statistical approaches might not work as effectively as those based on linguistic knowledge for some very highly inflected languages like Arabic (Larkey et al., 2002).

Besides being methods to produce good stemmers, statistical methods also help to refine stemmers based on linguistic knowledge: for example when a word is overstemmed and then leads to a stem that corresponds to words which have a different meaning. Following this idea, Xu and Croft (1998) presented useful methodological tools to implement effective stemmers that are independent of the language and employ statistical methods to refine stemmers and then improve retrieval effectiveness. Xu and Croft proposed a corpus-based stemming to refine the equivalence classes built by a stemmer – an equivalence class is formed by all the words that result in a common stem. In their research corpus-based stemming refers to splitting equivalence classes so that words that result in a common stem also result in a common meaning where the degree to which two words have a common meaning is estimated by the extent to which they co-occur within the same document fragments. Xu and Croft's interesting result is that their corpus-based stemming can work effectively even if the initial equivalence classes are formed by a trigram extraction algorithm – two words are in the same trigram equivalence class if they share the first three characters.

Other approaches or applications were recently investigated. Larkey et al. (2002) exploited Xu and Croft's methodology to design a stemmer for

Arabic. The research by Goldsmith (2001) yielded a information theoretic measure to select groups of prefixes and suffixes which co-occur together – this way a derivation (suffix) is associated to its stems (prefixes). Bacchin et al. (2002) exploited the mutual reinforcement relationship between prefixes and suffixes – a stem is a prefix that is followed by derivations, a derivation is a suffix that is preceded by stems. Using an algorithm similar to the procedure presented by Kleinberg (1999), Bacchin et al. were able to compute the scores that measure the degree to which a prefix is a stem and the degree to which a suffix is a derivation. The experiments showed that stemmers based on this mutual reinforcement relationship perform as effectively as Porter's stemmers for different languages and large test collections.

The variety of approaches and results is apparent to the reader. The abundance of evidence is the positive side-effect of the fact that stemming has been a research subject for decades. Most of the approaches and of the experimental results are based on linguistic knowledge which is sufficient to implement stemmers for widely used languages. Little used languages suffer the lack of linguistic resources, which make the development of stemmers harder. Statistical methods would be of help, yet few results for statistical approaches to stemming are available.

However, it is not always true that linguistic approaches requires much effort to develop stemmers; stemmers that normalize feminine and masculine, or plural and singular, like the the S-stemmer, are some examples, but it is also true that these stemmers can be used for a language or very few languages at a time – the S-stemmer cannot be used for Italian because there are no words ending with "s". The HMM-based stemmers proposed in this paper are designed to be used for any inflectional language where there are a significant number of words for which suffix stripping does make sense.

It is believed that additional research on statistical approaches to stemming are necessary to accomplish the design and development of systems that retrieve documents written in one or more out of the hundred languages in the world. Our contribution is proposed towards this goal as explained in the next sections.

3 Our Contribution

The goal of this paper is to describe a methodology to generate automatically stemmers. A statistical approach which allows to model the generation of words, and specifically of stems and derivations has been adopted. It should be clear that (*i*) our research concentrates on the class of inflectional languages whose words can be seen as the concatenation of a prefix and a suffix, and (*ii*) the prefix-stem-suffix structure has been simplified to the prefix-suffix structure, i.e. a stem is the prefix which results from a correct suffix stripping.

In the following, derivation and suffix are used exchangeable way. Even though "derivation" also means a process, it was preferred to keep that term as distinct in order to stress that the suffix-derivation has been extracted from a word which has been correctly stripped – the latter being the process of derivation. The "ambiguity" was wanted to evoke that a suffix correctly stripped from word is the result of a derivation. Our stemming approach must only remove suffixes. We are aware that the middle of a word may change due to the syntax: however, our interest was not in linguistically correct stemming, but in effective stemming from an IR point of view.

The proposed methodology can be described in an intuitive manner as follows: Let us view a perfect stemmer as a machine that produces words

in a language known in advance. The stemmer is called "perfect" because it correctly strips suffixes, i.e. first it emits the stem and then emits the derivation. The machine is almost completely unobservable because the detailed way it works is unknown. Yet three facts are supposed to be known: (*i*) the machine performs a sequence of steps and it generates a letter at each step; (*ii*) the machine first produces the stem and then the derivation, and (*iii*) the language in which the words are generated by the machine is known.

Therefore the machine is assumed to consist of two components: the *prefix-set* component generates the prefix of the word that approximates the stem; the *suffix-set* component generates the suffix of the word that approximates the derivation. A split point occurs when the machine switches from the prefix-component to the suffix-component. The problem of stemming a word then reduces to discover the most probable symbol of the word at which the split point occurs.

As described in Section 5, the most probable sequence depends on both the sub-sequences performed by the prefix-component and the one performed by the suffix-component. This sequence depends on the synergy between the two components and, thanks to this synergy, it is the most probable one even if the sub-sequence performed by the prefix-component, or the one performed by the suffix-component are not the most probable.

In order to model this machine, a class of probabilistic models called Hidden Markov Models (HMMs) can be exploited to model this machine, because they well represent the intuitive view of the stemmer. HMMs are particularly suited to model sequences of observations that depend on the evolution of an unknown, underlying process. Thus the machine is modeled with a suitable HMM which is in turn made of two components. Well-known algorithms are performed in order to have an estimate of system evolution

when prefix and suffix are generated, and to make a decision on how to stem a word. The next two sections present a general overview of HMMs and their application to stemming.

4 Hidden Markov Models

HMMs are powerful tools which have been successfully applied to different research domains, where it has been needed to model or to recognize a sequence of observations. The two well-known applications of HMMs are automatic speech recognition (Rabiner and Juang, 1993), where sequences of spoken utterances are modeled to recognize complete words, and biological sequence analysis (Durbin et al., 2000), where sequences of aminoacids or proteins are modeled for a number of bioinformatics applications. Furthermore, HMMs have been also applied to handwriting recognition (Nishimura et al., 1999), automatic music accompaniment (Orio, 2001), and fault detection (Smyth, 1994).

HMMs are probabilistic finite-state automata, where transitions between states are ruled by probability functions. Transition probabilities are assumed to depend only on the present state and not on past state transitions, that is why HMMs are "Markov" models. The presence of transitions with probability equal to zero defines a topology of the HMM, which limits the number of possible paths across the states. At each transition, the new state emits a symbol with a given probability. Only symbols can be observed, that is why HMMs are "hidden". Also emission probabilities are assumed to depend only on the present state and not on the history of previous state transitions and emissions.

Given a set of N states $Q = \{q_1, q_2, \ldots, q_N\}$, and a sequence of observa-

tions $O = \{o_1, o_2, \ldots, o_T\}$ from time 1 to time T, a HMM λ is completely defined by:

- An initial state S and a final state F.
- A probability distribution for state transitions $A = \{a_{ij}\}$, where $a_{ij} = P[q_j(t+1) \,|\, q_i(t)]$ is the probability to go from state i to state j in a single step; usually also self-transitions are allowed, that is $a_{ii} > 0$. The probability distribution is independent of time t and of previous history.
- a probability distribution for state emissions $B = \{b_j(k)\}$, where $b_j(k) = P[o_t = k \,|\, q_j(t)]$ is the probability of emitting symbol k from state j. The probability distribution is independent of time t and of previous history.

Both A and B undergo the stochastic conditions that at each time step the HMM has to perform a transition and it has to emit a symbol, that is the transition probabilities from state i to all the states, as well as the emission probabilities of state i, must sum to 1.

An example of applications of HMM can be done considering them as generators of words in a given language. The observation space is then the alphabet used in that language. The HMM starts at the initial state and, at each time step, it performs a transition to another state emitting the first letter of the word, then it goes on performing transitions and emitting the following letters, until it reaches the final state. It has to be noted that this approach does not apply to languages that are not based on an alphabet, like Chinese or Korean.

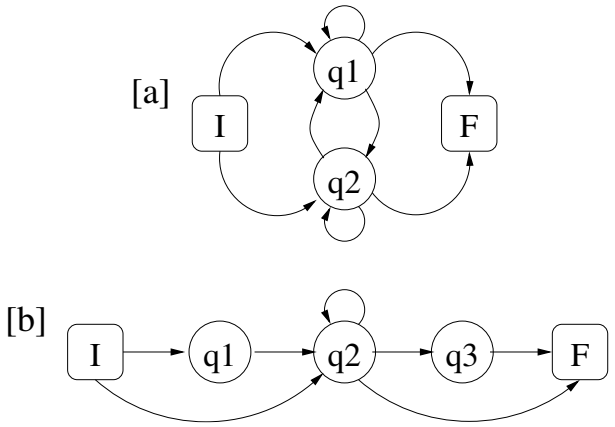

Figure 1: Two simple HMMs for generating sequences of letters.

A simple HMM is depicted in Figure 1[a]. The HMM has four states: the initial state S, the states q_1 and q_2, and the final state F. In this example and in the following, it is assumed that the initial and final state emit respectively the left and right boundary of a word with probability equal to 1, while states q_i emit letters of a known alphabet with different probability distributions. As an example, let us assume that q_1 has a high probability to emit a vowel, while state q_2 has a high probability to emit a consonant (anyway the probability that state q_1 emits a consonant is higher than zero, and vice-versa). According to the statistics on a given language, transition probabilities a_{S1} and a_{S2} correspond to the probabilities that a word begins with a vowel or a consonant respectively, a_{12} is the probability that a vowel is followed by a consonant, $_{a1F}$ is the probability that a word ends with a vowel, and so forth. The HMM starts at S, it performs a transition either to state q_1 or to state q_2, which emit a symbol according to their probability functions, and it performs new transitions until the final state is reached. Though simple, the model is hidden, because the same

word can be generated by different paths among the states; for instance, a word can be generated by continuously staying in state q_1. Intuitively, there are paths that more probably correspond to a given word, due to the fact that states q_1 and q_2 have different probability distributions for transitions and emissions.

The example highlights that an observed sequence, a word, can be considered as the instance of one underlying process, the HMM, whose exact behavior may be unknown. Moreover, it was assumed that the HMMs were completely defined, that is all the probability distributions were set in advance, which is not usually the case. The disclosure of the hidden process and the setting of probability distributions are typical problems of HMM, which can be summarizes as follows:

1. Given a sequence of observations O and a HMM λ, find the sequence of states $\hat{q} = q(1) \cdots q(T)$ that more probably correspond to O; e.g., the sequence that maximizes $P(q(1) \cdots q(T) \,|\, O, \lambda)$.

2. Given a set of observed sequences O_S, find the parameters $\lambda = [A, B]$ that more probably would generate the set of observations; e.g., the parameters that maximize $P(O_S \,|\, \lambda)$.

A solution for the first problem, which is usually addressed as *decoding*, is Viterbi algorithm (Viterb, 1967). Going back to the model depicted in Figure 1[a], decoding allows for finding the path across states q_1 and q_2 that more probably generated a given word. A solution for the second problem, which is usually addressed as *training*, is based on the Expectation-Maximization (EM) algorithm (Baum, 1972) that allows us to automatically set HMM parameters from a sample of observed sequences. A comprehensive

introduction of HMMs and their application to decoding and training can be found in (Rabiner and Juang, 1993; Durbin et al., 2000).

With the aim of describing both decoding and training at an intuitive level, a third HMM for word generation can be considered, which is depicted in Figure 1[b]. The HMM parameters are not known and have to be computed from a set of examples. As it often happens in speech recognition and in other application domains, some of the transition probabilities can be set to zero in order to impose a given topology to the model before training the other distributions, which is the reason why only a subset of transitions are drawn in Figure 1[b]. If the HMM is trained with a set of English words, the distribution for the emissions is expected to be different for the three states, in particular: $b_1(\cdot)$ will depend on the relative frequency of initial letters in English words, for instance $b_1(t) > b_1(e)$ even if letter "e" is much more frequent than letter "t"; $b_2(\cdot)$ will simply approximate the relative frequency of letters in English, because q_2 is the only state that may perform a self-transition and all the letters in the middle of a word have to be emitted by this state; $b_3(\cdot)$ will depend on the relative frequency of final letters, that is $b_3(s)$ will be much higher than all the others due to the way plurals are usually made in English. The only transition probabilities that need to be set are a_{S2}, which will be related to the probability of having the first letter of a word a frequent letter, and a_{22}, which will mainly depend on the average length of English words; all the other transition probabilities follow the axioms of probability theory. Like the other two HMMs of Figure 1, this HMM can generate any sequence of letters, each one with a different probability. It is more interesting for our aims that, for a given words, different paths among the states are possible. The most probable path can be computed through Viterbi algorithm;

for example, the word "tees" can correspond to the four alternative paths $[Sq_2q_2q_2q_2F; Sq_1q_2q_2q_2F; Sq_1q_2q_2q_3F; Sq_2q_2q_2q_3F]$, but the third one has a higher probability of being the one that generated the word.

The examples in Figure 1 have a limitation that has been already mentioned: they can generate with a high probability nonsense words made of repetitions of frequent letters. This limitation can be partially overcome by introducing more constraints on the state transitions, as shown in Section 5 (Figure 2 and Figure 4).

5 Methodology

After introducing HMMs and their applications to word generation, we scope can be extended to word stemming. Our goal is to develop a HMM that can be used to simulate the evolution of the perfect stemmer. To this aim, a HMM is designed where normal states are divided into two disjoint sets: the *prefix-set* and the *suffix-set*. The initial state S and the final state F do not belong to any of the sets. As for the simple HMM presented in the previous section, all the states emit the letters of the alphabet of a given language, while the initial and the final states emit respectively the left and right word boundaries with probability equal to one. As with the perfect stemmer, the prefix-set generates all the possible stems and the suffix-set generates all the possible derivations of a given language. For many Indo-European languages, there are some conditions that can be imposed to the HMM topology according to a number of assumptions that can be made. These conditions are represented in Figure 2 and listed in the following:

- the initial state may perform a transition only to a state in the prefix-set – i.e. a word always starts with a stem;

- the states in the prefix-set may perform a transition to any state in the HMM – i.e. a stem is formed by a sequence of letters possibly followed by a derivation;

- the states in the suffix-set may not perform a transition to states in the prefix-set, while transitions to states in the suffix-set are allowed – i.e. a word is the concatenation of a single stem followed by a single derivation, which is a sequence of letters at the end of the word;

- the final state can be reached from states belonging to both the prefix- and the suffix-set – i.e. a word can be made either by a stem only or by a stem followed by a derivation.

Let us assume that, for a language L, all the parameters of the HMM λ_L are defined, that is for all the N state transition probabilities a_{ij} and emission probabilities $b_i(\cdot)$ are set, and let us apply the HMM in Figure 2 to the generation of a word, the generation of words is carried out as in the examples provided in Section 4.

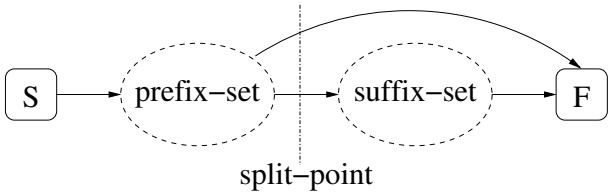

Figure 2: Modeling of word generation as the concatenation of two disjoint sets of states of a HMM.

As usual, the process of word generation is hidden, because the state sequence which corresponds to the generation of a word cannot be observed. Yet it is possible to compute the most probable path with decoding. For

the aims of word stemming, it is not important to discover the complete path across the HMM states, because the only important transition is the one that exits the prefix-set towards either the final state or the suffix-set. We define this transition the *split point* between stem and derivation, as highlighted in Figure 2. Hence, stemming can be carried out through the following steps:

- pick up a word $O = o_1 \cdots o_T$ from a text written in a known language L;
- consider the HMM λ_L for language L and compute the most probable path across its states, $\hat{q} = \text{argmax} P(q(1) \cdots q(T)|O\lambda_L)$;
- find the only transition $\hat{q}$ that goes from a state $q_k(t)$, with i in the prefix-set to a state $q_l(t+1)$, with j in the suffix-set – if there is no such a transition the word is already a stem;
- take the substring from letter 1 to letter t as the stem, and the substring from letter $t+1$ to letter T as the derivation.

In the previous discussion the HMM parameters were all set, but this is an unlikely assumption. The manual computation of transition and emission probabilities for the two sets, as well as the transition probabilities from the prefix-set to the suffix-set, is unfeasible because it would require too much human labor. As for all HMMs, the problem of parameter setting can be overcome by training. The first approach could be to create two training datasets, one with examples of stems, and the other with the examples of derivations, and train separately the two corresponding sets with the classical EM algorithm. But this way the complete information about the

actual words of the language which are formed by the stems and derivations would be lost. A second approach could be to use a single dataset, which contains manually stemmed words, and train the complete HMM using a slight modification of the EM algorithm to take into account that the split point has to correspond to the change from stem and derivation in the given example. Yet, both approaches require extensive human labor to provide a reasonable variety of examples for the training data.

Since our goal is to develop stemmers for different languages minimizing the amount of human labor, an unsupervised training of the HMM is proposed. The dataset used to train the model is a sample of the words of the language for which the stemmer is developed. For instance, words can be taken at random from documents that are available at indexing time. It can be noted that an unsupervised training does not guarantee that the prefix- and the suffix-set maintain their specialization in generating the stem and the derivation of a word. Hence the computation of the split point of the most probable path $q(1) \cdots q(T)$ across the states may not coincide with the correct stemming of a word. With the aim of creating such a relationship, more knowledge about the rules of word generation in European languages can be injected.

First let us assume that, for each language, the number of different derivations is small compared to the number of different stems. The set of derivations can be thought as the set of sequences of letters that can be modeled by chains of states of the HMM. Hence a particular topology of states in the suffix-set is designed, which can be made of a number of chains of different length. An example of the topology of a suffix-set is depicted in Figure 3, where it can be noted that: transition from the prefix-set can be made only to the first state of each chain; states in the chain can

only perform a transition to the next state, thus the path advances with probability one once it enters the chain; the last state of each chain can only perform a transition to the final state. The maximum length of state chains gives the maximum length of derivations that can be modeled by the HMM. Since no assumption is made on the prefix-set, as a first approximation its topology can be created using one or more states that are fully connected.

After the definition of the topology of the suffix-set, the HMM can be trained with EM algorithm on a set of words. It is likely that a sequence of letters that corresponds to a derivation will be frequently present in the training set. For this reason, the EM algorithm will give a high probability of emitting this sequence to the state chains in the suffix-set. For instance, the state in the one-state chain will emit with a high probability the letters that are commonly at the end of the words, the states in the two-state chain will emit with a high probability the bi-grams that are commonly at the end of the words, and so forth.

Preliminary tests with a topology for the suffix-set like the one shown in Figure 3 and simple topologies for the prefix-set gave unsatisfactory results, because they did not consider the relative importance of stems for finding the split point. After unsupervised training a HMM that models only derivations with state chains has a behavior similar to a simple stemmer that strips the most frequent word endings.

In order to overcome this problem, the topology of the prefix-set is made of state chains too. In this way, both stems and derivations are modeled in a similar fashion, as can be seen in Figure 4[a]. The main difference between the two kind of chains is that a state in the prefix-set may perform a self-transition, allowing to model stems of arbitrary length. The minimum length of a chain in the prefix-set gives the minimum length of a word stem.

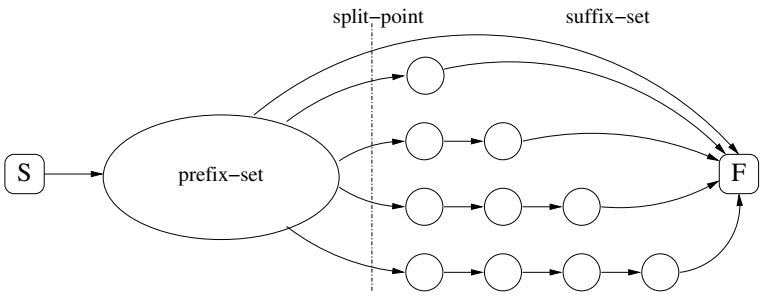

Figure 3: Modeling of the suffix-set as collection of state chains, without constraints for the prefix-set.

When a HMM with this new topology is trained with the EM algorithm, it is expected that the sequences of letters that are frequent in a stem and that are not frequent in a derivation, will be modeled with a high probability by one or more prefix-chains.

The computation of the position of the split point, which can be still carried out applying decoding, depends then both on the modeling of stems and on the modeling of derivations. In particular, there are three different components that contribute to the choice of the most probable path, from which the choice of the split point depends: the probability that the first subsequence of letters of a word is a stem, and it is modeled by the prefix-set; the probability that the second subsequence of letters is a derivation, and it is modeled by the suffix-set; the joint probability of stem and derivation, which is modeled by the transition from the prefix-set to the suffix-set.

The next sections present the results of a number of experiments on the effectiveness of the HMM-based stemming method, using three different topologies that are depicted in Figure 4. In particular, topology `b` simplifies the modeling of chains of the prefix-set by collapsing them in a single chain; topology `c` extends this same approach to the suffix-set.

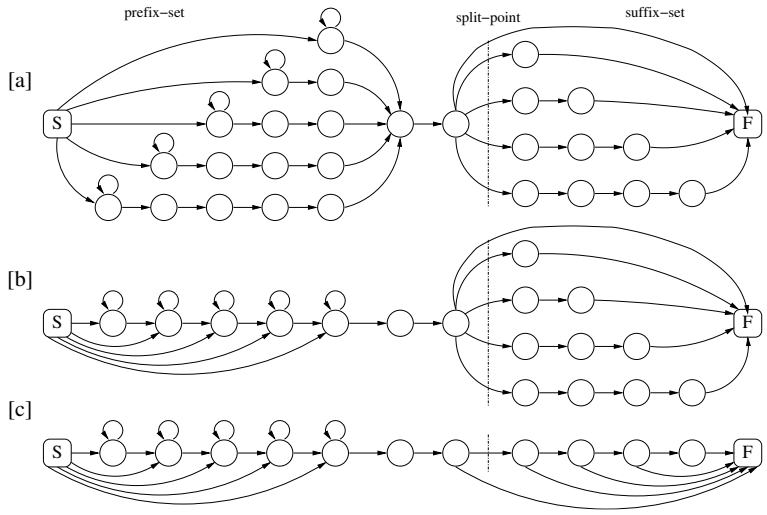

Figure 4: The three topologies tested in the experiments.

6 Design of Experiments

An experimental approach was chosen to assess the effectiveness of the proposed stemmer generation procedure. To obtain reliable results, several experiments using standard large data sets in a controlled laboratory environment were carried out. In our experiments, the Cranfield evaluation model was adopted, from which the evaluation procedure proposed by TREC was derived. The comparison between our experimental results and other research work has been made possible by the choice of the Cranfield methodology for carrying out the experiments and by the use of standard test collections. Because our methodology should be applied to different languages, our experiments were carried out using the data provided by the Cross-Language Evaluation Forum (CLEF), which supplies datasets in a variety of languages. Accordingly to the TREC framework, CLEF provides a different collection for each language and 50 topics for each collection (with the

	No. of docs	Size (MB)	No. of words	No. of distinct words
Dutch	190,604	540	22,543,343	699,611
English	113,005	425	22,105,852	228,727
French	87,191	243	13,136,932	246,689
Italian	108,578	278	15,586,080	345,843
Spanish	215,738	509	28,662,289	378,749

Table 1: A summary of the collections used for the experiments.

exception of English and French, which have 42 and 49 topics each respectively). All the collections are based on news stories appeared, in the same period, on European newspapers. Topics address different subjects, ranging from social studies to international events. The queries were built by using both the title and the description fields. CLEF provides also the corresponding relevance judgments for each topic. The experiments were carried out using data of the CLEF evaluation campaign of 2002, i.e. five collections of documents written in Dutch, English, French, Italian, and Spanish. The size of the collections is moderately large, as summarized in Table 1.

The main aim of our experiments was to assess the impact on retrieval effectiveness of a stemmer generated by a HMM trained with the corpus of documents to which stemming is applied. Moreover, there were other research questions to which an answer was required. The tested hypotheses were:

- H_1: stemming does not significantly affect retrieval performance,
- H_2: stemmers generated by HMMs with simple topologies are as effective as stemmers generated by HMMs with complex topologies,

- H_3: a stemmer generated by a HMM is as effective as a linguistic knowledge-based one: this is our main hypothesis.

These hypotheses are called "null hypotheses" as it is common practice in statistical testing. In fact, the term "null" is used to underline that two treatments are compared and judged as equivalent if the difference between their outcomes is null. In the following, the motivations of each hypothesis and the runs carried out to test it are discussed in detail.

6.1 H_1: Effectiveness of Stemming

The degree to which stemming is an effective device for different languages is an important parameter about the usefulness of the procedure for automatic stemmer generation. Therefore the hypothesis that stemming does not affect retrieval performance was tested. This hypothesis has been already tested in the past for different languages, especially English, but little experimental evidence about other languages is available. Thus the experiments using the CLEF sub-collections were reproduced. This test is important to assess whether the design and application of a stemmer are useful labors.

To test this hypothesis, Porter stemmers, which are available at the Snowball Web site Porter (2003), were run and the retrieval results were compared with those produced without stemming. Porter stemmers, which are a priori linguistic knowledge-based stemmers, are widely used for different languages. If linguistic stemmers did not improve retrieval effectiveness, it is not likely that statistical approaches will give significant improvements either.

6.2 H_2: Effect of Different Topologies

Effective HMM topologies were looked for to optimize the impact on retrieval effectiveness of a stemmer generated by a HMM. Therefore it was investigated if complex topologies would have had a greater impact on retrieval effectiveness than simple topologies. In particular, it was analyzed whether the processes that generate stems and derivations are more complex than a simple sequence of states or not, i.e. if the complexity of substring generation can or should be approximated by the complexity of HMM. If simple topologies were as much effective as complex ones the effectiveness of simple topologies would permit to easily implement stemmers for many languages or collections, because no additional knowledge is needed in terms of states and transitions added to the topology.

To test this hypothesis the three different topologies that are depicted in Figure 4 at different degrees of complexity were tested and compared. The chosen topologies reflect different hypothesis about the mechanisms that generate stems and derivations.

6.3 H_3: HMM vs. Porter Stemmers

Testing that the retrieval effectiveness of an IR system employing a HMM-generated stemming algorithm is equivalent to the effectiveness of a system which employs a linguistic knowledge-based algorithm is our main hypothesis. If it were not rejected the design and implementation of stemmers generated by a HMM would be a direction to follow and the confirmation that statistical approaches to stemming are not inferior to linguistic knowledge-based approaches, at least for IR applications. It is useful noting that the outcome of these hypothesis tests were based on statistical significance test-

ing – no quality control procedures were applied to assess the goodness of stemming from a linguistic point of view.

This hypothesis was tested through the comparison of a HMM generated stemmer with Porter stemmers, all the other indexing and retrieval components being equal. Yet it is not our main aim to improve Porter stemmers, but rather to get close to their performances with HMM-generated stemmers.

Another test has been carried out to highlight the effects of the size of the training set on retrieval effectiveness, by using different proportions of the collection of words that are employed to train the HMM-generated stemmers. As it is known, a reduced amount of training data may cause parameters overfitting – the parameters are optimally estimated for the used data but the model cannot generalize to a larger collection. The robustness to the amount of training data would permit to develop a HMM-generated stemmer using a small or a different word collection and to apply it to other, bigger collections. Moreover, the little sensitiveness of retrieval effectiveness to the training data set size would allow us to increase the document collection, and then the vocabulary without regenerating the stemmer and re-indexing the whole collection.

The runs were classified and named accordingly to the stemmer and to the language of the documents, which is the same as the language of the topics since monolingual retrieval for different languages was the focus of our research. The runs that involved a HMM-generated stemmer are labeled tps-l, the runs that involved a Porter stemmer are labeled `prtr`-l, and the runs without stemming are labeled `no`-l, where:

- t is a tested topology, i.e. `a`, `b`, `c`;

- p is the maximum length of a prefix chain, i.e. `5`, or `6`; note that self-transitions in the prefix-set allow us to generate prefixes of any length;
- s is the maximum length of a suffix chain, i.e. `4`, or `5`;
- l is the language label, i.e. `du` (Dutch), `en` (English), `fr` (French), `it` (Italian), or `sp` (Spanish).

The comparison was made on the basis of the effectiveness of retrieval, and specifically on precision and recall – precision is the proportion of retrieved documents that are relevant, recall is the proportion of relevant documents that are retrieved. To be specific, the evaluation measures of `trec_eval`, Buckley *et al.* (2003) i.e. average precision and recall-precision were used in our experiments – an explanation is reported by Voorhees (2004). The variations between the retrieval precision values obtained by the tested runs were observed. A simple comparison of the variation in percentages among the effectiveness measures of two or more methods would not have given enough information on whether the behaviors of the tested stemmers had been due to their structural nature and had not occurred by chance.

In order to validate the results, a statistical analysis based on significance testing was carried out. The retrieval effectiveness measures produced by each tested method were seen as independent samples. In other words the test topics were considered as statistical units of a sample and the measures were considered as treatments applied on the sample. In testing hypotheses using statistical methods a null hypothesis which means that the compared stemming methods are equally effective is selected for each pair of stemming

methods. The test computes the probability (p-value), under the null hypothesis, of obtaining the observed value. If the p-value is small, then the experiments suggest that the difference had not occurred by chance. The effectiveness of statistical testing is that a small set of topics can be built as a representative sample of the universe of all the possible queries. However, the representativeness of the sample and of each single topic are assumed: If the sample was biased towards a subject or many topics are not realistic, it would not be representative (Hull, 1996).

Statistical tests for paired samples were employed because the two sets of measures were considered as two measures associated to the same sample. In particular, a non-parametric statistical test, i.e. the paired Wilcoxon test was used because there was no evidence about the distribution of the differences in the average precision between stemmers (Mood et al., 1974). The paired Wilcoxon test starts from two paired series (x_i, y_i) of n observed values, one series for each out of the observed variables X, Y to be compared. Then, the differences $d_i = x_i - y_i$ are computed and their absolute values are ranked. For each observation, $r_i = sign(d_i) \cdot rank(|d_i|)$ is computed. Finally, $T = \sum_i r_i \ / \ \sqrt{\sum_i r_i^2}$ is the test statistics, which is approximated by a Normal variable for large n. Being based on ranks, the Wilcoxon test value is not affected by large differences in average precision, which can result in outliers. In comparing the runs, the decision on whether the precision of run X was significantly different from that of run Y was necessary. Tests using both the average precision (Av.P.) and the recall precision (R-P) were carried out. To compare two stemmers that generate X and Y, respectively, the significance tests were built in a way that the null hypothesis H_0 is that no differences exist between the two runs. It is worth noting that different tests unfortunately lead to different outcomes; the Sign test, for instance,

requires more evidence than the Wilcoxon test to reject the null hypothesis.

7 Experimental Results

In order to verify the three hypothesis, a number of tests have been carried out. An experimental information retrieval system, called IRON, was realized by our research group and was used for these experiments. IRON has been built on top of the Lucene 1.2 RC4 library (Lucene, Visited on April, 2004). Lucene implements the vector space model and a $tf \cdot idf$-based weighting scheme (Salton and McGill, 1983). Stop-words have been removed using the stop-lists suggested by the CLEF consortium. Of course, different indexes were generated using different stemmers.

The amount of data used for training may influence the performance of HMMs, due to parameters overfitting. To this end, a number of tests have been carried out using different proportions of words to train the model were compared, namely 100%, 50%, 33%, 25%, and 20% of the words for each of the collections. No significant variations in terms of average or recall precision were observed, for all the tested languages and all the proposed topologies. As the amount of training does not influence the retrieval effectiveness, the runs produced by the HMM-generated stemmers trained with all the words were compared in the remainder of the experiments.

In the presented experiments, large and general collection have been used. Yet, the relationship between the performances of the HMM-based stemmer and the corpus used for training need to be investigated in more detail. It is important to note that, in the proposed approach, the stemmer is trained using the same corpus that will be stemmed, and then it automatically tuned on the specific domain of each corpus. The experimental

results are reported in the following sections.

7.1 H_1: Effectiveness of Stemming

As regards hypothesis H_1, i.e. stemming affects retrieval performance, the results of Porter stemming algorithm were compared with those obtained without applying stemming, i.e. each word has been preserved and no stem has been extracted. For each pair of compared runs, Table 2 reports the average precision, the precision at 5 retrieved documents, the precision at 10 retrieved documents, the number of queries for which the first run was superior (column with label >), not significantly different (=), or inferior (<) in terms of precision, to the second run, and the p-value that is the probability that the null hypothesis is true if it is rejected by the experimenter; when an approach is significant different, the p-value is reported in bold. Table 3 reports similar results about recall precision. As it is well known, a common threshold to decide whether to reject a null hypothesis is a probability of $p < 5.00\%$.

I Run				*II* Run				No. of topics			
Label	Av.P.	P@5	P@10	Label	Av.P.	P@5	P@10	>	=	<	p-value
`no-du`	0.378	0.532	0.462	`prtr-du`	0.401	0.500	0.460	16	7	27	**3.00%**
`no-en`	0.425	0.486	0.376	`prtr-en`	0.445	0.500	0.419	14	3	25	7.87%
`no-fr`	0.338	0.420	0.376	`prtr-fr`	0.397	0.428	0.400	18	2	30	**0.46%**
`no-it`	0.347	0.461	0.406	`prtr-it`	0.377	0.477	0.426	20	0	29	5.18%
`no-sp`	0.419	0.588	0.532	`prtr-sp`	0.458	0.604	0.540	17	0	33	**3.39%**

Table 2: Comparison between `no-`l and `prtr-`l in terms of average precision, precision at 5 and precision at 10.

I Run		*II* Run		No. of topics			
Label	R-P	Label	R-P	$>$	$=$	$<$	p-value
`no-du`	0.385	`prtr-du`	0.418	3	32	15	**0.47%**
`no-en`	0.400	`prtr-en`	0.427	11	14	17	24.20%
`no-fr`	0.335	`prtr-fr`	0.398	7	19	24	**0.45%**
`no-it`	0.353	`prtr-it`	0.389	10	16	23	**2.64%**
`no-sp`	0.409	`prtr-sp`	0.453	12	13	25	**1.78%**

Table 3: Comparison between `prtr-`l and `no-`l in terms of recall precision.

The results show that stemming in most cases improves the retrieval effectiveness because the p-value is very low – an exception is English for which the compared runs are not significantly different. For the Italian experiments the results show as well that the average precision of the results after stemming using Porter algorithm can be considered significantly higher than the precision of the results without stemming, yet with error probability of 5.18% – however, the recall precision is significantly higher because the p-value is2.64%. This outcome is consistent with other experiments carried out in the past, which showed that the impact of stemming depends on the language and that the impact is higher when the language morphology is more complex. As English morphology is rather simple, stemming seems not to significantly affect performance, at least from a statistical point of view.

7.2 H_2: Effect of Different Topologies

As regards hypothesis H_2, i.e. HMMs with simple topologies are as effective as HMMs with more complex topologies, the runs obtained with HMM-generated stemmers with different topologies were compared. In particular,

referring to Figure 4, the most complex topology, labeled with `a`, was compared with the middle complex `b` and with the less complex `c`. The results using `a54` and `b54` topologies for all the languages are summarized in Tables 4 and 5, respectively on the average and the recall precision; as an example, results using `65` topologies are reported for English and Italian.

I Run		*II* Run		No. of topics			
Label	Av.P.	Label	Av.P.	$>$	$=$	$<$	p-value
`a54-du`	0.394	`b54-du`	0.380	31	6	13	**2.51%**
`a54-en`	0.420	`b54-en`	0.403	21	4	17	41.79%
`a54-fr`	0.365	`b54-fr`	0.357	21	1	28	94.42%
`a54-it`	0.340	`b54-it`	0.345	21	1	27	35.76%
`a54-sp`	0.457	`b54-sp`	0.433	25	0	25	20.77%
`a65-en`	0.448	`b65-en`	0.381	30	2	10	**0.02%**
`a65-it`	0.365	`b65-it`	0.364	25	1	23	96.81%

Table 4: Comparison between `a-` and `b-` type topologies in terms of average precision.

The results show that the `a54` topologies are equivalent to the `b54` topologies, because the p-values, i.e. the probabilities that the decision to reject the null hypothesis are wrong, are high. An exception is observed for Dutch since `a` topologies appear to be more effective than `b` topologies in terms of average precision – indeed the probability that the decision to reject the null hypothesis (the topologies are equally effective) is low.

Another particular behavior can be observed comparing topologies `54-en` and `65-en` topologies. It seems that, at least for English, the length of chains in the prefix-set and suffix-set affects the effectiveness of `b` topologies, which are likely to be less effective having a low p-value. For all the other lan-

I Run		*II* Run		No. of topics			
Label	R-P	Label	R-P	$>$	$=$	$<$	p-value
a54-du	0.402	b54-du	0.388	16	25	9	15.85%
a54-en	0.392	b54-en	0.388	6	22	14	15.57%
a54-fr	0.350	b54-fr	0.344	12	23	15	68.18%
a54-it	0.360	b54-it	0.352	11	32	6	23.40%
a54-sp	0.446	b54-sp	0.426	19	18	13	14.16%
a65-en	0.449	b65-en	0.388	14	22	6	**3.75%**
a65-it	0.368	b65-it	0.375	11	24	14	60.31%

Table 5: Comparison between a- and b- type topologies in terms of recall precision.

guages, simple topologies perform as effectively as more complex ones. To confirm the decision not to reject the null hypothesis, the a54 topologies were compared with the simpler c54 topologies; again, results using 65 topologies are reported for English and Italian. The results are summarized in Tables 6 and 7 with regard to the average and the recall precision, respectively. These results show that a54 topologies are also equivalent to the c54 topologies, since the p-values are quite high. The exception is the Italian language for which an even simpler topology improves significantly the retrieval performances in terms of average precision. The same behavior observed comparing results for 54-en and 65-en topologies is maintained also in this case.

In general, the results for hypothesis H_3 show that the proposed approach seems to be robust to the choice of the HMM topology, allowing for simple topologies that requires less computational power. The exception of Dutch in the comparison between a54 and b54, which has been observed

I Run		*II* Run		No. of topics			
Label	Av.P.	Label	Av.P.	$>$	$=$	$<$	p-value
a54-du	0.394	c54-du	0.388	25	3	22	83.37%
a54-en	0.420	c54-en	0.416	22	1	19	96.01%
a54-fr	0.365	c54-fr	0.395	28	1	21	21.50%
a54-it	0.340	c54-it	0.375	15	0	34	**1.05%**
a54-sp	0.457	c54-sp	0.454	21	1	28	60.31%
a65-en	0.448	c65-en	0.422	25	4	13	5.00%
a65-it	0.365	c65-it	0.356	23	0	26	80.26%

Table 6: Comparison between a- and c- type topologies in terms of average precision.

also in other runs, may show that for this language the suffix and the prefix sets need to be symmetrical, hence a simpler modeling of the prefix-set that is not balanced by a similar modeling of the suffix-set may lower the effectiveness of HMM-based stemmers. On the other hand, the exception of Italian in the comparison between a54 and c54 has not been observed in other runs and hence may be due to the particular topic set. Finally, the comparison between topologies for English seems to be related to the number of states in the models. It can be noted that English is less inflected than other languages and, on average, it has a small words length. This may mean that HMMs with longer chains of states are not good models for English, and this effect becomes more significant when they have simplified topologies.

I Run		*II* Run		No. of topics			
Label	R-P	Label	R-P	$>$	$=$	$<$	p-value
a54-du	0.402	c54-du	0.385	17	23	10	16.76%
a54-en	0.392	c54-en	0.397	7	24	11	29.37%
a54-fr	0.350	c54-fr	0.394	12	21	17	18.02%
a54-it	0.360	c54-it	0.380	8	23	18	5.61%
a54-sp	0.446	c54-sp	0.445	12	23	15	79.49%
a65-en	0.449	c65-en	0.404	20	15	7	**0.88%**
a65-it	0.368	c65-it	0.357	16	17	16	68.65%

Table 7: Comparison between a- and c- type topologies in terms of recall precision.

7.3 H_3: HMM vs. Porter Stemmers

As regards hypothesis H_3, i.e. the retrieval effectiveness of an IR system employing a HMM-generated stemming algorithm is equivalent to the effectiveness of a system employing a linguistic knowledge-based algorithm, the runs of HMM-generated stemmers were compared with the runs obtained by Porter stemmers. In Tables 8 and 9, the comparison between some "candidate" topologies and Porter stemmer is summarized for each language. The choice of the topology has been carried out after preliminary tests on retrieval effectiveness using a number of different topologies. The ones that perform better are used as our "candidates" to be compared to Porter stemmers. The results show that the HMM-generated stemmers are almost always equivalent to the corresponding Porter stemmer as far as the comparison is carried out in terms of retrieval precision. One exception occurred: for Dutch, the HMM-generated stemmer is less effective than the corresponding Porter stemmer in terms of recall precision, yet the

I Run		*II* Run		No. of topics			
Label	Av.P.	Label	Av.P.	$>$	$=$	$<$	p-value
a54-du	0.394	prtr-du	0.401	20	4	26	27.57%
a65-en	0.448	prtr-en	0.445	21	3	18	63.84%
c54-fr	0.395	prtr-fr	0.397	17	1	32	24.60%
c54-it	0.409	prtr-it	0.377	22	1	27	70.39%
a54-sp	0.457	prtr-sp	0.458	22	0	28	74.14%

Table 8: Comparison between the "candidate" HMM-generated stemmers and the Porter stemmers in terms of average precision.

absolute levels of precision are comparable. The results for Dutch can be explained considering that Dutch has a considerable presence of compound words, which are not modeled by the current HMM-based stemmers.

The average performance figures gave the reader a lot of information about the effectiveness of the tested HMM-generated stemmer especially if compared with a Porter's stemmer. The main finding is that the hypothesis that the compared stemmers are equally effective cannot be rejected for some European language as English, Spanish, French, and Italian, with the exception of Dutch (see Table 9).

7.4 Analysis of the Results

The standard way to proceed to analyze experimental retrieval results in a laboratory environment has been exemplified in Section 6: A test set of topics was submitted to an experimental system that produces as many rankings as the methods tested; recall-precision tables are computed for each topic and averaged over all the topics. The statistical tests employed often assumes that the error variance is constant across the topics. Since the

I Run		*II* Run		No. of topics			
Label	R-P	Label	R-P	$>$	$=$	$<$	p-value
`a54-du`	0.402	`prtr-du`	0.418	6	28	16	**3.66%**
`a65-en`	0.449	`prtr-en`	0.427	16	15	11	36.81%
`c54-fr`	0.394	`prtr-fr`	0.398	10	27	13	42.37%
`c54-it`	0.380	`prtr-it`	0.389	13	23	13	79.49%
`a54-sp`	0.446	`prtr-sp`	0.453	16	16	18	62.49%

Table 9: Comparison between the "candidate" HMM-generated stemmers and Porter's stemmers in terms of recall precision.

number of relevant documents is highly variable across the topics, a highly variable error variance is likely to occur. Indeed the rank of the retrieved relevant documents and then recall-precision values change dramatically across the tested methods with very few relevant documents. Furthermore mean precision values hide the information which can be provided by single topics to explain why a method failed or succeeded.

What is useful, if not necessary, is an accurate analysis of the topic-by-topic results. For each topic the difference between two methods should be computed and the ranks of the retrieved relevant documents should be compared; then for each retrieved relevant document the explanation of the reason why it was top-ranked or not should be given. However this per-query analysis would require a lot of human effort if performed individually, and it may be useless because many differences are likely to be small or determined by changes of the rank of very few relevant documents.

The approach followed in this research was to identify those topics which clearly demonstrated a significant difference between the tested methods as they should be the most informative to draw conclusions about the observed

total difference. Moreover some large differences could explain a large proportion of the total difference. This way useful information on how to modify a stemmer and significantly improve its performance would be provided. On the contrary the identification of factors affecting the performance of a stemmer would be little useful if drawn from the analysis of those topics for which the difference between the compared methods was small.

An evaluation has been carried out on the differences in terms of average precision between our "candidates" HMM-generated stemmers and Porter stemmers, for Italian and English (see Table 9). The choice of these two languages depends only on the fact that they are sufficiently known by the authors. Given the considerations on factors that can affect a detailed evaluation, the topics with performance variations above 20% and with more than 20 retrieved relevant documents were considered for Italian (15% and 15 for English, respectively). The choice of the threshold depends on the way that a user is likely to perceive a difference between the two methods in terms of retrieval results. The two thresholds are slightly different in order to have a similar number of topics between the two languages.

Table 10 shows for Italian and English: The first column includes the topic identifiers; the second column includes the percentage variation of the retrieval precision by using the HMM-generated stemmer with respect to the retrieval precision by using of Porter's stemmer, a negative value means that Porter stemmer performed less effectively; the third column includes the number of retrieved relevant documents by the method that performed best.

The analysis of the chosen topics can help understand when and why a HMM-generated stemmer performed less effectively than a Porter stemmer – this is a necessary condition to improve stemming technology (Hull, 1996).

Italian		
Topic Id.	Variation	Rel.Ret.
99	165.44%	35
115	59.46%	26
108	47.77%	26
125	20,77%	36
140	-43,11%	49
135	-35,01%	72
131	-24,39%	32

English		
Topic Id.	Variation	Rel.Ret.
107	203,44%	28
115	34,99%	24
122	18,54%	24
131	15,15%	37
108	-18,81%	17
104	-15,57%	18

Table 10: Italian (left) and English (right) topics that have been chosen for this evaluation; the percentage variation and the number of retrieved relevant documents is reported for each topic.

The detailed analysis of the results was omitted in order to not distract the reader from unnecessary data – a summary was preferred and the main observations drawn from some topics have been reported below. It should be remembered that the description words were used for building the queries, so the tables below include some non-title words.

It was observed, indeed, that a stemmer causes a drop in performance if it tends to overstem words. This phenomenon can be observed, for example, with Italian Topic No. 99 – *Spionaggio: il caso Ames* (The Ames espionage case). The word *impatto* (impact) was overstemmed by the HMM-generated stemmer thus matching out-of-context words such as *impatrio* (to impatriate) and impatiens (already in English in the Italian collection). The word *Russia* was stemmed to *rus* and then a lot of extraneous words were matched, like the proper names *Rusconi* or *Rusic*, other nouns such as *rus-*

tico (country-style) or *ruspa* (scraper). Verbs such as *trovino* matched many other derivations of the verb *trovare* (to find). Yet these verbs might be removed in a laboratory environment since they and other words frequently occurred in the test topics.

The previous consideration could be obvious. Yet overstemming is neither a sufficient nor necessary condition for a performance drop. The distribution of stems across relevant and non-relevant documents is a key factor. If the stem of an overstemmed word is more concentrated within relevant documents than within non-relevant ones, performances do not drop dramatically. For instance, in the case of Italian Topic No. 108 – *Secessione dello Yemen del Sud* (Southern Yemen Secession), the word *sociali* (social) was overstemmed by the HMM-generated stemmer because the stem *soc* matched a lot of extraneous and non-Italian words. It is likely that the overstemming of *furono* (they were) by Porter stemmer degraded effectiveness at a less extent than the overstemming of *sociali* did thanks to the distribution of the stem *fur* within relevant and non-relevant documents.

Something different happened with Topic No. 115 *Statistiche sul divorzio* (Divorce Statistics). Only the word *percentuale* (percentage) was stemmed to *percent* by the HMM-stemmer. However *percent* is the stem of *percentual* and *percentuali* only, which have the same meaning of *percentuale*. As there were few differences between Porter's stemming and HMM stemming results, the list of the retrieved relevant documents were looked. The use of Porter stemmer lead to the retrieval of more relevant documents. At the same time, Porter's stemmer produced a better ranking within the top thirty documents despite *forniscono* was correctly stemmed to *forn*, which is the root of the verb *fornire* (to supply), but also other words with a totally different meaning, like *forno* (oven), were stemmed to the same root. This suggests that

stemming is not always a crucial factor in retrieval performance.

Topic No. 140 *Telefoni cellulari* (Mobile phones) seemed to contradict the fact over-stemming would degrade performance. Although the HMM-generated stemmer reduced words at a larger degree than Porter the stems produced matched many relevant words and few extraneous words. Indeed the list of retrieved documents showed that the use of the HMM-generated stemmer allowed the system to retrieve more relevant documents than Porter, and in particular at middle ranks (30-100).

The hypothesis that too short roots are detriment of retrieval effectiveness was witnessed by the English Topic No. 122 *North American car industry*, which include three-character stems produced by the HMM-generated stemmer. As the English language in general include shorter words than Italian, the application of the HMM-generated stemmer can produce much more too short stems than those produced from Italian words.

Despite over- or under-stemming, and some inconsistent outcomes, HMM-based stemmers revealed a surprising non-negative role in retrieval effectiveness by thus suggesting the idea that stemming errors are not always crucial to retrieve relevant documents. However, the comparable on average effectiveness of the HMM-based stemmers has interesting implications, the most important being the possibility to deploy HMM-based stemmers within IR systems which manage collections of documents written in different languages.

8 Conclusions and Future Work

Summarizing, our experiments show that is possible to model the process of word generation as the pipeline of two sub-processes – the sub-process

generating prefixes and the other one generating suffixes – and that HMMs are effective tools for this task. The identification of the switch between the two sub-processes is equivalent to extracting the stem of a word. HMM training basically consists of probability estimation on the basis of a word collection. Decoding can be efficiently performed at search time for the input query words and also for words not used at training time. At first sight, the retrieval effectiveness might be dependent on the HMM topology but, accordingly to our experiments, a simple topology serves for this task as effectively as more complex ones. Our conclusion is that the retrieval effectiveness of the tested HMM-generated stemmers is equivalent to that of Porter stemmers. Yet some comments are useful.

Comparing the experimental results, the use of HMM-generated stemmers can improve the retrieval effectiveness of an IR system. Of course, this improvement was not expected to be higher than the one obtained with a stemmer based on prior linguistic knowledge, which can take into account special rules for the creation of word derivations. Moreover, our method do not take into account any peculiarities of a language like phonetic variants (i.e. "loveable" and "lovable") or compounding (i.e. "lovesong").

Similarly to many other statistical approaches, our method computes maximum likelihood estimation of the probabilities and thus performances depends also on the frequency of letters and substrings. Nevertheless, finding the split point between prefix and suffix is not merely lopping off the most frequent word endings. Indeed, also prefixes are modeled by a number of chains and the most probable path depends on the synergy between the modeling of prefixes and suffixes. The strength of the proposed method is that it does not require human labor, for the identification of morphological rules or for the creation of a manually stemmed training set. The develop-

ment of a stemmer for a new language can then be carried out straightforwardly, given the assumption that stemming can be performed with a suffix removal approach.

Since training is performed using the EM algorithm, the computational complexity for the generation of a stemmer is $O(l \times N^2 \times i)$, where l is the overall number of letters in the words in the training set, N is the number of states in the HMM, and i is the number of iteration required by the EM algorithm – in our experiments, trainings phase stop after approximately 30 iterations. Even if the generation of a stemmer has to be performed at indexing time, it might be useful to reduce the computational load by using simple topologies, which have a smaller number of states N and are as effective as more complex ones, according to hypothesis H_2. The computational load at query time is not an issue, because stemming is performed by Viterbi decoding, which has a computational complexity linear with the number of letters of the words of the query like many other stemmers.

The evaluation highlighted that HMM-based stemmers have the tendency to overstem words, at least compared with Porter algorithms. In particular, HMM-based stemmers tend to remove suffixes also when words are short, even if the proposed topologies have the constraints that stems cannot be shorter than three letters. The query-by-query evaluation showed that overstemming does not necessarily degrade retrieval effectiveness if stems are still distributed in a way relevant documents are discriminated from non-relevant ones.

The training of a HMM used to implement a stemmed depends on the corpus. This fact seems a weakness, yet it is also a strength because the training comes from the collection for which the stemmer is developed and a stemmer can be built of every collection in a few hours. This way a

stemmer for a collection might stem domain-specific words more effectively than a stemmer for another collection. It should be also noted that the experiments were run using large, general collections, and that one instance of each word is used to train the HMM. To make a stemmer more sensitive to the domain, the frequency of occurrence might be taken into account at probability estimation time, but these issues are left to the future work.

References

H. Abu-Salem, M. Al-Omari, and M.W. Evens. Stemming methodologies over individual query words for an arabic information retrieval system. *Journal of the American Society for Information Science and Technology*, 50(6):524–529, 1999.

G. Adamson and J. Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10(7-8):253–260, 1974.

M. Aljlayl and O. Frieder. On Arabic search: Improving the retrieval effectiveness via a light stemming approach. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 340–347, McLean, VA, USA, 2002.

J. Allan and W.B. Croft, editors. *Challenges in information retrieval and language modeling*, Report of a Workshop held at the Center for Intelligent Information Retrieval on September, 2002, University of Massachusetts, Amherst, MA, Published on April, 24th 2003. `http://ciir.cs.umass.edu/irchallenges/`.

M. Bacchin, N. Ferro, and M. Melucci. The effectiveness of a graph-based algorithm for stemming. In *Proceedings of the Internation Conference on Asian Digital Libraries*, pages 117–128, Singapore, 2002.

L.E. Baum. An inequality and associated maximization technique in statistical estimation for probablistic functions of markov processes. *Inequalities*, 3:1–8, 1972.

M. Braschler and B. Ripplinger. How effective is stemming and decompounding for German text retrieval. *Information Retrieval*, 7(3-4):291–306, 2004.

C. Buckley *et al.* The `trec_eval` evaluation package. `ftp://ftp.cs.cornell.edu/pub/smart/`, 2003. Visited at May, 2003.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 2000.

E.N. Efthimiadis. Query expansion. In M.E. Williams, editor, *Annual Review of Information Science and Technology (ARIST)*, volume 31, chapter 4, pages 121–185. Information Today for the American Society for Information Science, Medford, NJ, 1996.

W.B. Frakes. Stemming algorithms. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: data structures and algorithms.*, chapter 8. Prentice Hall, Englewood Cliffs, NJ, 1992.

J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):154–198, 2001.

M. Hafer and S. Weiss. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385, 1974.

D. Harman. How effective is suffixing. *Journal of the American Society for Information Science*, 42(1):7–15, 1991.

D.A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996.

J. Kleinberg. Authorative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.

W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 40–48, Zürich, Switzerland, 1996.

R. Krovetz. Viewing morphology as an inference process,. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1–203, 1993.

L.S. Larkey, L. Ballesteros, and M.E. Connell. Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 275–282, Tampere, Finland, 2002.

J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

Lucene. The Lucene search engine. `http://jakarta.apache.org/lucene/docs/index.html`, Visited on April, 2004.

C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. The MIT Press, 1999.

M. Melucci and N. Orio. A novel method for stemmer generation based on hidden Markov models. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 131–138, New Orleans, LA, USA, November 2003.

A.M. Mood, F.A. Graybill, and D.C. Boes. *Introduction to the theory of statistics*. McGraw-Hill, 1974.

H. Nishimura, M. Kobayashi, M. Maruyama, and Y. Nakano. Off-line character recognition using HMM by multiple directional feature extraction and voting with bagging algorithm. In *Proceedings of the 5th International Conference on Document Analysis and Recognition (ICDAR-99)*, pages 49–52, 1999.

C. O'Neill and C. Paice. The Lancaster stemming algorithm. `http://www.comp.lancs.ac.uk/computing/research/stemming/`, March 2004.

N. Orio. An automatic accompanist based on hidden Markov models. In F. Esposito, editor, *Advances in Artificial Intelligence (AI*IA)*, pages 64–70, 2001.

C.D. Paice. Another stemmer. *SIGIR Forum*, 24:56–61, 1990.

M. Popovic and P. Willett. The effectiveness of stemming for natural language access to Slovene textual data. *Journal of the American Society for Information Science*, 43(5):384–390, 1992.

M. Porter. Snowball. `http://www.snowball.tartarus.org/`, May 2003.

M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

L. Rabiner and B.H. Juang. *Fundamentals of speech recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

G. Salton. *Automatic text processing*. Addison-Wesley, 1989.

G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, NY, 1983.

J. Savoy. Effectiveness of information retrieval systems used in a hypertext environment. *Hypermedia*, 5(1):23–46, 1993.

J. Savoy. A stemming procedure and stopword list for general French corpora. *Journal of the American Society for Information Science*, 50(10): 944–952, 1999.

P. Smyth. Hidden Markov models for fault detection in dynamic systems. *Pattern Recognition*, 27(1):149–164, 1994.

A. Viterb. Error bounds for convolutional codes and an asymptotically decoding algorithm. *IEEE Transactions on Knowledge and Data Engineering*, IT(13):260–269, 1967.

E. Voorhees. Overview of TREC 2004. In *Proceedings of the Text Retrieval Conference (TREC)*, NIST Special Publication: SP 500-261, Gaithersburg, MD, USA, 2004. National Institute of Standards and Technology (NIST).

J. Xu and W.B. Croft. Corpus-based stemming using cooccurrence of word

variants. *ACM Transactions on Information Systems*, 16(1):61–81, January 1998.