HPS: High Precision Stemmer

Tomáš Brychcín[a,b,*], Miloslav Konopík[a,b]

[a] *Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic*

[b] *NTIS – New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic*

Abstract

Research into unsupervised ways of stemming has resulted, in the past few years, in the development of methods that are reliable and perform well. Our approach further shifts the boundaries of the state of the art by providing more accurate stemming results. The idea of the approach consists in building a stemmer in two stages. In the first stage, a stemming algorithm based upon clustering, which exploits the lexical and semantic information of words, is used to prepare large-scale training data for the second-stage algorithm. The second-stage algorithm uses a maximum entropy classifier. The stemming-specific features help the classifier decide when and how to stem a particular word.

In our research, we have pursued the goal of creating a multi-purpose stemming tool. Its design opens up possibilities of solving non-traditional tasks such as approximating lemmas or improving language modeling. However, we still aim at very good results in the traditional task of information retrieval. The conducted tests reveal exceptional performance in all the above mentioned tasks. Our stemming method is compared with three state-of-the-art statistical algorithms and one rule-based algorithm. We used corpora in the Czech, Slovak, Polish, Hungarian, Spanish and English languages. In the tests, our algorithm excels in stemming previously unseen words (the words that are not present in the training set). Moreover, it was discovered that our approach demands very little text data for training when compared with competing unsupervised algorithms.

Keywords: stemming, morphology, inflection, maximum entropy, maximum mutual information, language modeling, information retrieval

1. Introduction

Word stemming tasks are among the basic preprocessing techniques in NLP (Natural Language Processing). IR (Information Retrieval) tasks, MT (Machine Translation) systems, LM (Language Modeling), and many other applications in NLP benefit from reducing the number of word forms by applying a stemming method. Current word stemming methods [Goldsmith, 2001; Majumder et al., 2007; Paik et al., 2011a] are usually more task oriented and do not necessarily respect linguistic notations. They try to find different stems for words with different semantics and the same stems for words with the same semantics but a different function in the sentence. The distinction between the same and different semantics is given by the particular task to which the stemmer is being applied. For example, the words *friend* and *friendly* may be considered semantically equal for information retrieval but not for machine translation. The stemming results are often arbitrary parts of the input words (e.g., *dur* from *durable*) rather than linguistically correct morphological units – e.g., morphemes (such as *friend* from *friendship*). Creating correct morphological units would involve extra effort and might introduce errors, but having them is not always necessary. For

*Corresponding author

Email addresses: brychcin@kiv.zcu.cz (Tomáš Brychcín), konopik@kiv.zcu.cz (Miloslav Konopík)

the above mentioned tasks, it is sufficient to have a stem represented by a sequence of characters extracted from an input word that distinguishes meanings.

A large class of languages (in the linguistic typology, these languages are called *synthetic* languages) tends to modify the basic word form by adding prefixes and suffixes according to the function of the word in a sentence. These word forms usually share the same basic meaning. Many stemming methods strip off these affixes. However, such a task is rather complicated in many cases, as illustrated by the following examples. The pairs of English words A) *shin* and *shining*, B) *spar* and *sparing* and C) *speak* and *speaking* are lexically similar and differ in having / not having the suffix *ing*. The first and second pairs (A, B) consist of semantically different words, whereas words from the third pair (C) differ only in their verb tense. Stripping off the suffixes in A and B would be a stemming mistake, whereas in C it is a correct action. The same example can be made of the word pairs *blue* and *blues* or *word* and *words*. Again, the suffix *s* can mean two completely different words or just a different grammatical number. These examples illustrate that stemming algorithms cannot just strip off a known affix. It is instead necessary to decide in which case the affix can or cannot be stripped off.

In this article we describe a novel approach to stemming. The distinguishing property of the approach is the ability to provide very accurate stems (high precision) at the cost of a small decrease in the recall rate. This property constitutes the basis for the name of our stemmer: the *High Precision Stemmer* (HPS), where the word *precision* comes from preferring precision over recall. Our method works in a fully unsupervised manner (it does not require labeled data or any knowledge about the language itself) and is multilingual. In order to prove the multilingual property, we experiment with four different language families: Slavic, Uralic, Romance and Germanic. The Slavic languages are represented by Czech, Slovak, and Polish; the Uralic languages by Hungarian, the Romance languages by Spanish, and the Germanic languages are represented by English.

The rest of this article is organized as follows. In Section 2, we clarify some terms that are used throughout the article. Section 3 introduces state-of-the-art methods for stemming. Then we describe our algorithm in Section 4. We explain our motivation and the principles of the algorithm. In Section 5, we show the results of detailed performance tests of our method by comparing it with the morphologically annotated data and testing it on the IR and in language modeling tasks. The last sections, Section 6 and Section 7, are devoted to discussing, introducing open issues, describing avenues for further work, and drawing conclusions.

2. Definitions

Lexeme, lemma, stem: Throughout the article, we employ the terms *lexeme*, *lemma* and *stem*. To clarify these terms, we provide short definitions. *Lexeme* is a virtual dictionary entry of a given word. All inflectional variants of each word share the same lexeme. *Lemma* is one selected inflectional variant that is used to designate the lexeme. Lemmas have standardized morphological properties: it usually means that lemmas are words in the singular (nouns), masculine (nouns, adjectives), nominative (nouns), or infinitive (verbs). For example, the words *speak*, *speaks*, *speaking* share the same lexeme, which is designated by the lemma *to speak*.

The term *stem* has different meanings in linguistic sources. In some of them, a stem is defined as a part of a word with meaning that can create new words through different linguistic processes. According to [Huddleston, 1988], stems can be combined together by a process called *compounding* (e.g., *black-bird* or *day-dream*) or affixes can be attached by a process called *affixation* (e.g., *dur*-able). The stems *black* or *bird* are called *free stems* because they are words by them self. The stem *dur* is called a *bound stem* since it needs an affix to form a word. In other sources ([Kroeger, 2005]), a stem is the common part of the word that stays the same for all the inflectional variants (e.g., *daydream*-s, *daydream*-ing).

In this article, we use a third definition that was outlined in the introduction. We are interested in a common part of a word that carries the same meaning for all its lexical variants[1]. Our definition of meaning

[1]Lexical variants of words or lexically related words are the words that lexically resemble or lexically overlap one another: e.g., *dur*-able, *dur*-ation.

is task dependent (e.g., for information retrieval it is the part of the word that defines what a user looks for).

Stemming errors: We distinguish two basic types of stemming errors: *understemming* and *overstemming*. Understemming means that the word is not shortened enough and the resulting stem does not cover all variants of the word. Overstemming has the opposite meaning: the word is shortened too much and the resulting stem covers more lexemes.

Light and aggressive stemmers: Stemmers can be divided into *light* and *aggressive* stemmers. The light stemmers prefer precision over recall and are likely to understem the words. In disputable examples, the word is rather left intact instead of creating too short a stem (for example, reducing the words *durable* and *duration* to *dura*). The aggressive stemmers work the other way round. In disputable examples, their stemming is performed even at the risk of creating too short a stem (overstemming).

Inflectional morphology vs. derivation (linguistic process): As we noted in the Introduction, stemming tools usually work with affixes. We distinguish two main types of affixes, given their effect on words: *inflectional* and *derivational* affixes. An example for English is the following: *-s*, *-ed*, *-ing* forming the words work-*s*, work-*ed*, work-*ing* are inflectional affixes and *-able*, *-less*, *-ful*, *-ly*, *-ness* forming blame-*able*, blame-*less*, blame-*ful* blame-*less*-*ly*, blame-*less*-*ness* are derivational affixes. The example clearly illustrates the different roles of each affix. Inflectional affixes form morphological variants of a given word with the lemma staying the same. Derivational affixes create new words with more or less related meaning. We can also clearly see that removing derivational affixes can be sometimes risky. The words blame-*less* and blame-*ful* share the meaning, however, they are antonyms. The question of whether stemmers should or should not remove derivational affixes is difficult and we will address it in our experiments (see Section 5.4) and in the discussion (Section 6).

Stemming vs. lemmatization: Stemming and lemmatization are two related fields. In NLP, both the methods are often used for similar purposes: to reduce the number of word forms in a text. The fundamental difference is the different kind of results. The product of lemmatization is a lemma which is a valid linguistic unit. In contrast, the stem, as defined in the Introduction, is mostly task-oriented in NLP. Moreover, some stemmers also remove derivational affixes, whereas lemmatizers are restricted to inflections only. However, both stems and lemmas are intended for reducing the size of the dictionary. Stemming and lemmatization thus can replace one another in some cases. Stemming cannot be used if the output is requested to be a valid word form of a language, just as lemmatization can be too weak for some tasks (e.g., *vague* and *vaguely* have different lemmas – in this case, the lemmas are the same as the words: *vague*, *vaguely* – which may be a problem for IR). Another difference is that there are currently no means for training a lemmatizer in an unsupervised way: a labeled training corpus or set of manually created rules is needed. Moreover, stemmers are usually more semantically oriented: aggressive stemmers tend to join together semantically related lexemes. For example, *runner* and *running* may have one stem, *run*, but these would have two lemmas and two lexemes. A different example is *familiar* and *unfamiliar*. These would have one lemma (one lexeme) but usually two stems since the words have contradictory meaning.

3. State of the art

The current state-of-the-art stemming algorithms usually belong to one of two basic categories: the *rule-based* stemmers and the *statistical* ones. Rule-based stemmers attempt to transform the word form to its base form by using a set of language-specific rules created manually by linguists. The statistical stemmers usually use unsupervised training to estimate the parameters of a stemming model. The basic qualitative difference is that the rule-based stemmers tend to be better at applying rather complex linguistic rules. They are not limited to stripping off affixes, but they can also change the entire word when necessary. Creating such rules is, however, very time demanding[2] and preferably requires a linguistic expert or at least a speaker of that particular language. On the other hand, statistical stemmers benefit from a large database

[2]There are some languages (artificial or very regular) where a short list of simple rules is sufficient. This is, however, not the case for all tested languages.

of automatically learned rules or parameters. Due to their principle of processing large quantities of texts, they can capture less frequent and less obvious cases. Introducing a new language or a new dialect of a language is straightforward provided that the new language meets the assumptions[3] that were made for the given statistical stemmer. However, they fail when the particular linguistic process is outside the scope of the statistical model (e.g., statistical stemmers would fail for words such as *sing* and *sang*, *foot* and *feet*, etc., although they are quite frequent in English).

3.1. Rule-based approaches

The first published stemming algorithm ever is Lovin's stemmer [Lovins, 1968], which was designed for stemming English. It needs only two steps for stemming a word according to predefined endings and transformation rules. This makes the algorithm very simple and very fast.

Another popular algorithm called Porter's stemmer [Porter, 1980] evolved into a whole stemming framework called *Snowball*. Snowball is a string-handling programming language developed by M. F. Porter. Stemming algorithms can be easily defined in this language. In addition, ANSI C or Java programs can be automatically generated. The framework is briefly described at `http://snowball.tartarus.org`, together with stemmers for several languages.

In [Dolamic and Savoy, 2009], two rule-based stemmers (light and aggressive) for the Czech language are introduced[4]. The aggressive stemmer exhibits slightly better results in IR than the light one. The authors present a MAP (mean average precision) improvement of about 46% by using the aggressive stemmer, and 42% by using the light stemmer in IR systems, compared with no stemming.

In [Savoy, 2008], the investigation of information retrieval in Hungarian is presented. The Hungarian language is characterized by a complex morphology, thus two rule-based stemmers (light and aggressive) are used to improve IR. When compared to an IR scheme without stemming, the light stemmer was able to improve MAP by about 53% on average, and the aggressive stemmer, by about 67% on average.

3.2. Statistical approaches

Many studies of the unsupervised learning of the morphology of a language have been published. An outstanding and exhaustive survey can be found in [Hammarström and Borin, 2011], which provides a description and comparison of the different approaches that deal with morphology at different levels of detail. In terms of that article, our approach belongs to the *same-stem decisions* level, which is defined as follows: *Given two words, decide if they are affixations of the same lexeme.*

The authors in [Xu and Croft, 1998] present a method that uses the word form co-occurrences in a corpus to upgrade or create a stemmer. Their work is based on the assumption that word variants (inflected forms of the same word) should occur close to each other (perhaps within a 100 word text window). To model this fact, a variant of expected mutual information is used. The initial distribution of equivalence classes given by some aggressive stemmer (such as Porter's) is refined using the co-occurrence statistics. According to experiments, the authors show that this additional information enhances the quality of a stemming algorithm.

An interesting method for unsupervised stemming was described in [Goldsmith, 2001]. This method is based on the principle of MDL (Minimum Description Length). The algorithm tries to find the optimal breakpoint for each word. Each instance of a given word in a corpus uses the same breakpoint, which splits this word into stem and suffix. The model for the optimal distribution of breakpoints minimizes the number of bits to encode the whole collection of words (this is mathematically equal to minimizing the entropy of this collection). The MDL criterion causes breakpoints to segment the words into relatively common stems as well as common suffixes. This method is implemented as a framework called Linguistica[5] [Goldsmith, 2006].

[3] In every statistical stemmer some assumptions about the language are made. For example, the assumption that new word forms are derived from a basic form by adding affixes. Some languages may not conform to such an assumption, and then a different stemming approach must be used.

[4] Available at `http://members.unine.ch/jacques.savoy/clef/index.html`.

[5] Available at `http://linguistica.uchicago.edu`.

Automatic suffix discovery is investigated in [Oard et al., 2001]. At first, the frequencies of each n-gram character suffix (for $n = 1, 2, 3, 4$) are counted from each word in the collection. The frequency of each n-gram suffix is subtracted from the frequency of the adequate suffix n-gram of the lower order $n - 1$ (for example, the frequency of *ing* is subtracted from the frequency of *ng*). The altered frequencies are consequently sorted and a threshold for the optimal number of suffixes for each length is chosen. It is computed by plotting the frequency rank ratios and finding the local extreme. The suffixes with a frequency higher than the threshold are then stored so as to be stripped off during the stemming process. The suffixes are processed starting from the longest ones.

In [Bacchin et al., 2005] a new probabilistic model for word stemming is presented. The mutual relation between stems and suffixes is investigated. Two sets of substrings (prefixes and suffixes) are generated from the word lexicon by splitting the words at all possible positions. From these sets, the probabilities of prefixes and suffixes are estimated using the MLE (Maximum Likelihood Estimation) method. Three models for combinations of prefix and suffix probability estimations are defined. The stemmer selects the most probable split between stem and suffix given a chosen model. The authors experiment with several languages and measure retrieval performance in an IR system. The proposed algorithm produces results just as good as those produced by Porter's stemmer for these languages.

In [Majumder et al., 2007], YASS[6] stemmer was introduced. It is a simple approach based on word clustering. All the information needed is again taken entirely from the word lexicon. The set of string distance measures between word pairs is defined. These measures should approximate the morphological similarity between words. The lexicon is then clustered to discover morphologically related words (the equivalence classes). The authors present comparable results with rule-based stemmers (Porter's or Lovin's stemmers for English) in terms of retrieval effectiveness. Also for the French and Bengali languages, this approach improves results when compared with no stemming.

Another unsupervised approach to stemming was introduced in [Paik et al., 2011b]. The method uses simple co-occurrence statistics reflecting how often word variants (sharing a common prefix of a given length) occur in the same document. A graph-based algorithm for merging morphologically similar words is then presented. The authors evaluate their stemmer on several languages, including European languages (Czech, Bulgarian, Hungarian, English) and Asian languages (Marathi, Bengali) in the context of IR. Stemmer outperforms YASS, XU stemmer [Xu and Croft, 1998], and rule-based stemmers.

The novel graph-based stemmer GRAS (GRAph-based Stemmer) was introduced in [Paik et al., 2011a]. Similarly to the approach of YASS, GRAS is focused only on lexical information about words. The stemmer also works only with the collection of distinct words (given by the text collection). The morphological relation is represented by a graph, where the words are treated as nodes and potentially related word pairs are connected by edges. Then the pivot nodes are identified. The idea is that pivots having many neighbors are likely to be potential roots. The authors perform retrieval experiments on seven languages. According to the presented results, GRAS outperforms YASS, Linguistica, and stemmer by [Oard et al., 2001] as well as the rule-based stemmer in all seven languages in the information retrieval task. For some languages, GRAS provides a more than 50% performance improvement in the IR task when compared with no stemming.

3.3. Stemmer evaluation

Stemmers are usually evaluated indirectly via a target application, e.g., measuring the improvement in IR with and without stemming. However, there have been some attempts how to measure a stemmer's quality directly (without a target application).

One approach to direct measurement is described in [Paice, 1994]. In the article, stemming is compared with manually created groups of morphologically and semantically related word forms. They measure overstemming and understemming errors, using indices denoted by *OI* and *UI*. The indices are defined as the ratios between the number of incorrectly merged words and the total merges, and incorrectly not-merged words and the total merges. The test is designed not to take into account the frequencies of words. An

[6] YASS (Yet Another Suffix Stripper) available at `http://www.isical.ac.in/~clia/resources.html`.

error in a word with frequency 1 has the same impact on the indices as that involving a word with, e.g., frequency 100. They also consider only distinct words and stems, discarding context information.

Some of these attributes of the test may be perceived as problematic. Firstly, errors in highly frequent words have surely a higher impact on the performance of a target application than some infrequently occurring words. Secondly, decisions about the stems of words may be context dependent (i.e., the group of morphologically and semantically related word forms may differ for different contexts). Finally, the results of Paice's test are hard to interpret as it is difficult to compare the stemmers with one another (the OI and UI indices are not in the same order).

Due the above described reasons, we designed a novel approach to direct stemming evaluation, introduced in Section 5.4.

4. The proposed stemming method

Our approach consists of two main stages. The first one is based upon the idea that stemming should preserve the semantic information and remove the morphosyntactic information contained in words. The semantics should be an important clue to successful stemming. To model the semantic information, we use the findings from [Charles, 2000; Rubenstein and Goodenough, 1965], which claim that word meaning can be determined from its context. It is expected that the more similar two words are in meaning, the more similar contexts they usually share. This assumption was confirmed in these articles by empirical tests carried out on human test groups. The implication of the studies is that it is possible to compute the semantic similarity of words by a statistical comparison of their contexts.

In our work we use this finding by clustering together words occurring in similar contexts and sharing enough long common prefix (they are semantically and lexically similar). The output of this method can be directly used as the stemming result by reducing all the words in a given cluster to their longest common prefix. However, this method alone does not yield the best results. Instead, we use it to automatically generate the training data for the second stage.

The second stage of our stemmer is motivated by the assumption that stemming is subject to some rules. Given a word, these rules can decide whether and how to stem the word, depending on some conditions. These conditions can model certain properties of the given word, for example, an occurrence of particular characters in the word, the presence of a certain suffix, etc. This motivation led us to treat the second stage as a classification problem, which naturally encodes the above-mentioned rules into features. We used the maximum entropy classifier that outputs stemming decisions for given words. As a training data, we employed the stemming examples generated from all clusters at the first stage. We also relied on two expectations. First, although the clusters from the first stage (the training data) may contain incorrect stems, we assume that from the statistical point of view, these errors are not significant. Second, we expect the learned rules to be general and thus our approach should work on previously unseen data. The results in Section 5 verify both expectations. Our approach is very successful for both known (seen) and unknown (unseen) words.

The architecture of our system is depicted in Figure 1. The first stage (clustering) is used only for generating the training data for the second stage. It is thus no longer required when the trained stemmer is used for a particular task. In the architecture of our system it is possible to replace the first-phase clustering algorithm with a different way of preparing the training data. A small set of manually prepared training data, another stemming algorithm, or a lemmatizer are viable ways of preparing the training data for the maximum entropy classifier. However, we believe that clustering based on semantic assumptions is the best approach, especially when no manually prepared training data are available.

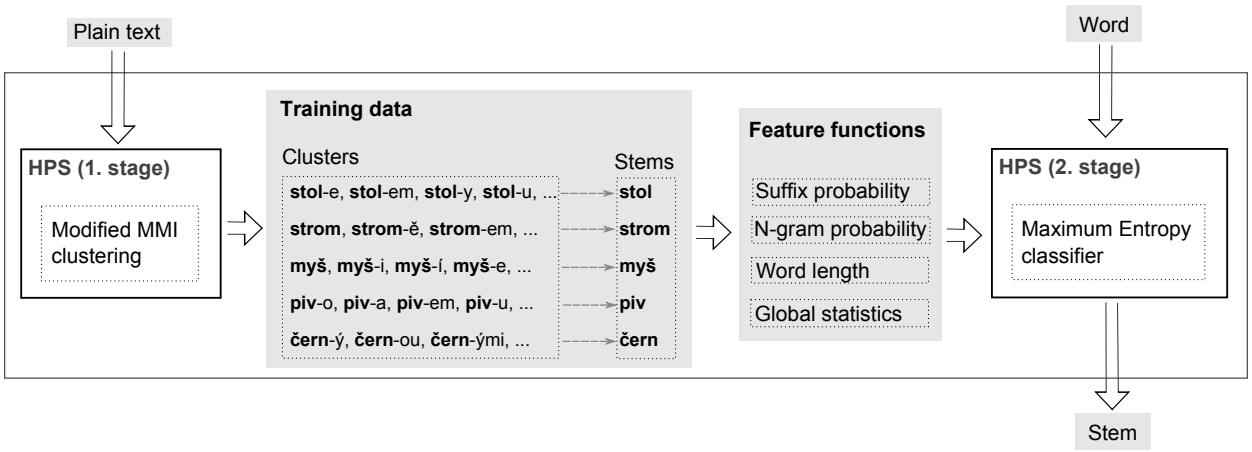


Figure 1: HPS architecture.

4.1. Stage 1: Clustering

Our approach to clustering is motivated by the MMI (Maximum Mutual Information) clustering algorithm described in [Brown et al., 1992]. The algorithm was originally developed to improve the language modeling task. In that task the clusters were created using the minimal mutual information loss scenario. After we manually observed the resulting clusters, it became apparent that they are semantically related. We believe that the semantic information comes from the principle of the algorithm to minimize the mutual information loss. As will be shown later, there is a direct connection between the similarity of neighboring words and the mutual information loss. The more similar are the neighboring words, the less mutual information is lost. A clustering method based on the similarity of neighboring words satisfies the conditions presented in [Rubenstein and Goodenough, 1965; Charles, 2000]. In these studies, the words occurring in similar contexts (having similar neighbors) are observed to be semantically similar.

In our approach we take advantage of the MMI algorithm's ability to find semantically related classes. At the same time we successfully reduce the computational costs by processing only words with a minimal (higher than a preset threshold) lexical similarity score. It is defined in the following subsection.

4.1.1. Lexical similarity

We define the lexical similarity between two words as the length of their longest common prefix normalized by the maximum of their lengths:

$$S\left(w_a, w_b\right) = \frac{|LCP(w_a, w_b)|}{\max\left(|w_a|, |w_b|\right)}, \tag{1}$$

where $LCP(w_a, w_b)$ is the longest common prefix of words w_a and w_b.

It is expected that a word stem is related to the initial part of the word. Therefore, if two words are supposed to share a stem, it is expected that they share a significantly long initial part. After normalization, $S\left(w_a, w_b\right)$ as a similarity metric for words w_a and w_b is supposed to measure a certainty that $LCP(w_a, w_b)$ is the stem of the words (from a lexical point of view).

In later stages of the clustering algorithm, the words are already members of some clusters. To compare two different clusters, we use the *complete linkage algorithm*:

$$S\left(c_a, c_b\right) = \min_{w_i \in c_a, w_j \in c_b} S\left(w_i, w_j\right), \tag{2}$$

where the resulting similarity is calculated as the minimum similarity between any member of first cluster and any member of the second.

4.1.2. Description of the MMI algorithm and its proposed modifications

Let W denote the set of possible words (word vocabulary) and C denote the set of word clusters (class vocabulary). Note that in the following we make no distinction between class and cluster. Let m be a mapping function $m : W \to C$, which maps words $w \in W$ to a class $c \in C$ ($c = m(w)$). The goal of our modified MMI clustering is to find the optimal mapping m for the stemming problem.

The original MMI clustering [Brown et al., 1992] is based on maximizing the average mutual information of adjacent classes $I(C^L; C^R)$

$$m^* = \underset{m}{\operatorname{argmax}}\ I\left(C^L; C^R\right), \tag{3}$$

where mutual information is defined as

$$I\left(C^L; C^R\right) = \sum_{c^L c^R} P\left(c^L c^R\right) \log \frac{P\left(c^L c^R\right)}{P\left(c^L\right) P\left(c^R\right)}, \quad c^L \in C^L, c^R \in C^R. \tag{4}$$

The symbol $c^L c^R$ denotes any two consecutive classes (class bigram) in the training data. The probabilities $P(c^L c^R)$, $P(c^L)$ and $P(c^R)$ are determined using MLE (Maximum Likelihood Estimation). The superscripts L and R always denote the left side and right side word classes in a bigram, respectively.

However, there is no way to find such a partitioning m^* that maximizes the average mutual information over so many possibilities ($|W|^{|W|}$). In the original paper, the problem is approximated by a greedy algorithm which is further tuned in order to decrease the complexity to the order of $|W|^3$. Such a complexity is however still very problematic. The iterative greedy algorithm merges two clusters into one cluster while maintaining a minimal mutual information loss. This means that in each step, it must find two clusters (clusters consist of already merged words or a single word) whose connection has a minimal impact on the mutual information of the whole training data. This can be formally described as follows: in each iteration i, let $m_i : W \to C_i$ denote the mapping function we are trying to optimize, where the set of word clusters C_i in the ith step is derived from merging the two particular clusters c_a and c_b from the preceding step (the other clusters remaining unchanged) into a new cluster c_{ab}:

$$C_i = ((C_{i-1} \setminus \{c_a\}) \setminus \{c_b\}) \cup \{c_{ab}\}, \quad a \neq b, \quad c_a, c_b \in C_{i-1}, \quad c_{ab} = c_a \cup c_b. \tag{5}$$

Note that the operator $\setminus$ denotes the set difference. The mapping m_i that minimizes the mutual information loss compared to the preceding step can be expressed by the following formula:

$$m_i = \underset{c_a, c_b}{\operatorname{argmin}}\ \left[I\left(C^L_{i-1}; C^R_{i-1}\right) - I\left(C^L_i; C^R_i\right)\right], \quad c_a \neq c_b, \quad c_a, c_b \in C_{i-1}, \tag{6}$$

where the clusters C_i are given by formula 5. This step is repeated until the desired final number of clusters is achieved.

In our modification of the original MMI algorithm, not all possible pairs are allowed to be merged. The merge candidates are instead limited to those which fulfill a minimal lexical similarity score.

Factoring the mutual information loss and the lexical similarity into formula 6 gives

$$m_i = \underset{c_a, c_b}{\operatorname{argmax}}\ \frac{S\left(c_a, c_b\right)}{I\left(C^L_{i-1}; C^R_{i-1}\right) - I\left(C^L_i; C^R_i\right)}, \quad c_a \neq c_b, \quad c_a, c_b \in C_{i-1}, \tag{7}$$

where $S\left(c_a, c_b\right)$ is the lexical similarity between clusters c_a and c_b (see Section 4.1.1).

Using formula 7, the distribution of clusters heads toward maximizing the lexical similarity and minimizing the mutual information loss.

The last issue of the algorithm is the selection of the termination criterion. The optimal number of clusters depends on the morphology of the analyzed language and it is not known in advance. Our solution is to repeat the process of merging clusters while there are still two clusters with lexical similarity $S(c_a, c_b) \geq \delta$. The threshold δ is chosen empirically (see Section 5). The complete clustering process is shown by the following simplified algorithm transcript 1.

Algorithm 1 Find a word mapping m into morphologically related clusters

$\delta \Leftarrow$ minimal lexical similarity between clusters
$C_0 \Leftarrow W$
$m_0 \Leftarrow W \rightarrow C_0$
$i \Leftarrow 0$
while $\exists c_a, c_b : c_a \neq c_b, S(c_a, c_b) \geq \delta$ **do** ▷ Repeat while there are still lexically similar clusters.
  $i \Leftarrow i + 1$
  $m_i \Leftarrow \underset{c_a, c_b}{\operatorname{argmax}} \frac{S(c_a, c_b)}{I(C_{i-1}^L; C_{i-1}^R) - I(C_i^L; C_i^R)}, \quad c_a \neq c_b, \quad c_a, c_b \in C_{i-1}$ ▷ Find the mapping.
  $c_{ab} \Leftarrow c_a \cup c_b$
  $C_i \Leftarrow ((C_{i-1} \setminus \{c_a\}) \setminus \{c_b\}) \cup \{c_{ab}\}$ ▷ Merge the clusters c_a and c_b into one cluster.
end while
return m_i ▷ The resulting mapping maps lexically and semantically similar words into clusters.

By introducing the lexical similarity constraint, we managed to reduce the complexity of the algorithm from $O(|W|^3)$, to $O(|W|^2 g)$ where $g << |W|$ is the average size of a group of lexically similar words. In greedy clustering, it is no longer required to compare all clusters (words) to each other, but only to compare those pairs that satisfy the minimal lexical similarity constraint. It is apparent that g is very likely to be much smaller than $|W|$, by several magnitudes.

4.2. Stage 2: Maximum entropy classifier

This section describes the second stage of our approach: the maximum entropy classifier. The stages are linked together by the clusters created in the first stage. In the second stage, they are taken as the training data for the classifier. In this way, we can use a supervised classifier while still the whole system remains unsupervised.

The principle of the classifier consists in estimating the conditional probability $p(y|x)$ of the random variable y, which is the observation on the output of a process given by the knowledge x about y. y is a member of a finite set of all possible outputs Y and x is a member of a finite set of all possible pieces of knowledge X.

The training data are used to set constraints for the conditional distribution. Each constraint expresses a characteristic (knowledge) about the training data that is requested to be present in the final probability distribution. The facts (the knowledge) about the training data are captured by n real-valued feature functions $f_i(x, y) \in \langle 0, 1 \rangle$.

The final model distribution is restricted in such a way that it has the same expected values for all features as seen in the training data. This can be formalized as

$$E(f_i(x, y)) = \tilde{E}(f_i(x, y)), \quad 1 \leq i \leq n, \tag{8}$$

where $\tilde{E}(f_i(x, y))$ is the expected value of a feature $f_i(x, y)$ estimated from the training data and $E(f_i(x, y))$ is the expected value of this feature given by the final model.

It was shown in [Berger et al., 1996] that the requested conditional probabilities $p(y|x)$ given the model from formula 8 have exponential form and can be estimated as follows:

$$p(y|x) = \frac{1}{Z(x)} \prod_{i=1}^{n} e^{\lambda_i f_i(x,y)}, \tag{9}$$

where $Z(x) = \sum_{y \in Y} \prod_{i=1}^{n} e^{\lambda_i f_i(x,y)}$ is a normalization function. The parameters λ_i of the maximum entropy model can be estimated by some algorithm for finding the global maximum of a function, such as IIS[7] or by

[7] IIS (Improved Iterative Scaling) is a hill-climbing algorithm for finding optimal parameters in log-likelihood space. The algorithm is described for example in [Berger et al., 1996].

some more sophisticated method, for example by OWL-GN[8].

In order to apply the maximum entropy classifier to the word stemming task (suffix stripping), we need to solve a few issues. Firstly, we define y as the length of a suffix of a given word (it is the suffix that is being stripped off) and $Y = \{0, 1, \ldots, M\}$, where M is the maximum of all possible lengths of all suffixes. The x is the word itself. Secondly, we need to define a set of features which add constraints to the final model. We use four types of features, which are described in detail in the following sections. Finally, we use the clusters given in the previous Section 4.1 as the training data for the maximum entropy classifier.

4.2.1. Variables for features

Before we describe the various features for the maximum entropy approach, we need to define a few variables. Firstly, let

$$w = l_1 l_2 \cdots l_L = l_1^L, \quad L = |w|, \tag{10}$$

denote a character string of the word w, where L is the length of the word. Then l_a^b denotes the substring of the word from position a to position b.

Let the stem be the longest common prefix of a word group c (note that the groups are provided by stage 1 of our algorithm and may contain errors):

$$stem\,(w) = l_1^{\min|LCP(w,w_i)|}, \quad w, w_i \in c, \tag{11}$$

where w is a given word for which we need to find a stem and w_i are other words belonging to the same cluster c.

Then the suffix of a given word is the remaining part following the stem:

$$suff(w) = l_{\min|LCP(w,w_i)|+1}^{L}, \quad L = |w|, \quad w, w_i \in c. \tag{12}$$

Now, we define an arbitrary ending of a word. It is simply a K character long ending of a word w:

$$end\,(w, K) = l_{L-K+1}^{L}, \quad L = |w|. \tag{13}$$

4.2.2. Suffix length statistics

This feature represents the global distribution of suffixes according to the word length. The probability that an L character long word contains a suffix of length m is estimated using MLE:

$$P_{stats}\,(L, m) = \frac{\#\,\{w \in W : |w| = L, |suff(w)\,| = m\}}{\#\,\{w \in W : |w| = L\}}, \quad 0 \leq m \leq M. \tag{14}$$

This is simply the number of times that the L character long word contains an m character long suffix, normalized by the total number of words with length L. The function # denotes the number of elements in a set. M is the maximum length of suffix to be stripped off.

The feature function is

$$f_{stats}\,(w, m) = P_{stats}\,(|w|\,, m)\,, \quad 0 \leq m \leq M, \tag{15}$$

where m is the position in the word w where the word should be split between stem and suffix.

The motivation for this feature is the assumption that the length of the suffixes depends on the length of the stems. It adds $M + 1$ features to the maximum entropy classifier (one for every possible length of a suffix).

[8]OWL-GN (Orthant-Wise Limited-memory Quasi-Newton) described in [Andrew and Gao, 2007] is an algorithm for the efficient optimization of larger numbers of parameters in log-linear models. It is based upon the L-BFGS (Limited-memory variation of the Broyden–Fletcher–Goldfarb–Shanno) algorithm. However, the authors show that it is much faster than other algorithms.

4.2.3. The probability of being a suffix

The most important feature (our experiments show that removing this feature from the feature set causes the highest performance drop) for the classifier is the probability of being a suffix. It is defined as the probability that the word ending is the correct suffix. This probability is based on assessing the training data created in stage 1. For example, if the word ending *ing* is observed, it need not be the correct suffix (*king, ring, sparing*), but it can be (*drinking, swimming, sleeping*). This probability represents the amount of certainty that the observed word ending is a suffix causing the inflective form of the word (it is not a part of the stem) and therefore it needs to be stripped off. We can estimate the probability as follows:

$$P_{suff}\left(suff(w)\right) = \frac{\#\left\{w_i \in W : suff(w_i) = suff(w)\right\}}{\#\left\{w_i \in W : end\left(w_i, |suff(w)|\right) = suff(w)\right\}}, \tag{16}$$

which is essentially the number of times where *suff*(w) follows the stem of word w_i (for each word in each cluster), divided by the number of all times where the word w_i ends with *suff*(w).

The corresponding feature for the classifier has the following form:

$$f_{suff}(w, m) = P_{suff}\left(end\left(w, m\right)\right), \quad 0 \leq m \leq M. \tag{17}$$

The function adds $M + 1$ features to the final classifier.

4.2.4. The probability of an n-gram's standing before a suffix

As shown earlier, the word ending that resembles a correct suffix (e.g., *ing*) does not always means that stripping it off is a correct action: e.g., *drinking* vs. *king*. In order to disambiguate such cases we introduce a feature that captures the context of the characters that precede the suffix.

Let

$$ngram\left(w, N, K\right) = l_{L-N-K+1}^{L-K} \tag{18}$$

denote an N-character substring (N-gram) of the word w, which ends K characters before the end of the word. This means that this substring starts at the position $L - N - K + 1$ and ends at the position $L - K$, where $L = |W|$. Then we define the probability

$$P_{ngram}\left(ngram\left(w, N, K\right)\right) = \frac{\#\left\{w_i \in W : end\left(stem\left(w_i\right), N\right) = ngram\left(w, N, K\right)\right\}}{\sum_{m=0}^{M} \#\left\{w_i \in W : ngram\left(w_i, N, m\right) = ngram\left(w, N, K\right)\right\}}, \tag{19}$$

as the probability of stripping off a suffix after the observation of the N-gram $ngram(w, N, K)$ in the word w. This probability is calculated using MLE as the number of times where the N-gram is observed at the end of the stem, divided by the total number of times where the N-gram is observed in a word.

The feature function is then defined as

$$f_{ngram}\left(w, m\right) = P_{ngram}\left(ngram\left(w, N, m\right)\right), \quad 0 \leq m \leq M. \tag{20}$$

We have experimentally discovered that the best results are achieved by using N-grams of lengths 1, 2 and 3 (N is set to 1, 2, and 3). This means that this feature produces $3(M + 1)$ features for the maximum entropy classifier.

4.2.5. Word length

We also assume that decisions about stemming depend on the length of the words. Therefore, we introduce the last type of feature function:

$$f_{length}\left(w, m\right) = \begin{cases} 1 & if\left(|w| = L\right) \\ 0 & otherwise \end{cases}, \quad 0 \leq m \leq M, \tag{21}$$

where L ranges from 1 to L_{max}. Thus, $L_{max}(M + 1)$ features are added to the maximum entropy classifier.

The total number of all features for all possible splitting positions is then $M + 1 + M + 1 + 3(M + 1) + L_{max}(M + 1) = (5 + L_{max})(M + 1)$, where M is the maximum length of suffix to be stripped off.

5. Experimental results

In this section we provide the results of experiments from three different perspectives. Firstly, we look at the stemmer from the inflection removal point of view (Section 5.4). This experiment indicates how well the stemmer removes the inflection of the word forms. The second perspective is the retrieval performance, which is the most frequently used way of measuring the performance of a stemmer (Section 5.5). Finally, stemmers are used to improve language modeling (Section 5.6). Although the first and third experiments are not considered as traditional testing environments, they may uncover the degree of versatility of each tested stemmer. A versatile stemmer should cope well in variety of tasks. Also, successes and failures in different tasks reveal the properties of particular stemmer methods. For example, if a stemmer performs well in IR but fails in inflection removal tasks, it indicates that it produces stems that do not resemble lemmas. Such information may be useful when a similar task (that is know to work well with lemmas) is to be solved.

We also measure the performance of other competitive stemmers (namely, GRAS, YASS, Linguistica as well as rule-based stemmers) on the same data and with the same constraints and conditions. Experiments are conducted for several languages, namely Czech, Slovak, Polish, Hungarian, Spanish, and English. The settings of each stemmer are described in the following Section 5.1.

5.1. Stemmer settings

This section gives an overview of the settings of all tested stemmers that were used in our experiments. Information about stop words was not taken into account in our experiments, in order to make our approach completely unsupervised. The settings of stemmers are as follows:

- **HPS**: The max length of suffix M was set to 3 for all languages, which means we classify into 4 classes (4 possible lengths of suffix, i.e., from 0 to 3 characters). The minimum similarity between two clusters (the stopping condition for clustering) was set to $\delta = 0.7$ for Czech, Slovak, English, and Spanish. For Polish and Hungarian, $\delta = 0.6$. Stemming is performed in two iterations for all languages (see Section 5.2).

  We implemented HPS on the Java™ platform and we used Brainy [Konkol, 2014] implementation of maximum entropy classifier.

- **GRAS**: The suffix frequency cut-off coefficient (which is used to prune invalid suffix pairs) was set to 4. The *cohesion* threshold used to measure whether two nodes are morphologically related or not was set to 0.8. Both parameters are recommended by the authors of GRAS.

- **YASS**: The clustering threshold value was set to 1.5 for all languages. This setting is recommended by the authors of this stemmer.

- **Linguistica**: This does not require any special settings. However, the number of tokens used for training is limited in the only available implementation of this stemmer. The maximal amount of tokens is set to 5,000,000.

- **Rule-based stemmers**: We used Porter's stemmers for languages available via the Snowball framework. For Czech, we chose to use the light stemmer presented in [Dolamic and Savoy, 2009]. Despite the fact that, according to the authors, the aggressive stemmer performs slightly better in IR (our experiments agree with this), the results of the light stemmer on inflection removal are much better than the results of the aggressive one. For Polish and Slovak, we have not found any rule-based stemmers.

Note that the δ parameter for HPS is set empirically. By changing δ it is possible to tune the aggressivity of the stemmer. The lower δ is, the more aggressive a stemmer is created, because more words are grouped by a clustering. The parameter δ can be perceived as the minimal ratio between the length of the stem and the length of the word. We recommend setting lower values of δ for languages that tend to have long suffixes (e.g., Polish and Hungarian) and higher values of δ for other languages. We recommend 0.6 as the lower value and 0.7 as the higher value. However, it is possible to tune the stemmer using δ for a specific

task and a language. We did not do so, in order to have results that were not tuned for the test data or the task (our aim is to design a multi-purpose stemmer). In fact, δ is the only information about the language that needs to be determined for training HPS.

We also created three new stemmers by extending GRAS, YASS, and Linguistica by our second stage of HPS (maximum entropy classifier). These stemmers are denoted by **GRAS+HPS**, **YASS+HPS**, and **Linguistica+HPS**, respectively. The original stemmers are used only for generating the training data for the classifier (in the same way as our modified MMI clustering). All other settings remain unchanged.

5.2. Iterative stemming

In our initial experiments, it turned out that some words with more complicated morphology remain understemmed. Repeated or iterative stemming proved to be beneficial for such words. The stage 2 algorithm is repeated more than once. During such a process, the understemmed words are shortened. However, there is a risk of overstemming (creating too short a stem). This process is efficient especially for languages with more complex morphology and it leads to increasing the recall rate in particular.

Consider the following example, which deals with the complex inflectional morphology of Czech. The words *mlad-ší* (younger – 1st case) and *mlad-ší-mi* (younger – 7th case) share the same stem *mlad* with suffixes *ší* and *mi*. In the second word, the suffix *mi* follows the suffix *ší*. In such a case, the second suffix *mi* may be removed during the first iteration and the first suffix *ší* in the second one. The result of the second iteration is the correct stem *mlad*. Another example concerns derivation in English: *fear-less-ly*. The first iteration produces *fear-less* (the correct stem) and the second one *fear* (overstemmig).

The above mentioned examples show the advantages and drawbacks of iterative stemming. When applying this method, we should be particularly careful with derivational suffixes since the method may easily produce overstemming errors (see the example above). On the other hand, if we deal with inflectional suffixes, iterative stemming can only be beneficial. No inflectional suffix can change the lemma and thus it can not change the meaning.

In our tests we experimented with the setting of the number of iterations. We found that the optimal value is two iterations for all languages and all tests. Higher values have little impact on the quality of the results. We discovered that when we use iterations, the stemmer shortens the understemmed words but the well-stemmed words are left intact.

5.3. Training corpora

We experimented with six languages, namely Czech (CZ), Slovak (SK), Polish (PL), Hungarian (HU), Spanish (ES), and English (EN). The stemmers were trained on the unlabeled corpora mentioned below. Since the implementation of the Linguistica stemmer limits the training size to 5,000,000 tokens, we used this limit for all stemmers. The exceptions are the tests with variable training data sizes. We used up to 15,000,000 tokens there.

Statistics for the corpora are presented in Table 1. We distinguish between (word) *tokens* and *words*. *Token* refers to a single occurrence of a word in the text. By *word*, we mean one particular word that can occur many times in a text. In the table, we count the total number of words and the number of words that occur in texts at least five times.

The training corpora are:

- **CZ**: This corpus contains news in Czech on various topics, such as political, business, sports, international, and other news gathered from one year. The data in this corpus are provided by the Czech News Agency.
- **SK**: A huge number of texts from the Slovak National Corpus[9] oriented towards the arts, journalism, and the professions.

[9]The prim-5.0-public-all subcorpus of the Slovak National Corpus available at http://korpus.juls.savba.sk.

- **PL** and **HU**: The corpora for Polish and Hungarian which we use in this work are part of a multilingual parallel corpus available through the Joint Research Center (JRC)[10].
- **ES**: The Spanish texts are taken from the Reuters Multilingual Corpus (RCV2). The data are gathered from 1996 and 1997. The corpus is available through the NIST (National Institute of Standards and Technology) Standard Reference Data Products[11].
- **EN**[12]: The data represent a sampling of approximately 40% of the articles published by the Los Angeles Times in the two-year period from Jan 1, 1989 to December 31, 1990.

Table 1: Parameters of the corpora used for training the stemmers. The number of distinct words in a corpus is denoted by *words min. 1*. The number of distinct words occurring at least 5 times in a corpus is denoted by *words min. 5*. The full size of a corpus in terms of the number of tokens is denoted by *tokens*.

	CZ	SK	PL	HU	ES	EN
tokens	35,538,656	85,817,979	35,703,800	33,640,074	23,561,417	74,993,849
words min. 1	577,200	1,031,215	395,140	584,400	254,463	435,123
words min. 5	189,976	333,511	117,653	149,657	78,977	139,328

5.4. Inflection removal experiments

We start our evaluation with a direct test. Our motivation is the same as in [Paice, 1994]. We desire a deeper insight into a stemmer's quality without the effect of a target application. However, we decided to design our test differently. The reason is the presence of the two problematic attributes of the test mentioned in the state-of-the-art section, and mainly due to the lack of manually annotated data for languages other than English. Instead of groups of words that should have same stem, we use readily available groups of words sharing the same lemma. And instead of overstemming and understemming indices, we use precision, recall, and F-measure. The precision directly relates to overstemming errors (the higher the precision, the lower the frequency of errors) and recall to understemming errors. Our values respect the frequencies of words and are computed directly from the texts (not from dictionaries), so they reflect the contexts.

Our test is based upon measuring the ability of a stemmer to remove an inflection from an inflected word form, by comparing groups of words with the same stem with groups of words with the same lemma. Ideally, these should be equivalent. Naturally, such a test is focused solely on inflective morphology and omits the derivation linguistic process. In fact, the test measures the ability of the stemmer to approximate lemmas. The score from the test thus should indicate the ability of the stemmer to replace a lemmatizer in applications where lemmatizers are working well, e.g., language models [Brychcín and Konopík, 2011], machine translation [Koehn and Hoang, 2007], etc. Naturally, the score will be less related to applications which require aggressive stemming and working with derivational linguistic processes, e.g., information retrieval. Therefore, the test does not cover all aspect of stemming. However, we believe that a universal test of stemmers can not be constructed, simply because stemming is always to some extent task dependent.

In our test, we go through the test corpus and for each position in the text we analyze two word groups. The first one consists of all words sharing the same stem with the word at its actual position (the result of the tested stemmer). The second group contains all words sharing the same lemma with the actual word (given by the lemmatized data). We calculate precision (P), recall (R) and their harmonic mean, the F-measure (F_m):

$$P = \frac{tp}{tp + fp}, \quad R = \frac{tp}{tp + fn}, \quad F_m = \frac{2PR}{P + R}. \tag{22}$$

[10] Available at http://langtech.jrc.it/.
[11] Available at http://trec.nist.gov/data/reuters/reuters.html.
[12] Available from NIST Standard Reference Data Products at http://www.nist.gov/srd/nistsd23.cfm.

Here, tp denotes the number of times the word in the stemmer group matched a word in the lemma group and fp denotes the number of times the stemmer group contained the wrong word. Finally, fn denotes the number of times the stemmer group missed the correct word. The higher the precision rate, the fewer times the stemmer produces an overstemming error. On the other hand, the higher the recall rate, the fewer the understemming results. The F-measure takes both these errors into account and thus can be used as a final evaluation of the stemming quality.

5.4.1. Test corpora

The test corpora contain manual morphological annotations with lemmas. The list of the corpora follows:

- **CZ**: The data are part of the Prague Dependency Treebank 2.0[13]. The texts in the corpus consist of manually annotated articles from several newspapers and journals in Czech.
- **SK**: The manually annotated part of the Slovak National Corpus[14] mainly consists of artistic, journalistic, and professional texts.
- **PL**: The manually annotated part of the National Corpus of Polish[15].
- **HU**: The manually annotated Hungarian texts of the Szeged Corpus 2.0[16].
- **ES**: The Spanish texts are taken from the Ancora 2.0[17] [Taulé et al., 2008] corpus, which is mainly oriented to journalism.
- **EN**: These texts are part of the Open American National Corpus (OANC)[18].

Some statistics of the corpora are shown in Table 2.

Table 2: Parameters of the annotated corpora used for testing the stemmers. The number of distinct words in a corpus is denoted by *words min. 1*. The number of distinct words occurring at least 5 times in a corpus is denoted by *words min. 5*. The full size of a corpus in terms of the number of tokens is denoted by *tokens*.

	CZ	SK	PL	HU	ES	EN
tokens	1,748,519	1,033,345	1,215,415	1,214,490	455,702	3,490,816
words min. 1	168,442	137,236	143,820	154,240	44,353	107,087
words min. 5	35,350	24,512	23,545	25,689	8,809	29,902

5.4.2. Results

This section presents the inflection removal results. Each stemmer was trained on the first 5,000,000 tokens from the appropriate corpus (Section 5.3) and then evaluated on the corresponding annotated corpus (Section 5.4.1). The results for our novel test are shown in Table 3.

[13] More information at http://ufal.mff.cuni.cz/pdt2.0/.
[14] The r-mak-3.0 subcorpus of Slovak National Corpus available at http://korpus.juls.savba.sk.
[15] Available at http://nkjp.pl/.
[16] Available at http://www.inf.u-szeged.hu/projectdirs/hlt/index_en.html.
[17] Available at http://clic.ub.edu/corpus/en/ancora.
[18] Available at http://www.anc.org/data/oanc.

Table 3: Inflection removal results. P[%], R[%] and F_m[%] denote the stemming precision, recall and F-measure, respectively. The results are expressed in percentages.

(a) Czech (CZ)

CZ	P[%]	R[%]	F_m[%]
Rule	69.3	40.4	51.1
HPS (1. phase)	**79.0**	35.9	49.3
HPS (both phases)	72.5	42.2	**53.4**
GRAS	47.7	46.4	47.0
GRAS+**HPS**	47.1	**47.9**	47.5
YASS	52.5	43.0	47.3
YASS+**HPS**	51.1	45.1	47.9
Linguistica	52.2	36.7	43.1
Linguistica+**HPS**	54.0	43.0	47.9
No stemming	100	14.0	24.5

(b) Slovak (SK)

SK	P[%]	R[%]	F_m[%]
Rule	/	/	/
HPS (1. phase)	**83.9**	37.4	51.7
HPS (both phases)	75.3	46.5	**57.5**
GRAS	48.5	51.3	49.9
GRAS+**HPS**	52.8	**52.4**	52.6
YASS	60.5	44.4	51.2
YASS+**HPS**	56.1	48.3	51.9
Linguistica	61.1	38.7	47.4
Linguistica+**HPS**	61.6	47.5	53.6
No stemming	100	15.4	26.6

(c) Polish (PL)

PL	P[%]	R[%]	F_m[%]
Rule	/	/	/
HPS (1. phase)	**75.2**	28.9	41.7
HPS (both phases)	67.9	34.8	**46.1**
GRAS	62.5	30.3	40.8
GRAS+**HPS**	56.4	35.1	43.2
YASS	60.3	24.9	35.3
YASS+**HPS**	55.8	**36.5**	44.1
Linguistica	70.4	25.0	36.9
Linguistica+**HPS**	64.8	34.6	45.1
No stemming	100	10.8	19.6

(d) Hungarian (HU)

HU	P[%]	R[%]	F_m[%]
Rule	73.1	**61.7**	**66.9**
HPS (1. phase)	**75.5**	35.3	48.1
HPS (both phases)	56.1	49.3	52.5
GRAS	63.5	44.3	52.2
GRAS+**HPS**	58.0	46.2	51.4
YASS	67.3	22.4	33.7
YASS+**HPS**	53.5	50.7	52.1
Linguistica	51.4	24.4	33.1
Linguistica+**HPS**	48.8	46.3	47.5
No stemming	100	7.9	14.6

(e) Spanish (ES)

ES	P[%]	R[%]	F_m[%]
Rule	58.8	**50.6**	**54.4**
HPS (1. phase)	**86.6**	32.2	46.9
HPS (both phases)	72.8	37.3	49.7
GRAS	52.0	49.7	50.8
GRAS+**HPS**	66.0	41.7	51.1
YASS	60.1	39.0	47.3
YASS+**HPS**	60.4	40.3	48.3
Linguistica	64.7	32.0	42.8
Linguistica+**HPS**	70.5	37.9	49.3
No stemming	100	22.1	36.1

(f) English (EN)

EN	P[%]	R[%]	F_m[%]
Rule	69.4	**77.2**	**73.1**
HPS (1. phase)	76.2	58.5	66.2
HPS (both phases)	**80.4**	63.2	70.8
GRAS	55.8	70.8	62.4
GRAS+**HPS**	60.7	68.7	64.4
YASS	41.4	71.0	52.3
YASS+**HPS**	46.1	68.1	55.0
Linguistica	54.1	65.6	59.3
Linguistica+**HPS**	55.1	68.1	60.9
No stemming	100	50.9	67.4

Before analyzing the results for the tested stemmers, we should note the following. As we explained in the Introduction, the goal of a stemming algorithm depends on the particular task at hand. Some aggressive stemmers usually remove derivational suffixes as well as inflectional suffixes. In this testing scenario, removing derivational suffixes is penalized since such an action incorrectly creates the same stem for different lemmas. Thus, we can interpret the results of this test only in relation to the inflection removal task. Nevertheless, we also pointed out that removing derivational suffixes is risky whereas removing inflectional

suffixes is safe. Thus, we can expect that a stemmer successful in this test should produce a low number of overstemming results.

If we use the F-measure (F_m) as a quality measure for the inflection removal experiments, we can conclude the following. On all languages except Hungarian and Spanish, HPS yields the best results of all unsupervised stemmers. For Hungarian and Spanish, the results of HPS are similar to those of GRAS.

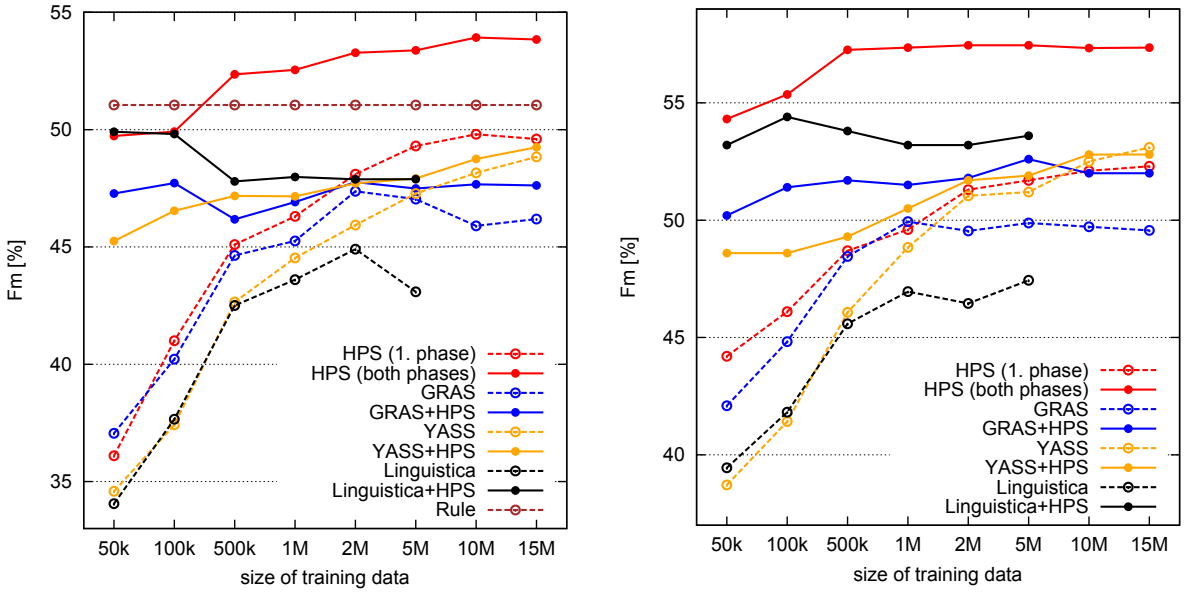
On Czech and Slovak, the results of GRAS and YASS are similar. On other languages, GRAS performs much better than YASS. Linguistica performs the worst in this testing scenario. Furthermore, we can see that HPS always achieves one of the best precision rates. In contrast, GRAS always has one of the best recall rates.

In the tables we also show the results for the stand-alone first phase of HPS to see how large an improvement the second phase of HPS (maximum entropy classifier) produces. The first phase of HPS does not lead to the best results, but has the best precision (except on English).

Our maximum entropy extension also improves the other stemmers (GRAS+HPS, YASS+HPS and Linguistica+HPS). However, the combination of maximum entropy and the first phase of HPS is generally the best one. We assume this to be due to the high precision of the first phase of HPS (with little overstemming errors) which leads to creating better training data for the maximum entropy classifier.

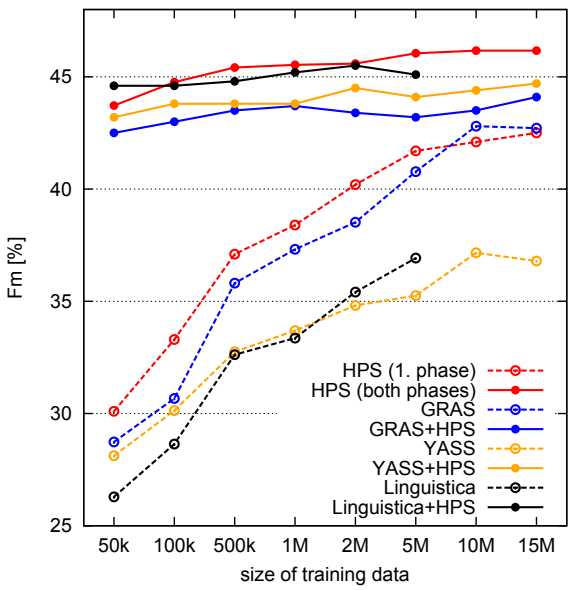
5.4.3. The impact of the size of the training dataset

In this section, we study how the quality of the stemming changes depending on the amount of training data available. The stemmers are evaluated on the same data as in the previous section, but different numbers of tokens being used for training. We start with a very small amount of training data (50,000 tokens) for each language, and continue up to 15,000,000 tokens, which we believe is a sufficient amount for all methods. The results are shown in Figure 2.

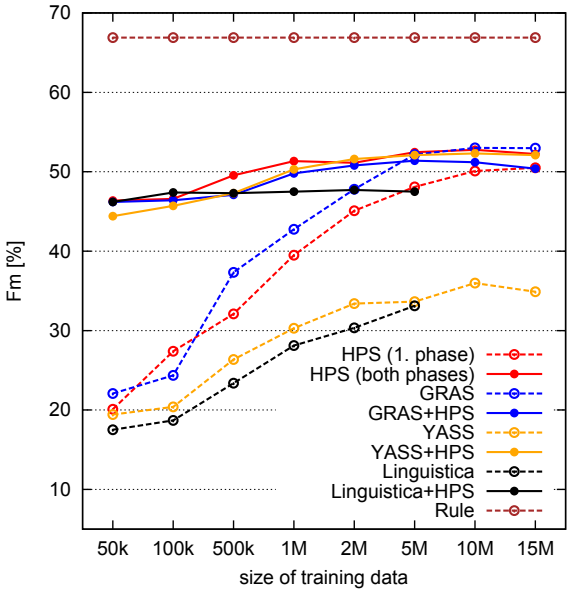


(a) Czech (CZ)

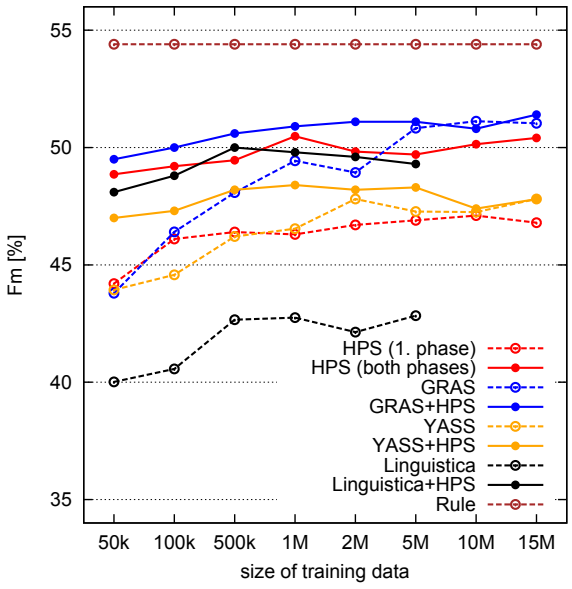
(b) Slovak (SK)



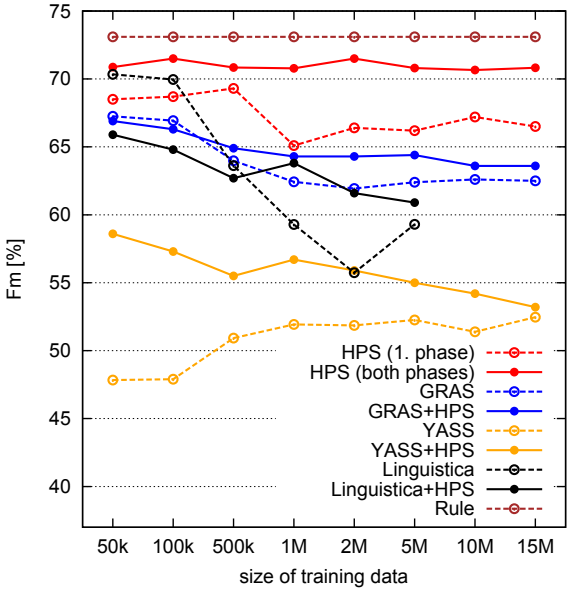
(c) Polish (PL)



(d) Hungarian (HU)



(e) Spanish (ES)



(f) English (EN)

Figure 2: Inflection removal experiments with different amounts of training data available.

From the figures presented above, we can conclude that our stemmer excels in experiments for all sizes of training data and all tested languages. The F-measure results are better for Slavic languages (Czech, Slovak, Polish) in particular.

A very interesting and also important fact is that our stemmer gives very promising results even for an amount of training data as small as 50,000 tokens for each language. In addition, it is possible to say that after 1,000,000 tokens for training, the results of our stemmer do not improve significantly. Thus, we

can state that 1,000,000 tokens are optimal for satisfactory results. This property is caused by the second stage of HPS, as we observe large improvements even for the other stemmers when they use this extension (GRAS+HPS, YASS+HPS and Linguistica+HPS).

As we expected, the quality of other stemmers (without the second stage of HPS) rises significantly with an increasing amount of training data. After a certain point (5M or 10M tokens) the results are not improved any more. The exception is the English (our less inflected language), where the performance of the stemmers decreases or stagnates. We believe this is due to focusing on the purely lexical level of the words, where with a rising amount of training data, the larger number of word forms can yield an increase in the recall rate but a very significant drop in precision (some stemmers start to overstem the words). The outcome is that the F-measure drops. Note that this problem is specific to the inflection removal experiments. However, in the information retrieval experiments (Section 5.5), this does not mean a definite retrieval drop.

5.5. IR experiments

In this section, we experiment with using different stemmers for the information retrieval task. We compare the results of our stemmer with those of other competitive stemmers in four languages. We use the open source search engine Terrier[19], which implements state-of-the-art indexing and retrieval functionalities. Terrier is written in the Java™ platform and was developed by the School of Computing Science at the University of Glasgow.

In our experiments, we used the I(F)B2 model for term weighting. I(F)B2 denotes the model I(F) (*tf-itf* model: term frequency – inverse collection term frequency), with the normalization factor B (Bernoulli after-effect given by the ratio of two Bernoulli processes) and with the assumption of the hypothesis H2 (the term frequency density is inversely related to the length of document). The derivation and definition of this function can be found in [Amati and Van Rijsbergen, 2002]. The authors of the article also show a comparison of several models for IR. The I(F)B2 is suggested to be one of the best performing models.

5.5.1. Corpora

The evaluation presented in this section is based upon the collection built during the CLEF[20] evaluation campaign (CLEF Evaluation Package AdHoc News 2004-2008).

The collection comprises queries which follow the guidelines of the TREC ad-hoc task. Each query is structured into three sections: title (reflects the queries that users send to search engines), description (one sentence description of the requested data), and narrative part (a few sentences describing the criteria for the requested data). A set of relevant documents (correct answers) is provided for every query in the collection. In our experiments, we used the title and description parts only.

Table 4: Parameters of corpora used for IR experiments.

	CZ	HU	ES	EN
documents	81,735	49,530	454,045	113,005
words	461,999	542,075	556,482	226,529
tokens	19,544,124	8,379,455	82,392,138	37,743,118
evaluation queries	50	150	60	100
relevant documents	762	3,158	2,368	2,096

5.5.2. Results

In this section we use the stemmers listed in Section 5.1 to improve retrieval performance for four languages. As described in [Majumder et al., 2007; Paik et al., 2011a], the unsupervised stemmers YASS and GRAS should give as good results in the retrieval context as rule-based stemmers. Our experiments confirm this. However, we also present the retrieval experiments from a slightly different point of view.

[19] Available at http://terrier.org.
[20] Available at http://www.clef-campaign.org/.

In the real world, the indexing performed by an IR system is an iterative process. New data arise continually, and they need to be indexed. However, this fact is usually not taken into account when testing stemmers (YASS and GRAS). During the tests, it is expected that a stemmer is trained on all the data that are indexed. However, in reality, the new data are unseen by the stemmers. Retraining the stemmer is computationally expensive, although possible. However, this process has one big problem. The retrained stemmers are likely to stem some already seen words differently because in many stemmers the stemming decisions are learned from all the data. The direct implication is that the complete data set needs to be reindexed. Reindexing is a process that takes a lot of computer time, especially for large data sets. It also introduces scalability problems when distributing the index[21]. We, however, believe that the retraining and reindexing steps can be done much less frequently or even completely avoided. In such a scenario, the stemmer that is being used needs to be able to handle unseen data (data not seen during training). Taking these facts into account, we have decided to perform the retrieval experiments using stemmers trained on both types of data:

- **Seen**: The stemmers are trained on the same data as they are used for indexing, which causes all words that are indexed to be known by the stemmers. This experiment allows comparisons to be made with the results in the original papers about competitive stemmers. Unfortunately, this test does not include results for Linguistica, since its implementation limits the training data size to 5,000,000 tokens only.

- **Unseen**: The stemmers are trained on the data used for the inflection removal experiments described in Section 5.4.1, which means that the indexed data are previously unseen by the stemmers. This way of testing corresponds to the scenario we introduced above. We used a completely different corpus for training the stemmers because we want to emphasize the ability of a stemmer to work with unseen data. However, we must note that in the scenario where the newly indexed data (the unseen data) are from the same domain, the difference between seen and unseen data probably will not be so big.

To calculate all performance scores, we used the *TREC-EVAL* program, which is the standard tool for the evaluation of TREC results using the standard NIST evaluation procedures.

Retrieval performance was measured using several measures. In the tables below, *MAP* denotes the Mean Average Precision. *R-prec* is an R-precision measure that represents the precision at the Rth position in retrieved documents for a query that has R relevant documents. The symbols P@5 and P@10 denote the precisions at fixed low levels of retrieved results (5 and 10 documents). The number of retrieved relevant documents is denoted by *Rel-ret*. The results are shown in Table 5.

[21] An input query should be preprocessed with the same stemmer as the index. When the index is distributed, the stemmers need to be synchronized for all parts of the index.

Table 5: Information retrieval results. The numbers in brackets are the relative improvements compared with no stemming.

(a) Czech (CZ)

CZ	MAP	R-Prec	P@5	P@10	Rel-ret
No stemming	.227	.248	.324	.260	556
Rule	.326 (43.6%)	.328 (32.3%)	.428 (32.1%)	.334 (28.5%)	663
CZ-unseen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.296 (30.4%)	.295 (19.0%)	.396 (22.2%)	.314 (20.8%)	660
HPS (both phases)	.324 (42.7%)	.327 (31.9%)	.404 (24.7%)	.338 (30.0%)	672
GRAS	.305 (34.4%)	.284 (14.5%)	.372 (14.8%)	.312 (20.0%)	671
GRAS+**HPS**	.317 (39.6%)	.312 (25.7%)	.392 (21.0%)	.328 (26.2%)	660
YASS	.299 (31.7%)	.282 (13.7%)	.380 (17.3%)	.306 (17.7%)	660
YASS+**HPS**	.316 (39.1%)	.302 (21.8%)	.392 (21.0%)	.332 (27.7%)	660
Linguistica	.283 (24.7%)	.291 (17.3%)	.396 (22.2%)	.294 (13.1%)	654
Linguistica+**HPS**	.325 (43.2%)	.305 (22.9%)	.428 (32.1%)	.340 (30.8%)	659
CZ-seen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.325 (43.2%)	.314 (26.6%)	.425 (31.2%)	.336 (29.2%)	672
HPS (both phases)	.327 (44.1%)	.316 (27.4%)	.420 (29.6%)	.334 (28.5%)	667
GRAS	.332 (46.3%)	.317 (27.8%)	.380 (17.3%)	.316 (21.5%)	679
GRAS+**HPS**	.322 (41.6%)	.314 (26.6%)	.384 (18.5%)	.316 (21.5%)	667
YASS	.317 (39.6%)	.316 (27.4%)	.404 (24.7%)	.330 (26.9%)	667
YASS+**HPS**	.319 (40.4%)	.312 (25.7%)	.404 (24.7%)	.330 (26.9%)	669

(b) Hungarian (HU)

HU	MAP	R-Prec	P@5	P@10	Rel-ret
No stemming	.216	.231	.317	.273	1,963
Rule	.315 (45.8%)	.333 (44.2%)	.427 (34.7%)	.365 (33.7%)	2,518
HU-unseen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.255 (18.1%)	.282 (22.1%)	.365 (15.1%)	.315 (15.4%)	2,305
HPS (both phases)	.309 (43.1%)	.318 (37.7%)	.427 (34.7%)	.355 (30.0%)	2,589
GRAS	.253 (17.1%)	.277 (19.9%)	.356 (12.3%)	.307 (12.5%)	2,281
GRAS+**HPS**	.315 (46.0%)	.328 (42.2%)	.441 (39.2%)	.364 (33.3%)	2,573
YASS	.247 (14.4%)	.265 (14.7%)	.347 (9.5%)	.303 (11.0%)	2,264
YASS+**HPS**	.311 (44.1%)	.324 (40.2%)	.437 (37.8%)	.355 (29.9%)	2,589
Linguistica	.241 (11.6%)	.252 (9.1%)	.340 (7.3%)	.294 (7.7%)	2,191
Linguistica+**HPS**	.305 (41.3%)	.314 (35.8%)	.419 (32.1%)	.351 (28.7%)	2,587
HU-seen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.315 (45.8%)	.330 (42.9%)	.425 (34.1%)	.364 (33.3%)	2,598
HPS (both phases)	.319 (47.7%)	.333 (44.2%)	.428 (35.0%)	.367 (34.4%)	2,637
GRAS	.324 (50.0%)	.340 (47.2%)	.425 (34.1%)	.369 (35.2%)	2,689
GRAS+**HPS**	.315 (45.9%)	.328 (41.8%)	.423 (33.3%)	.360 (31.9%)	2,575
YASS	.315 (45.8%)	.327 (41.6%)	.429 (35.3%)	.372 (36.3%)	2,587
YASS+**HPS**	.320 (48.3%)	.332 (43.6%)	.445 (40.5%)	.368 (34.8%)	2,641

(c) Spanish (ES)

ES	MAP	R-Prec	P@5	P@10	Rel-ret
No stemming	.379	.361	.513	.430	2,086
Rule	.437 (15.3%)	.428 (18.6%)	.530 (3.3%)	.480 (11.6%)	2,198
ES-unseen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.405 (6.9%)	.395 (9.4%)	.533 (3.9%)	.470 (9.3%)	2,155
HPS (both phases)	.411 (8.4%)	.399 (10.5%)	.543 (5.8%)	.478 (11.2%)	2,179
GRAS	.418 (10.3%)	.412 (14.1%)	.530 (3.3%)	.477 (10.9%)	2,183
GRAS+**HPS**	.421 (11.1%)	.414 (14.8%)	.527 (2.7%)	.470 (9.3%)	2,137
YASS	.400 (5.5%)	.392 (8.6%)	.540 (5.3%)	.463 (7.7%)	2,181
YASS+**HPS**	.413 (8.9%)	.409 (13.4%)	.520 (1.4%)	.455 (5.8%)	2,141
Linguistica	.386 (1.8%)	.384 (6.4%)	.493 (-3.9%)	.437 (1.6%)	2,135
Linguistica+**HPS**	.425 (12.2%)	.420 (16.3%)	.523 (2.0%)	.467 (8.5%)	2,141
ES-seen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.416 (9.8%)	.402 (11.4%)	.525 (2.3%)	.468 (8.8%)	2,180
HPS (both phases)	.424 (11.9%)	.413 (14.4%)	.547 (6.6%)	.482 (12.1%)	2,191
GRAS	.415 (9.5%)	.405 (12.2%)	.527 (2.7%)	.497 (15.6%)	2,193
GRAS+**HPS**	.413 (9.0%)	.402 (11.4%)	.535 (4.3%)	.460 (7.0%)	2,190
YASS	.409 (7.9%)	.398 (10.2%)	.507 (-1.2%)	.465 (8.1%)	2,172
YASS+**HPS**	.405 (6.9%)	.397 (10.1%)	.497 (-3.2%)	.465 (8.1%)	2,168

(d) English (EN)

EN	MAP	R-Prec	P@5	P@10	Rel-ret
No stemming	.320	.316	.378	.316	1,771
Rule	.368 (15.0%)	.348 (10.1%)	.410 (8.5%)	.348 (10.1%)	1,880
EN-unseen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.348 (8.7%)	.338 (7.0%)	.415 (9.8%)	.340 (7.6%)	1,859
HPS (both phases)	.360 (12.5%)	.344 (8.9%)	.412 (9.0%)	.343 (8.5%)	1,873
GRAS	.369 (15.3%)	.349 (10.4%)	.438 (15.9%)	.356 (12.7%)	1,863
GRAS+**HPS**	.365 (14.1%)	.358 (13.2%)	.444 (17.5%)	.353 (11.7%)	1,877
YASS	.352 (10.0%)	.340 (7.6%)	.412 (9.0%)	.339 (7.3%)	1,846
YASS+**HPS**	.350 (9.4%)	.339 (7.2%)	.406 (7.4%)	.339 (7.3%)	1,884
Linguistica	.338 (5.6%)	.336 (6.3%)	.406 (7.4%)	.325 (2.8%)	1,804
Linguistica+**HPS**	.339 (6.0%)	.335 (6.0%)	.414 (9.5%)	.336 (6.3%)	1,874
EN-seen	MAP	R-Prec	P@5	P@10	Rel-ret
HPS (1. phase)	.362 (13.1%)	.347 (9.8%)	.422 (11.6%)	.345 (9.2%)	1,871
HPS (both phases)	.364 (13.8%)	.352 (11.4%)	.414 (9.5%)	.332 (5.1%)	1,873
GRAS	.369 (15.3%)	.348 (10.1%)	.404 (6.9%)	.347 (9.8%)	1,871
GRAS+**HPS**	.364 (13.7%)	.347 (9.8%)	.422 (11.6%)	.335 (6.0%)	1,885
YASS	.361 (12.8%)	.335 (6.0%)	.418 (10.6%)	.344 (8.9%)	1,864
YASS+**HPS**	.364 (13.8%)	.341 (7.9%)	.418 (10.6%)	.347 (9.8%)	1,871

5.5.3. Significance test

In the preceding section, the stemmers were tested in the information retrieval task. There was, however, only a limited number of queries provided for each language. It is therefore crucial to evaluate the differences in results using a statistical significance test. For the evaluation, the paired t-test at a confidence level of 0.95 is used (see [Hull, 1993]). The hypotheses are defined as follows. The preliminary assumption (the null

hypothesis H_0) is that there is no difference between these two stemmers in terms of their stemming quality. The alternative hypothesis H_1 means that one stemmer is significantly better than the other one.

The results of the significance testing are presented in Table 6 by the following symbols:

- Symbol "$<$" if the row's stemmer is worse than the column's stemmer. The hypothesis H_0 is rejected.
- Symbol "$>$" if the row's stemmer is better than the column's stemmer. The hypothesis H_0 is rejected.
- Symbol "$=$" if the row's stemmer is equal to the column's stemmer. The hypothesis H_0 is not rejected.

The p-value is calculated for two measures of performance: for average precision MAP (the first symbols in the table) and for R-precision (the second symbols in the table).

Table 6: Significance tests for comparing the stemmers. The Linguistica results are not presented for *seen* data, as it is not possible to train the available version on a corpus of more than 5,000,000 tokens.

		unseen										seen							
		HPS (1. phase)	**HPS** (both phases)	GRAS	GRAS+**HPS**	YASS	YASS+**HPS**	Linguistica	Linguistica+**HPS**	Rule	No stemming	**HPS** (1. phase)	**HPS** (both phases)	GRAS	GRAS+**HPS**	YASS	YASS+**HPS**	Rule	No stemming
CZ	**HPS** (1. phase)	==	=<	==	=<	==	==	==	==	=<	>=	==	==	==	==	==	==	==	>>
	HPS (both phases)	=>	==	=>	==	>>	==	>>	=>	==	>>	==	==	==	==	==	==	==	>>
	GRAS	==	=<	==	=<	==	==	==	==	=<	>=	==	==	==	==	==	==	==	>>
	GRAS+**HPS**	=>	==	=>	==	=>	==	>=	==	==	>>	==	==	==	==	==	==	==	>>
	YASS	==	<<	==	=<	==	==	==	==	<<	>=	==	==	==	==	==	==	==	>>
	YASS+**HPS**	==	==	==	==	==	==	>=	==	==	>=	==	==	==	==	==	==	==	>>
	Linguistica	==	<<	==	<=	==	<=	==	<=	<<	>=								
	Linguistica+**HPS**	==	=<	==	==	==	==	>=	==	==	>>								
	Rule	=>	==	=>	==	>>	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	No stemming	<=	<<	<=	<<	<=	<=	<=	<<	<<	==	<<	<<	<<	<<	<<	<<	<<	==
HU	**HPS** (1. phase)	==	<<	==	<<	==	<<	>>	<<	<<	>>	==	==	==	==	==	==	==	>>
	HPS (both phases)	>>	==	>>	==	>>	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	GRAS	==	<<	==	<<	==	<<	>>	<<	<<	>>	==	==	==	==	==	==	==	>>
	GRAS+**HPS**	>>	==	>>	==	>>	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	YASS	==	<<	==	<<	==	<<	==	<<	<<	>>	==	==	==	==	==	==	==	>>
	YASS+**HPS**	>>	==	>>	==	>>	==	>>	>>	==	>>	==	==	==	==	==	==	==	>>
	Linguistica	<<	<<	<<	<<	==	<<	==	<<	<<	>>								
	Linguistica+**HPS**	>>	==	>>	==	>>	<<	>>	==	==	>>								
	Rule	>>	==	>>	==	>>	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	No stemming	<<	<<	<<	<<	<<	<<	<<	<<	<<	==	<<	<<	<<	<<	<<	<<	<<	==
ES	**HPS** (1. phase)	==	==	==	==	==	==	>=	==	<<	>>	==	==	==	==	==	==	==	>>
	HPS (both phases)	==	==	==	==	==	==	>=	==	<<	>>	==	==	==	==	==	==	==	>>
	GRAS	==	==	==	==	==	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	GRAS+**HPS**	==	==	==	==	==	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	YASS	==	==	==	==	==	==	==	<=	<=	=>	==	==	==	==	==	==	==	=>
	YASS+**HPS**	==	==	==	==	==	==	>>	==	==	>>	==	==	==	==	==	==	==	=>
	Linguistica	<=	<=	<<	<<	==	<<	==	<<	<<	=>								
	Linguistica+**HPS**	==	==	==	==	>=	==	>>	==	==	>>								
	Rule	>>	>>	==	==	>=	==	>>	==	==	>>	==	==	==	==	==	==	==	>>
	No stemming	<<	<<	<<	<<	=<	<<	=<	<<	<<	==	<<	<<	<<	<<	=<	=<	<<	==
EN	**HPS** (1. phase)	==	==	<=	<=	==	==	==	==	==	>>	==	==	==	==	==	==	==	>>
	HPS (both phases)	==	==	==	==	==	==	==	==	==	>>	==	==	==	==	==	==	==	>>
	GRAS	>=	==	==	==	>=	>=	>=	>=	==	>>	==	==	==	==	==	==	==	>>
	GRAS+**HPS**	>=	==	==	==	>=	>=	>=	>>	==	>>	==	==	==	==	==	==	==	>>
	YASS	==	==	<=	<=	==	==	==	==	==	>=	==	==	==	==	==	==	==	>=
	YASS+**HPS**	==	==	<=	<=	==	==	==	==	==	>=	==	==	==	==	==	==	==	>=
	Linguistica	==	==	<=	<=	==	==	==	==	<=	==								
	Linguistica+**HPS**	==	==	<=	<<	==	==	==	==	<=	==								
	Rule	==	==	==	==	==	==	>=	>=	==	>>	==	==	==	==	==	==	==	>>
	No stemming	<<	<<	<<	<<	<=	<=	==	==	<<	==	<<	<<	<<	<<	<=	<=	<<	==

From Table 6, we can deduce the behavior of all stemmers in the information retrieval context. In the case of seen data, it was discovered that all tested stemmers perform equally well (there is no significant difference between them).

In the case of unseen data, HPS (both phases), GRAS+HPS, YASS+HPS and rule-based stemmers perform the best. For less inflected languages (ES and EN) the performance of GRAS and YASS is on the same level, but for highly inflected languages (CZ and HU) their performance is significantly worse.

This is expected behavior, because many word forms are previously unseen, leading to a significant OOV (out-of-vocabulary) rate. These facts prove that our second stage of HPS is able to work very well with unknown word forms.

In both the cases of seen and unseen data, all evaluated stemmers significantly improve the results of the IR system when compared with no stemming. If we summarize all these results, HPS (both phases), GRAS+HPS, YASS+HPS, and rule-based stemmers consistently give the best results even though HPS was not designed purely for the IR task, but rather to be a multi-purpose stemmer. GRAS and YASS perform slightly worse and Linguistica is the least efficient stemmer for the IR task.

5.6. Language models

This section presents experiments with the application of stemmers to language modeling. The purpose is to reveal the performance of stemmers in yet another scenario.

Language modeling is a crucial task in many areas of NLP. Speech recognition, optical character recognition, machine translation, information retrieval, and many other areas depend heavily on the quality of the language model that is being used. Each improvement in language modeling can also improve the particular job where the language model is used.

Morphological information has already been proved to be useful in language modeling. For example, in [Brychcín and Konopík, 2011], we use lemmatization and part-of-speech (POS) tags to significantly improve the perplexity of Czech and Slovak language models. In this section, we present a similar approach, but instead of lemmas we used stems, and instead of POS tags we used suffixes.

5.6.1. Architecture

We choose class-based language models as the architecture for incorporating the morphological information. We have derived two kinds of class-based models:

- **Stem:** word classes represent the words with the same stem.
- **Inflection:** words with the same inflection (suffix following the stem) are grouped into one class.

We use the *modified Kneser–Ney interpolation* [Chen and Goodman, 1998] for smoothing the baseline n-gram language model as well as the stand-alone class-based models. The order (n) of all models is 3.

These two class-based models are combined with the baseline model by bucketed linear interpolation [Bahl et al., 1983]:

$$P^{BLI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k\left(w_{i-n+1}^{i-1}\right) \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right), \tag{23}$$

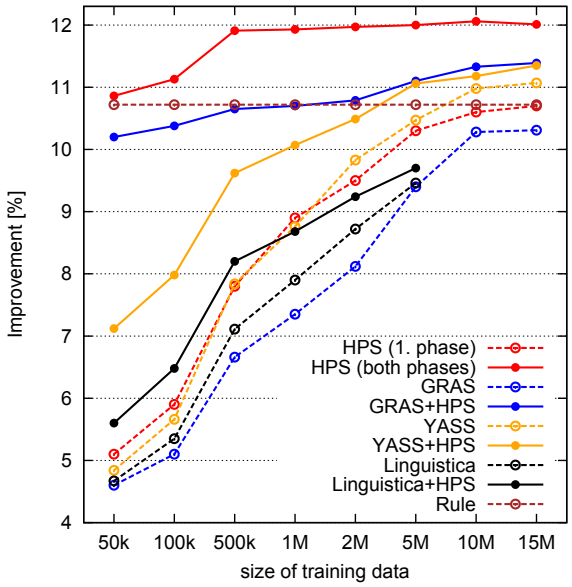
where $\lambda_k()$ is the weight of the kth language model, $P_k()$. We use the EM (Expectation Maximization) algorithm described in [Dempster et al., 1977] to calculate the optimal weights λ_k by maximizing the probability of the held-out data. In bucketed interpolation, the weights are functions of the frequency of word history. The main idea behind the interpolation is that the weights λ_k should be different for words with histories of different frequencies. In our experiments, 20 buckets were used.

5.6.2. Results

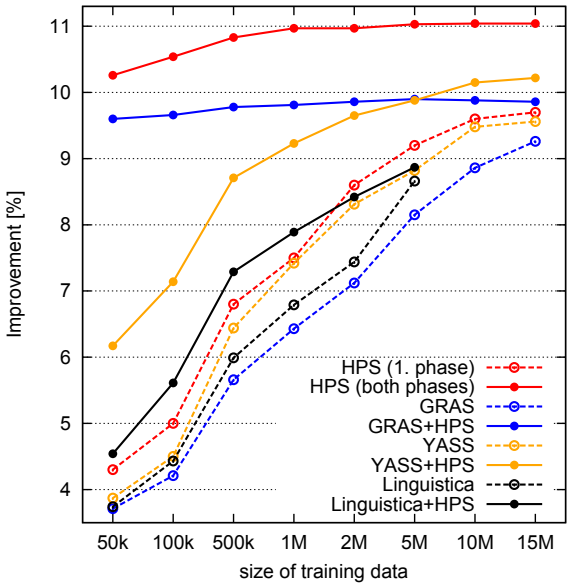
The performance of a language model is typically measured in terms of the perplexity of the model on the unseen test corpus. The perplexity can be seen as the confusion of the model. Lower perplexity means a better prediction ability of language model. It was shown by many authors that the reduction of perplexity often leads to improving whole system where the language model is used (for example machine translation in [Brychcín and Konopík, 2014] or speech recognition in [Watanabe et al., 2011]).

The stemmers were trained on the same data as during the inflection removal experiments (see Section 5.3), this means on 50k, 100k, 500k, 1M, 2M, 5M, 10M, and 15M tokens. Each language model was trained on 15M tokens. An additional 2M tokens were used as held-out data. Then, an additional 5M tokens were used to calculate the perplexity of the language model.

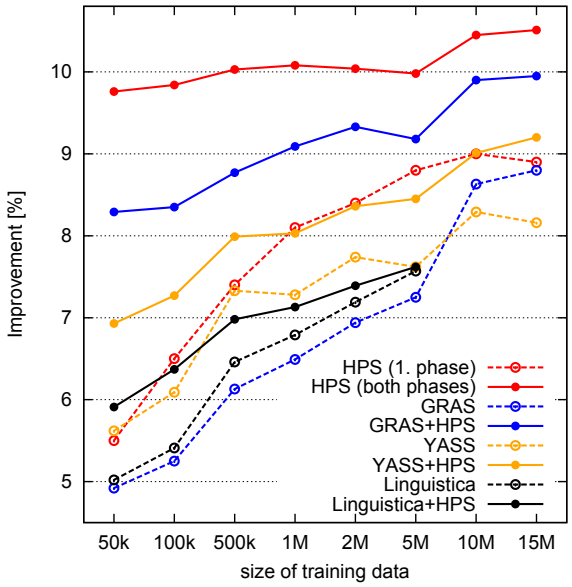
Figure 3 shows the improvement in perplexity when compared to the baseline language model.



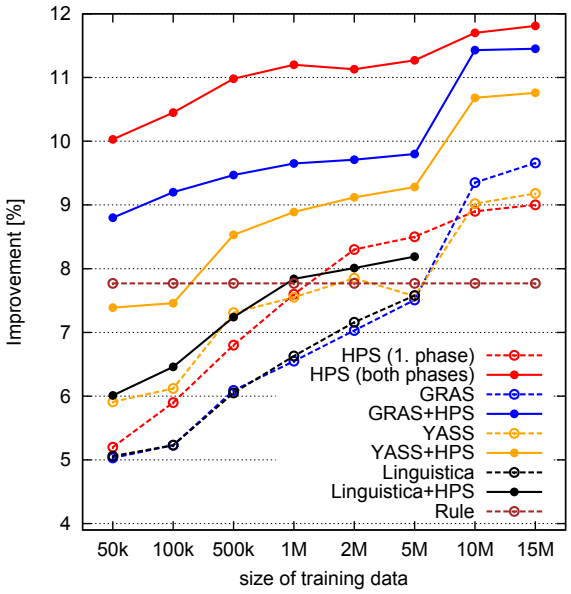
(a) Czech (CZ): baseline perplexity 472.



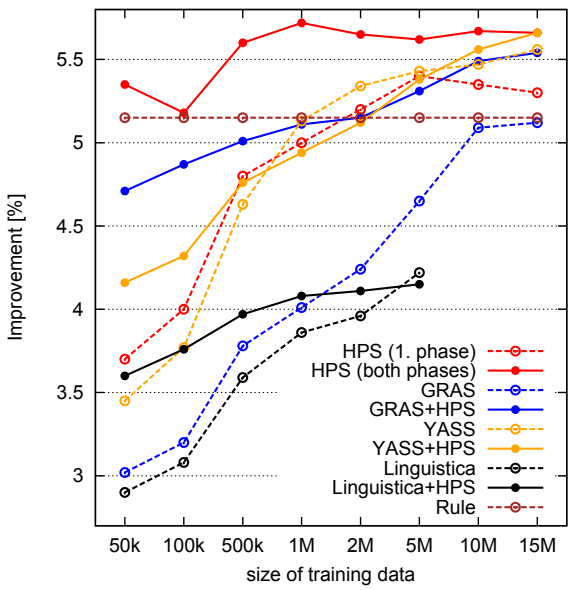
(b) Slovak (SK): baseline perplexity 527.



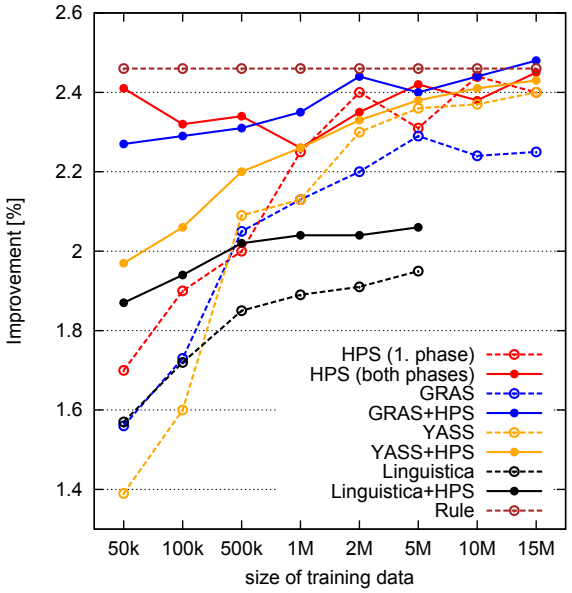
(c) Polish (PL): baseline perplexity 153.



(d) Hungarian (HU): baseline perplexity 188.



(e) Spanish (ES): baseline perplexity 124.



(f) English (EN): baseline perplexity 238.

Figure 3: Perplexity improvements compared to the baseline language model.

The experiments with language modeling confirm the conclusions of the preceding experiments. Again, HPS obtains on average the highest perplexity drops. GRAS and YASS swapped the order of their results. In this test, YASS tends to perform better than GRAS. For smaller training data sizes, HPS has no competitor. The second stage of HPS again produces large improvements especially when little training data is used.

5.7. Performance tuning

In this section, we investigate some possible tweaks which decrease the computational costs of the stemming. These tweaks have virtually no impact on the stemming results.

Firstly, it must be remembered that our modification of maximum mutual information clustering (described in Section 4.1) gives accurate results only for words which occur frequently enough in a corpus. The infrequent words have a negligible impact on the average mutual information of the data, and their clustering may be very inaccurate, and can even decrease the quality of the whole stemmer. We recommend clustering only words with a frequency higher than some threshold (for example 10) by the MMI algorithm. The remaining words should be clustered using only lexical information (presented in Section 4.1.1). Moreover, the complexity of the clustering increases with the square of the number of words being clustered. Thus, limiting the words being clustered has a positive impact on the processing time.

An additional acceleration of our algorithm is possible by incorporating just enough occurring word bigrams in the calculation of the average mutual information of the data. We recommend incorporating word bigrams that occur at least 2 or 3 times in the training corpus. This also leads to a significant speeding up of the clustering, while the efficiency of the final stemmer stays almost the same.

We would like to point out that setting these parameters has no impact on the quality of the stemming results. Setting these parameters is not mandatory.

For the sake of a better grasp of these efficiencies, we also present the running times needed to train HPS, as well as that needed for the stemming itself. We implemented our HPS method on the Java™ platform, and tested it on a computer with a Core i7 3.4 GHz processor. Training the model on 5M tokens of Czech data (the same model used for the experiments with the above recommended settings, i.e., word frequency at least 10 and word pair frequency at least 2) takes about 36 min. The stemming of 10M tokens of text with this model takes about 33 sec. These results clearly show that once we already have a trained model, the stemming itself is very fast.

6. Discussion

In Section 5, we thoroughly tested our HPS from three different perspectives. To put the performance of HPS into the context of the state of the art, we also provided a comparison with other competitive stemmers (namely GRAS, YASS, Linguistica, and rule-based stemmers). In addition, we extended these competitive stemmers with our second stage of HPS (maximum entropy classifier). During our experiments, we also measured the amount of training data needed to give satisfactory results.

The first experiment (Section 5.4) was focused on evaluating how well the stemmers remove the inflection of words by comparing classes with the same stem with classes with the same lemma obtained from manually annotated data. HPS was at the top for all languages, only for Spanish and Hungarian did GRAS performed equally well.

The second experiment (Section 5.5) investigated the use in an information retrieval task. Retrieval effectiveness was tested on Czech, Hungarian, Spanish, and English, for both previously seen and unseen data. Significance testing was done to make the results more appropriate. It was discovered that our stemmer provides significantly better retrieval scores than competitive stemmers in the case of indexing new (unseen) data. HPS tends to be more suitable for languages with rich morphology (Czech, Hungarian) than other stemmers, because of the significant OOV (out-of-vocabulary) rate. For Spanish and English the significance testing proved that there is no significant difference between stemmers for both seen and unseen data. However, such results could have been easily predicted since the stemming of less inflected languages does not play as important a role in the IR task as the stemming of highly inflected languages.

The last experiment (Section 5.6) examined the effect of using stemming in language modeling. Class-based models were derived from stemming results and they were coupled with a baseline n-gram language model. Again, for highly inflected languages, HPS performed best of all. For less inflected languages, HPS is on the same level as YASS, but again, stemming does not play a key role here. It is interesting that GRAS, which in previous experiments performed very well, gives one of the worst results in language modeling. We suppose this is caused by the fact that GRAS is too focused on the recall rate and so it often overstems the words.

We explain the superior performance of HPS by its building a stemmer in two stages. The first stage uses the idea of involving latent semantic information in the stemming task. Other approaches deal with the problem on a purely lexical level. By involving semantic information, our stemmer can better decide about the appropriateness of removing different suffixes. The suffixes that can alter the semantics of the words should be left intact in our approach. For example, compare the words *spar* and *spar-ing*. From the lexical point of view, there is no reason to leave the suffix *ing* intact. However, from the semantic point of view, *spar* and *spar-ing* are completely different words. Naturally, semantic information retrieved from the statistical comparison of word contexts is not flawless. Nevertheless, the increased performance of HPS indicates that latent semantics is beneficial in the stemming task. Although, the idea to build a stemmer in two stages is not new (in [Xu and Croft, 1998], the co-occurrence statistics were used to refine equivalence classes given by some aggressive stemmer, decreasing number of overstemming errors), our second stage goes deeper. It is not limited to work with aggressive stemmers only. The second stage of HPS extracts rules from the word clusters given by the first stage, and combines these rules using a maximum entropy classifier. These rules were proved by our experiments to be very general and efficient, because HPS is able to stem previously unseen word forms.

The preceding paragraph relates to the question of whether a stemmer should or should not remove derivational suffixes. Firstly, it again depends on the task. As we already illustrated, reducing the word *friendly* to the word *friend* may be useful for IR but not for machine translation. Secondly, the question also relates to the semantics. We can generally say that we should remove a suffix only in cases where they do not change the meaning of the stemmed word. We believe that employing some semantic information is appropriate, given this perspective.

By taking all the results into account, HPS seems to be the most effective and most universal approach for stemming aimed at languages with rich morphology. GRAS is very efficient in IR tasks and even performs well in inflection removal experiments. YASS provides consistently good results and so it is also very universal. Linguistica was not as efficient as the other stemmers in our tests.

A very positive fact proved by our experiments is that HPS requires only a small amount of training data to give very satisfactory results. This property is caused by our second stage of HPS, the maximum entropy classifier. It was shown that this second stage also improves other stemmers: they are denoted by GRAS+HPS, YASS+HPS, and Linguistica+HPS, when supplemented by this second stage. In this regard, HPS has no competitor. Other stemmers perform poorly if they have little data for training. Our experiments show that a corpus of only 50,000 tokens is sufficient for efficiently training HPS. The corpora of 1,000,000 tokens seem to be more than enough for training, because when increasing the amount of the training data, the results do not improve significantly. The explanation is as follows. In our experiments, we classify only to 4 classes (4 possible lengths of suffix, i.e., from 0 to 3 characters) and we use only a few feature functions, which is something that leads to needing only a small number of parameters to be estimated from the training data. The rules formulating the various endings of word forms are supposed to be common and often repeated in a corpus, and this is the reason why HPS performs well even with as small a training dataset as 50,000 tokens. These facts also explain why the performance of HPS does not improve as much as in the case of other stemmers with increasing amounts of training data.

It may seem that in an era of vast linguistic resources, such a property would be insignificant. However, we would like to point to the idea presented in [Hammarström and Borin, 2011]. There are a huge number of languages that have a very limited number of speakers. For such languages, rich linguistic resources are not obtainable. Our stemmer should be successful particularly in this area. We plan to target these languages in the future.

As stated at the beginning of our paper, precision is preferred over recall in our approach. Thus, it is possible to call our stemmer *light*. It was proven (for example in [Dolamic and Savoy, 2009; Savoy, 2008]) that aggressive stemmers usually perform better in the retrieval context than light stemmers. By more aggressive stemming, the recall rate is increased and the size of the storing index is decreased at the same time. However, it was not our aim to create an unsupervised stemmer focused solely on information retrieval, but to design a multi-purpose stemmer performing well in multiple scenarios without any modification.

7. Summary

7.1. Contributions

The contributions of the paper are the following:

- We present a new approach to stemming that has a very universal scope. It outperforms the state of the art in unsupervised ways of stemming in the inflection removal test, the information retrieval test with unseen data, and the language modeling task. In the information retrieval test with seen data, it provides comparable results with the state of the art.
- Our proposed method is by far the most effective one when little training data are available.
- We provide tests on four language families (six different languages).
- In the article, we deal with several aspects of the stemming problem, such as the difference between removing inflectional and derivational suffixes, and the relation of stemming to lemmatization.
- We also introduce a novel evaluation method for comparing stemmers, which improves on the method presented in [Paice, 1994] in several ways.

7.2. Future work

In future work we would like to focus on the analysis of words where the inflected forms are formed by changing significant parts of the words, not just suffixes. This is essentially the weakness of all stemmers. A representative example are the irregular verbs in English. In most languages, the verbs in different tenses often differ in large parts of the words.

We suppose that the rules causing these inflections often repeat in natural language texts and so there has to be a way to discover them. We plan to design more elaborate rules for finding candidates for words

with the same stem. In this way, many of the current stemming mistakes can be resolved, thus enhancing the performance of our stemmer.

Another possibility for future work is to investigate different ways of modeling semantic relations. Semantic spaces, which are quite a new branch of corpus statistics, could be used during clustering (Section 4.1) instead of the mutual information loss algorithm. Note that we already experimented with semantic spaces for improving language modeling (see [Brychcín and Konopík, 2014]).

Also, we plan to target languages with a limited number of speakers and limited linguistic resources in order to prove that our stemmer is suitable for them.

Finally, we would also want to test our stemmer in other scenarios. We have already provided the stemmer to our colleagues. In [Habernal et al., 2013; Habernal and Brychcín, 2013; Habernal et al., 2014; Steinberger et al., 2014], they discovered that our stemmer significantly improves sentiment analysis. Our preliminary results indicate that HPS is also very useful (and significantly better than competing stemmers) in named entity recognition and machine translation tasks.

7.3. Conclusion

In this article we presented a very effective stemming method that further shifts the boundaries of the current state of the art in unsupervised stemming. We successfully accomplished the goal of creating a multi-purpose stemming tool. Its design opens up possibilities for solving non-traditional tasks, such as approximating lemmas, improving language modeling or sentiment analysis. However, it still provides very good results in the traditional task, information retrieval.

Our approach learns morphological rules from an unannotated corpus without any knowledge about the language or any additional information. A clustering method that discovers semantically related words creates the basis for learning the stemming rules. These rules successfully approximate the morphology of a language and can be used even for unknown (previously unseen) word forms. Our experiments show that the stemming of unknown words is as effective as the stemming of known words, which is essentially one of the greatest advantages of our stemmer compared with other competitive stemmers.

The second very positive property of our approach is that it does not require a huge amount of training data. Our experiments confirm that for successful training, a corpus of only one million tokens is sufficient. Very satisfactory results can be, however, achieved with only 50,000 tokens for training, where other stemmers fail. This property could be very important, mainly for poor-resource languages.

Even in the cases where other stemmers have enough data for training, HPS does not lose. The comparison with other stemmers (namely GRAS, YASS, Linguistica as well as with rule-based stemmers) was done on several languages, including highly inflected languages as well as less inflected languages. In the stemming of less inflected languages, there is no significant difference between the stemmers that have been tested. The stemming however play a key role for highly inflected languages, where our stemmer is significantly better than competing unsupervised stemmers and approximately on the same level as rule-based stemmers. Our HPS implementation is available at `https://liks.fav.zcu.cz/HPS`.

Acknowledgements

This work was supported by grant no. SGS-2013-029 Advanced computing and information systems, by the European Regional Development Fund (ERDF) and by project "NTIS - New Technologies for Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090. Access to the MetaCentrum computing facilities provided under the program "Projects of Large Infrastructure for Research, Development, and Innovations" LM2010005, funded by the Ministry of Education, Youth, and Sports of the Czech Republic, is highly appreciated. The access to the CERIT-SC computing and storage facilities provided under the programme Center CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, reg. no. CZ. 1.05/3.2.00/08.0144 is acknowledged. We also thank the Czech News Agency for providing a huge number of texts in Czech.

References

Amati, G., Van Rijsbergen, C. J., Oct. 2002. Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Transactions on Information Systems 20 (4), 357–389.

Andrew, G., Gao, J., 2007. Scalable training of L1-regularized log-linear models. In: Proceedings of the 24th International Conference on Machine Learning. ACM, New York, pp. 33–40.

Bacchin, M., Ferro, N., Melucci, M., January 2005. A probabilistic model for stemmer generation. Information Processing and Management 41, 121–137.

Bahl, L. R., Jelinek, F., Mercer, R. L., 1983. A maximum likelihood approach to continuous speech recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5 (2), 179–190.

Berger, A. L., Pietra, V. J. D., Pietra, S. A. D., Mar. 1996. A maximum entropy approach to natural language processing. Computational Linguistics 22, 39–71.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., Lai, J. C., 1992. Class-based n-gram models of natural language. Computational Linguistics 18, 467–479.

Brychcín, T., Konopík, M., 2011. Morphological based language models for inflectional languages. In: Proceedings of IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems.

Brychcín, T., Konopík, M., 2014. Semantic spaces for improving language modeling. Computer Speech & Language 28 (1), 192 – 209.

Charles, W. G., 2000. Contextual correlates of meaning. Applied Psycholinguistics 21 (04), 505–524.

Chen, S. F., Goodman, J. T., 1998. An empirical study of smoothing techniques for language modeling. Tech. rep., Computer Science Group, Harvard University.

Dempster, A. P., Laird, N. M., Rubin, D. B., 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B 39 (1), 1–38.

Dolamic, L., Savoy, J., November 2009. Indexing and stemming approaches for the Czech language. Information Processing and Management 45, 714–720.

Goldsmith, J., Jun. 2001. Unsupervised learning of the morphology of a natural language. Computational Linguistics 27, 153–198.

Goldsmith, J., December 2006. An algorithm for the unsupervised learning of morphology. Natural Language Engineering 12, 353–371.

Habernal, I., Brychcín, T., 2013. Semantic spaces for sentiment analysis. In: Habernal, I., Matoušek, V. (Eds.), Text, Speech, and Dialogue. Vol. 8082 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 484–491.

Habernal, I., Ptáček, T., Steinberger, J., June 2013. Sentiment analysis in Czech social media using supervised machine learning. In: Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. Association for Computational Linguistics, Atlanta, Georgia, pp. 65–74.

Habernal, I., Ptáček, T., Steinberger, J., 2014. Supervised sentiment analysis in czech social media. Information Processing & Management 50 (5), 693 – 707.

Hammarström, H., Borin, L., 2011. Unsupervised learning of morphology. Computational Linguistics 37, 309–350.

Huddleston, R., 1988. English Grammar: An Outline. Cambridge University Press.

Hull, D., 1993. Using statistical testing in the evaluation of retrieval experiments. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, pp. 329–338.

Koehn, P., Hoang, H., 2007. Factored translation models. In: EMNLP-CoNLL. pp. 868–876.

Konkol, M., 2014. Brainy: A machine learning library. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L. A., Zurada, J. M. (Eds.), Artificial Intelligence and Soft Computing. Vol. 8468 of Lecture Notes in Computer Science. Springer International Publishing, pp. 490–499.

Kroeger, P., 2005. Analyzing Grammar. Cambridge University Press.

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

Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P., Datta, K., October 2007. YASS: Yet another suffix stripper. ACM Transactions on Information Systems 25.

Oard, D. W., Levow, G., Cabezas, C. I., 2001. CLEF experiments at Maryland: Statistical stemming and backoff translation. In: Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation. Springer-Verlag, London, pp. 176–187.

Paice, C. D., 1994. An evaluation method for stemming algorithms. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Springer-Verlag, New York, pp. 42–50.

Paik, J. H., Mitra, M., Parui, S. K., Järvelin, K., Dec. 2011a. GRAS: An effective and efficient stemming algorithm for information retrieval. ACM Transactions on Information Systems 29, 19:1–19:24.

Paik, J. H., Pal, D., Parui, S. K., 2011b. A novel corpus-based stemming algorithm using co-occurrence statistics. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, pp. 863–872.

Porter, M. F., 1980. An Algorithm for Suffix Stripping. Program 14 (3), 130–137.

Rubenstein, H., Goodenough, J. B., Oct. 1965. Contextual correlates of synonymy. Communications of the ACM 8 (10), 627–633.

Savoy, J., 2008. Searching strategies for the Hungarian language. Information Processing & Management 44 (1), 310–324.

Steinberger, J., Brychcín, T., Konkol, M., June 2014. Aspect-level sentiment analysis in czech. In: Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. Association for Computational Linguistics, Baltimore, Maryland, pp. 24–30.

Taulé, M., Martí, M. A., Recasens, M., 2008. AnCora: Multilevel annotated corpora for Catalan and Spanish. In: Proceedings of the 6th International Conference on Language Resources and Evaluation. European Language Resources Association, Marrakech, Morocco.

Watanabe, S., Iwata, T., Hori, T., Sako, A., Ariki, Y., April 2011. Topic tracking language model for speech recognition. Computer Speech & Language 25, 440–461.

Xu, J., Croft, W. B., January 1998. Corpus-based stemming using cooccurrence of word variants. ACM Transactions on Information Systems 16, 61–81.