University of West Bohemia
Faculty of Applied Sciences

Distributional Semantics in Language Modeling

Ing. Tomáš Brychcín

Doctoral Thesis
Submitted in partial fulfillment of the requirements
for a degree of Doctor of Philosophy
in Computer Science and Engineering

Supervisor: Ing. Roman Mouček, Ph.D.
Consulting Specialist: Ing. Miloslav Konopík, Ph.D.
Department of Computer Science and Engineering

Plzeň, 2015

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Distribuční sémantika v jazykovém modelování

Ing. Tomáš Brychcín

Disertační práce
k získání akademického titulu doktor v oboru Informatika a výpočetní technika

Školitel: Ing. Roman Mouček, Ph.D.
Konzultant–specialista: Ing. Miloslav Konopík, Ph.D.
Katedra informatiky a výpočetní techniky

Plzeň 2015

Prohlášení

Předkládám tímto k posouzení a obhajobě disertační práci zpracovanou na závěr doktorského studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji tímto, že jsem tuto práci vypracoval samostatně, s použitím odborné literatury a dostupných pramenů uvedených v seznamu, jenž je součástí této práce.

V Plzni dne 16. února 2015 Ing. Tomáš Brychcín

Acknowledgement

First of all, I would like to thank Dr. Roman Mouček and Dr. Miloslav Konopík, my supervisors, for their guidance, encouragement, and support.

This doctoral thesis would arguably never exist without numerous stimulating discussions and great amount of valuable comments from my colleagues, Michal Konkol, Dr. Ivan Habernal, Dr. Pavel Král, Dr. Josef Steinberger, and Dr. Kamil Ekštein.

I also thank my lab colleagues and friends Michal Konkol, Tomáš Hercig, and Lukáš Svoboda for friendly atmosphere in the lab and that they have always let me win (at least they say so) chess games against them after each lunch.

Last but not the least, I would like to express my deepest gratitude to my parents, Milan and Libuše, to my sister Petra, and to my girlfriend Lenka for love, support, inspiration, and especially patience with me during my study and during the writing of this thesis.

Abstract

Language models are crucial for many tasks in natural language processing and n-grams are probably the best way to build them. Huge effort is being invested in improving the n-gram language models. By introducing external knowledge (morphology, syntax, etc.) into the models, a significant improvement can be achieved. The models can, however, be improved without external knowledge and the better smoothing is an excellent example of such improvement. By discovering hidden patterns in unlabeled training corpora, we can enhance the language modeling with the information that is already present in the corpora.

This thesis studies three different ways of latent information discovery. Global semantics is modeled by latent Dirichlet allocation and brings long-range dependencies into language models. Word clusters given by semantic spaces enrich these language models with short-range semantics. Finally, our own unsupervised stemming algorithm is used to further enhance the performance of language modeling for inflectional languages.

Our research shows that these three sources of information enrich each other and their combination dramatically improves language modeling. All investigated models are acquired in a fully unsupervised manner. We show the efficiency of our methods for several languages within different language families, proving their multilingual properties.

Abstrakt

Jazykové modely jsou důležitou součástí mnoha úloh ve zpracování přirozeného jazyka a n-gramy jsou pravděpodobně nejlepší způsob jak je vytvořit. Vylepšování n-gramových jazykových modelů bylo věnováno značné úsilí. Použitím externí informace (morfologie, syntaxe, apod.) v těchto modelech může dojít k výraznému vylepšení. Tyto modely však mohou být vylepšeny i bez externí informace a efektivnější vyhlazování je reprezentativní příklad takového vylepšení. Pokud pochopíme skryté vzory v neoznačkovaných korpusech, můžeme zvýšit kvalitu jazykového modelování pouze s informací, která je již v těchto korpusech přítomna.

Tato práce se zabývá třemi různými směry odkrývání latentní informace. Globální sémantika je modelována pomocí Latentní Dirichletovy alokace a zahrnuje globální relace do jazykových modelů. Slovní třídy, získané pomocí sémantických prostorů, obohacují tyto jazykové modely o lokální sémantiku. Nakonec je použit náš vlastní stemovací algoritmus, založený na trénování bez učitele, který ještě navyšuje výkonnost jazykových modelů u flektivních jazyků.

Náš výzkum ukazuje, že tyto tři zdroje informací se obohacují navzájem a že jejich kombinace vede ke dramatickému vylepšení jazykových modelů. Všechny zkoumané modely jsou trénované bez učitele. Ukazujeme účinnost našich modelů na několika jazycích různých typů, což prokazuje nezávislost na konkrétním jazyce.

Contents

Chapter 1

Introduction

"No one ever won a game by resigning"

— Saviely Tartakower

Language modeling is a crucial task in many areas of natural language processing (NLP). Automatic speech recognition (ASR), optical character recognition (OCR) and many other areas heavily depend on the performance of the language model that is being used. Each improvement in language modeling may also improve the particular task where the language model is used.

Research into language modeling has started more than 20 years ago and has evolved into a mature discipline. Now it is very difficult to outperform the state-of-the-art techniques. Many studies have suggested improving n-gram language models by adding external information (morphology, syntax, etc.) about the language and significant improvements have been achieved. Such approaches supervised by linguists (or at least native speakers) are likely to be very efficient because they are based upon the prior knowledge of the language. There are, however, several disadvantages. It is not always possible to have an expert in this field. Creation of linguistic rules or manual annotation of data is an expensive and time consuming process.

In recent years the unsupervised methods have become popular. These methods usually require more complex techniques, however, they ensure a negligible cost of modeling an additional language (compared to the supervised methods).

In this thesis we explore fully unsupervised methods for language modeling (which require no labeled data and no information about the language itself). We study three different ways to improve language modeling. We focus on distributional semantics, a research area based upon distributional hypothesis that investigates (up to some degree) the semantics of the natural text. We believe that the semantic analysis is an essential part of NLP.

Semantic models in this work can be roughly divided into local semantics and global semantics models. The global semantics depicts long range dependencies among words while the local semantics models are used to achieve the better smoothing of the n-gram probabilities. We also developed a novel unsupervised stemmer and used it to deal with morphologically complicated languages.

To prove the multilingual properties of our models we experimented with several languages from different language families.

The majority of languages we investigate in this thesis is characterized by a high level of inflection and relatively free word order (from the syntactic point of view, the words in a sentence can usually be reordered in several ways to carry a slightly different meaning). Such language properties complicate the language modeling task. The great number of word forms and large number of possible word sequences lead to a much higher number of n-grams.

Data sparsity is a common problem of language models but for highly inflected languages this problem is even more significant. The highly inflected languages in this thesis belong rather among non-mainstream languages for which the language modeling task has not gained as much attention as it has for English, for example. We thus believe there is a considerable potential for improvement. However, we experiment even with English for mutual comparison with the state of the art.

This thesis is written in the form of thesis by publications because we believe the presented articles carry a complete solution of the thesis goals (see next section).

1.1 Thesis goals

The main goal of this doctoral thesis is to propose novel unsupervised methods for improving the performance of language models with special emphasis on inflectional languages. The work will be focused on the following research tasks:

- Deal with specific properties of the Czech language and other inflectional languages. Study the relationship between morphology and language modeling.
- Use semantic information to improve language modeling. Unsupervised training is preferred.
- Focus on the analysis of out-of-vocabulary words. Explore both the unsupervised morphological analysis and the analysis of the context.

1.2 Thesis overview

This work is conceived as a thesis by publications.

We start with a short introduction to the presented matter. Chapter 2 describes the basic principles of language modeling and the problems that the language mod-

els are facing. Chapter 3 introduces the research area based upon distributional hypothesis, studying the ways how to derive the word meaning from a raw text. These two chapters serve as the theoretical basis for better understanding the thesis publications.

In Chapter 4, we present the thesis publications, the motivation for them, and their contributions. In Chapter 5 we summarize the thesis, we discuss the fulfillment of the thesis goals, and we outline some future research directions.

The thesis publications, in the form they were accepted for publication, are included in Appendix A.

All author's publications, written during his doctoral study, are listed in Appendix B.

Chapter 2

Language models

"Language is a process of free creation; its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation. "

— Noam Chomsky

The goal of a language model is to estimate the probability of any word string possible in the language. The task may look simple, however, a satisfactory solution for a natural language is very complicated.

2.1 Statistical language models

In this chapter, we will consider the language as an information source that produces word sequences from some vocabulary. Let W denote the word vocabulary. The $W^{\mathbb{N}}$ is the set of all combinations of word sequences that are possible to be created from the vocabulary W. Let

$$\mathcal{L} \subseteq W^{\mathbb{N}} \tag{2.1}$$

be a set of all possible word sequences in a language.

The sequence of words (i.e. a sentence) can be expressed as

$$S = w_1^l = w_1, \cdots, w_l, \qquad S \in \mathcal{L}, \tag{2.2}$$

where w_i denotes the word at position i and l denotes the length of the sequence.

The language model tries to capture the regularities of a natural language by setting constraints on sequences S. These constraints can be either *deterministic* (some sequences are possible, some are not) or *probabilistic* (some sequences are more probable than the others).

In this work we will look at the language modeling from the probabilistic point of view, where the main goal is to estimate the probability $P(S)$ of occurrence of a word string S.

The only way to calculate this probability correctly should be to process all utterances (that have ever been written, spoken, etc.) of that language $\mathcal{L}$ and calculate the frequency of this sequence over all possibilities. At the first glance this is impossible. Actually, it is even more complicated because the natural language is an evolving process. New expressions regularly enter the language while others die out. Thus the word string probabilities also change across the time.

Through the nature of problems mentioned above, it is clear that we can only estimate these probabilities from the training data as large as possible. In the following text, the probability estimation of $P(S)$ will be referred to as $\tilde{P}(S)$:

$$P(S) \approx \tilde{P}(S). \tag{2.3}$$

By application of *chain rule* the probability $P(S)$ can be decomposed into the product of conditional probabilities

$$P(S) = P(w_1^l) = P(w_1)P(w_2|w_1)\cdots P(w_l|w_1^{l-1}) = \prod_{i=1}^{l} P(w_i|w_1^{i-1}). \tag{2.4}$$

2.2 Evaluation

In order to be able to compare two different language models, we need to have a qualitative measure for these language models. The best way how to do it should be to evaluate the whole system where the language model is used. However, it is not always possible to measure the performance of the whole system, thus we must determine a measure for evaluation of a stand-alone language model without any other part of the whole system.

The most often used measures for language models are *entropy* H and *perplexity* PP, which are based upon the information theory. The information theory measures the amount of information by the entropy of its source (i.e. of the language in our case). Entropy of a language is defined as

$$H(P) = -\lim_{l\to\infty} \frac{1}{l} \sum_{w_1^l \in \mathcal{L}} P(w_1^l) \log_2 P(w_1^l). \tag{2.5}$$

In the information theory, the term *Shannon entropy* is sometimes used and it quantifies the expected value of information contained in a message (if the base of the logarithm is 2, the entropy is measured in bits). In terms of language modeling, the entropy represents the average number of bits needed to encode each word in the text. The entropy is a measure of uncertainty. Lower entropy means that the appropriate word in the text is more predictable on average.

There are of course few problems. We do not know the correct probabilities $P(S)$ of the language, therefore we can only estimate the entropy. *Cross-entropy* is defined as

$$H(P, \tilde{P}) = -\lim_{l\to\infty} \frac{1}{l} \sum_{w_1^l \in \mathcal{L}} P(w_1^l) \log_2 \tilde{P}(w_1^l). \tag{2.6}$$

According to *Shannon-McMillan-Breiman theorem* (Cover and Thomas, 1991), this formula can be simplified into

$$H(P, \tilde{P}) = -\lim_{l\to\infty} \frac{1}{l} \log_2 \tilde{P}(w_1^l). \tag{2.7}$$

This says that the estimate of $H(P, \tilde{P})$ can be obtained from a knowledge of $\tilde{P}$ on a sufficiently long word sequence. We can again only approximate this formula on the sample of text as large as possible

$$H(P, \tilde{P}) \approx -\frac{1}{l} \log_2 \tilde{P}(w_1^l). \tag{2.8}$$

The cross-entropy $H(P, \tilde{P})$ is an upper bound estimation of the entropy of the source $H(P)$ that produces the data:

$$H(P) \leq H(P, \tilde{P}). \tag{2.9}$$

It means that it is not possible to create a better language model than the original source model. The lower the cross entropy our language model has, the better it approximates the given language.

Let

$$PP = 2^{H(P)} \approx 2^{H(P,\tilde{P})} \tag{2.10}$$

denote the perplexity measure which is essentially 2 powered to the entropy. Similarly to the entropy, in language modeling we try to reduce the perplexity as much as possible. The perplexity expresses the average number of words (uniformly distributed) that can follow after the current word history. Again, the fewer words may follow, the better predictability our language model has.

2.3 N-gram language models

The probability of each word w_i in formula 2.4 is conditioned by a complete history of words w_1^{i-1}. However, the problem is still the same. There is no way how to process all possible histories of words with all possible lengths l. The number of training parameters needed to be estimated rises exponentially with the length of the history.

Thus, the word history is truncated to decrease the number of training parameters. The probability of word w_i is estimated only from $n-1$ preceding words (not from a complete history)

$$P(S) = P(w_1^l) \approx \prod_{i=1}^{l} \tilde{P}(w_i | w_{i-n+1}^{i-1}). \tag{2.11}$$

These models are referred to as the *n-gram language models*. N-gram language models have been the most often used architecture for language modeling for a long time. N-grams, where $n = 1$, are called *unigrams*. However, the more often used ones are called *bigrams* ($n = 2$) and *trigrams* ($n = 3$).

Note that even for the trigram model of a natural language, the number of training parameters is still enormous. For example, in the case of 65,536 word vocabulary, there is 2.8×10^{14} potential parameters to train. There will never be enough data for training all these parameters. Moreover, even if it could ever be, the storage space for the parameters and the probability estimate retrieval time would probably not be satisfactory. The lack of training data is sometimes referred to as the *data sparsity problem*.

Now, let us try to estimate the probability of a word w_i conditioned by the fact that the history of $n-1$ words corresponds to w_{i-n+1}^{i-1}. Let the $cnt(w_{i-n+1}^{i})$ denote the frequency of n-gram w_{i-n+1}^{i} in training data. Then the *maximum likelihood estimation (MLE)* is defined as

$$P^{MLE}(w_i | w_{i-n+1}^{i-1}) = \frac{cnt(w_{i-n+1}^{i})}{\sum_{w_i \in W} cnt(w_{i-n+1}^{i})} = \frac{cnt(w_{i-n+1}^{i})}{cnt(w_{i-n+1}^{i-1})}, \tag{2.12}$$

i.e. the number of times where the word w_i occurred after the history w_{i-n+1}^{i-1} over the frequency of this history.

At the first look, it seems that this method leads to very good probability estimations but there is a problem with data sparsity. The probability estimate of word w_i that has never been seen after the history w_{i-n+1}^{i-1} in the training data, is 0. Even in the cases where the words can follow in this order, the MLE method still gives the zero probability estimates. This problem is sometimes called the *zero problem* and it is solved by *smoothing* the probabilities.

2.4 Smoothing

The goal of smoothing is to spread out part of the probability mass of seen events (n-grams) to unseen events and eliminate the zero problem. Detailed overview of the smoothing techniques is presented in Chen and Goodman (1998). In the following sections, we describe the few of them that are important for this thesis.

2.4.1 Linear interpolation

The *linear interpolation* is a simple but effective technique for combining different language models. It is defined as follows

$$P^{LI}(w_i|w_{i-n+1}^{i-1}) = \sum_{k=1}^{K} \lambda_k \cdot P_k(w_i|w_{i-n+1}^{i-1}), \tag{2.13}$$

where λ_k is the weight of the k-th language model $P_k()$. To make the probability distribution sum up to 1, it is required that

$$\sum_{k=1}^{K} \lambda_k = 1. \tag{2.14}$$

The linear interpolation can be used as a smoothing method if we interpolate the maximum likelihood n-gram language models with different n. This is sometimes referred to as the *deleted interpolation smoothing*.

The linear interpolation can be extended to a method called *bucketed linear interpolation*, where weights become the function of the frequency of word history (Bahl et al., 1983). The main idea is that the weights λ_k should be different for words with histories with varying frequencies. The formula of linear interpolation is transformed into

$$P^{BLI}(w_i|w_{i-n+1}^{i-1}) = \sum_{k=1}^{K} \lambda_k(w_{i-n+1}^{i-1}) \cdot P_k(w_i|w_{i-n+1}^{i-1}). \tag{2.15}$$

The weights λ_k certainly cannot be different for each possible frequency of history because of data sparsity. Instead, the whole frequency spectrum is divided into buckets, where each bucket holds some range of frequencies. Histories with frequencies in the same bucket receive the same weight. The number of buckets can be tuned but it generally depends on the amount of training data available. The more training data is available, the more buckets can be used.

The optimal weights of linear interpolation can be estimated using the *expectation-maximization (EM)* algorithm (Dempster et al., 1977). The EM algorithm is an iterative process for finding maximum likelihood estimations of parameters in statistical models (this is equivalent to minimizing the entropy). The weights of linear

interpolation are tuned on the *held-out data* (the data different from the training and the testing data) to prevent overfitting.

Let ε denote the stopping criterion. The iterative process for maximum likelihood estimation of linear interpolation weights goes as follows:

1. The initial estimation of weights λ_k^0 is performed. Weights must sum up to 1 (equation 2.14) and must be nonzero. The EM algorithm guarantees the convergence, thus the initial estimation could only speed up the process. For simplicity, $\lambda_k^0 = \frac{1}{K}$, for $1 \leq k \leq K$.

2. *E-step*: The expected values $\hat{\lambda}_j^{t+1}$ of weights are calculated by

$$\hat{\lambda}_j^{t+1} = \sum_{i=0}^{L} \frac{\lambda_j^t \cdot \tilde{P}_j(w_i|w_{i-n+1}^{i-1})}{P^{LI}(w_i|w_{i-n+1}^{i-1})} = \sum_{i=0}^{L} \frac{\lambda_j^t \cdot \tilde{P}_j(w_i|w_{i-n+1}^{i-1})}{\sum_{k=1}^{K} \lambda_k^t \cdot \tilde{P}_k(w_i|w_{i-n+1}^{i-1})},$$

   where L is the size of the held-out data and t is the iteration number.

3. *M-step*: λ_j^{t+1} are obtained by normalization

$$\lambda_j^{t+1} = \frac{\hat{\lambda}_j^{t+1}}{\sum_{k=1}^{K} \hat{\lambda}_k^{t+1}}.$$

4. If the condition

$$|\lambda_j^{t+1} - \lambda_j^t| < \varepsilon$$

   is satisfied then the iterative process ends. Otherwise, t is increased by one and the process continues by step number 2.

The estimation of the weights of bucketed linear interpolation can be done in the same way but the weights are calculated for each bucket separately.

2.4.2 Modified Kneser-Ney interpolation

Modified Kneser-Ney interpolation (introduced in Chen and Goodman (1998)) is currently the state-of-the-art approach to smoothing methods in language modeling. The formula for smoothing of word probabilities is

$$P^{MKN}(w_i|w_{i-n+1}^{i-1}) = \frac{cnt(w_{i-n+1}^{i}) - D(cnt(w_{i-n+1}^{i}))}{\sum_{w_i} cnt(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1}) P^{MKN}(w_i|w_{i-n+2}^{i-1}), \tag{2.16}$$

where P^{MKN} is the probability given by the modified Kneser-Ney interpolation model. The objective of the discounting function $D(cnt)$ is to save some probability mass for lower-order models. The normalization function $\gamma(w_{i-n+1}^{i}) \in (0,1)$ makes the probability distribution sum up to 1. The definitions and derivations of these functions can be found in the original paper (Chen and Goodman, 1998).

The main advantage of the modified Kneser-Ney smoothing is the clever way in which it calculates the unigram probability distribution:

$$P^{MKN}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}. \tag{2.17}$$

The symbol $\bullet$ means an arbitrary word and $N_{1+}(w_{i-n+1}^{i})$ is the number of n-grams with frequency 1 and more (i.e. the number of such n-grams where $cnt(w_{i-n+1}^{i}) \geq 1$). In different words, the unigram probability of w_i is given by the number of different bigrams ending in w_i divided by the total number of different bigrams.

The interpolation scheme (equation 2.16) in the modified Kneser-Ney smoothing ends the model recursion by taking the 0th order distribution to be the uniform distribution over words ($\tilde{P}(w_i) = \frac{1}{|W|}$). This allows to smooth the previously unseen word forms (out-of-vocabulary – OOV words) too, where the unigram model (equation 2.17) as well as higher order models produce zero probabilities.

After each history, discounting function $D(cnt)$ saves an appropriate part of the probability mass for lower order models. It is repeated until the uniform model is used. Thanks to the recursive model, the OOV words receive different probability estimates according to the history. In some context, the OOV word is more likely to occur than in others.

2.5 Class-based n-gram models

Class-based modeling is one of the most popular techniques used for reducing the huge vocabulary-related sparseness of statistical language models (Brown et al., 1992). Individual words are clustered into a much smaller number of classes. As a result, less data are required to train a robust class-based language model. Both manual and automatic word-clustering techniques are used. Standalone class-based models usually perform poorly (see for example work of Whittaker and Woodland (2003)) which is the reason why they are usually combined with other models (for example by linear interpolation – Section 2.4.1).

Let C denote a class vocabulary. Then we can define a mapping function $m : W \to C$ which maps every word $w_i \in W$ to some class $c_i \in C$.

The probability estimation of word w_i conditioned by its history w_{i-n+1}^{i-1} is given by the following formula:

$$\tilde{P}(w_i|w_{i-n+1}^{i-1}) = \tilde{P}(w_i|c_i) \cdot \tilde{P}(c_i|c_{i-n+1}^{i-1}). \tag{2.18}$$

The MLE of the word occurrence given by its class is calculated as

$$\tilde{P}(w_i|c_i) = \frac{cnt(w_i, c_i)}{cnt(c_i)}, \tag{2.19}$$

where $cnt(w_i, c_i)$ is the number of times the word w_i is mapped to the class c_i over the frequency of class c_i. Again, in real applications the raw MLE cannot be used, and probabilities $P(w_i|c_i)$ must be smoothed. In this case, some simple smoothing technique is satisfactory (for example *Good-Touring estimation* (Good, 1953)).

Chapter 3

Distributional semantics

"Words differently arranged have a different meaning, and meanings differently arranged have different effects."

— Blaise Pascal

The research area resulting from the so-called *distributional hypothesis* (see Section 3.1.1) became very popular in last two decades mainly for empirical reasons because it suggests a simple and practical method to derive word meaning representations from a large scale data. This research area is sometimes referred to as the *distributional semantics*.

3.1 Hypotheses

3.1.1 Distributional hypothesis

Famous quotes:

- Harris (1954): *"If we consider words or morphemes A and B to be more different in meaning than A and C, then we will often find that the distributions of A and B are more different than the distributions of A and C."*
- Firth (1957): *"You shall know a word by the company it keeps."*

This principle is known as the *distributional hypothesis* and suggests that we can induce (to some degree) the meaning of words from texts. The claim has multiple theoretical roots in psychology, structural linguistics, or lexicography (Harris, 1954; Firth, 1957; Rubenstein and Goodenough, 1965; Miller and Charles, 1991; Charles, 2000). The direct implication is that two words are expected to be semantically similar if they occur in similar contexts (they are similarly distributed in the text).

3.1.2 Bag-of-word hypothesis

In this context, the term *bag* means a *set* where the order has no role, however, the duplicates are allowed (the bags a, a, a, b, b, c and c, a, b, a, b, a are equivalent).

The early reference can be found in Harris (1954), but the first practical application was arguably in information retrieval. In work of Salton et al. (1975), the documents were represented as bags-of-words and the frequencies of words in a document indicated the relevance of the document to a query. The implication is that two documents tend to be similar if they have similar distribution of similar words, no matter what is their order. This is supported by the intuition that the topic of a document will probabilistically influence the author's choice of words when writing the document.

Similarly, the words can be found related in meaning if they occur in similar documents (where document represents the word context). Thus, both hypotheses (bag-of-words hypothesis and distributional hypothesis) are related.

This intuition later expanded into many often used models for meaning extraction, such as latent semantic analysis (LSA) (Deerwester et al., 1990), probabilistic latent semantic analysis (PLSA) (Hofmann, 1999), latent Dirichlet allocation (LDA) (Blei et al., 2003), and others.

3.2 Approaches

The models based upon distributional hypothesis are often referred to as the *distributional semantics models*.

These models typically represent the word meaning as a vector which reflects the contextual information of a word across the training corpus. Each word $w \in W$ is associated with a vector of real numbers $\boldsymbol{w} \in \mathbb{R}^k$. Represented geometrically, the meaning is a point in a k-dimensional space. The words that are closely related in meaning tend to be closer in the space. This architecture is sometimes referred to as the *vector space model* (VSM) or *semantic space* (SS).

In last years, the extraction of meaning from a text became the backbone research area in natural language processing. It led to impressive results. For example, the semantic space by Rapp (2003) achieved score of 92.5% on multiple-choice synonym questions from the *Test of English as a Foreign Language* (TOEFL), while the average score given by the human test takers was 64.5%.

Several authors have made a huge effort to give an overview of the current state of the art in computational methods for extracting meaning from text (Turney and Pantel, 2010; Riordan and Jones, 2011; McNamara, 2011).

3.2.1 Context types

Different types of context induce different kinds of semantic space models. Riordan and Jones (2011) and McNamara (2011) distinguish *context-word* and *context-region* approaches to the meaning extraction. In this thesis we use the notion *local context* and *global context*, respectively, because we think this notion describes the principle of the meaning extraction better.

- **Global context:** The models that use the global context are usually based upon bag-of-word hypothesis, assuming that the words are semantically similar if they occur in similar documents, and that the word order has no meaning. The document can be a sentence, a paragraph, or an entire text. These models are able to register long-range dependencies among words. For example, if the document is about *hockey*, it is likely to contain words like *hockey-stick* or *skates*, and these words are found to be related in meaning.

- **Local context:** The models that collect short contexts around the word using moving window to model its semantics. These methods do not require text that is naturally divided into documents or pieces of text. Thanks to the short context, these models can take the word order into account, thus they usually model semantic as well as syntactic relations among words. In contrast to the global semantics models, these models are able to find mutually substitutable words in the given context. Given the sentence *The dog is an animal*, the word *dog* can be for example replaced by *cat*.

Examples of such models can be found in Table 3.1.

3.2.2 Model architectures

There are several architectures that have been successfully used to extract meaning from raw text. In our opinion, the following four architectures are the most important:

- **Co-occurrence matrix:** The frequencies of co-occurring words (often taken as an argument of some weighting function, e.g. *term frequency – inverse document frequency* (TF-IDF), mutual information, etc.) are recorded into a matrix. The dimension of such matrix is sometimes found to be problematic (it is proportional to the number of contexts in the text), and thus the *singular value decomposition* (SVD) or different algorithm can be used for dimensionality reduction.

- **Topic model:** The group of methods based upon the bag-of-word hypothesis that try to discover latent (hidden) topics in the text are called *topic models*. They usually represent the meaning of the text as a vector of topics but it is

also possible to use them for representing the meaning of a word. The number of topics in the text is usually set in advance.

- **Random indexing:** Such models usually start by creating random high-dimensional sparse vectors that are unlikely to overlap and are assumed to be nearly orthogonal. Co-occurrence of these vectors are then recorded into the final matrix representing the meanings. These method take advantage from setting the dimension at the beginning.

- **Neural network:** In the last years, these models have become very popular. It is the human brain that defines semantics, so it is natural to use a neural network for the meaning extraction. The principles of the meaning extraction differ with the architecture of a neural network.

Examples of such models can be found in Table 3.1.

Table 3.1: List of distributional semantics models with various architectures.

model	**context type**	**architecture**
VSM (Salton et al., 1975)	global	co-occurrence matrix
LSA (Deerwester et al., 1990)	global	co-occurrence matrix
HAL (Lund and Burgess, 1996)	local	co-occurrence matrix
COALS (Rohde et al., 2004)	local	co-occurrence matrix
PLSA (Hofmann, 1999)	global	topic model
LDA (Blei et al., 2003)	global	topic model
PAM (Li and McCallum, 2006)	global	topic model
CTM (Blei and Lafferty, 2006)	global	topic model
RI (Sahlgren, 2005)	local	random indexing
BEAGLE (Jones and Mewhort, 2007)	local	random indexing
RRI (Cohen et al., 2010)	local	random indexing
(Huang et al., 2012)	local+global	neural network
CBOW (Mikolov et al., 2013)	local	neural network
Skip-gram (Mikolov et al., 2013)	local	neural network

3.2.3 Vector similarity measures

The vector representation allows us to measure similarity or distance (dissimilarity) between meanings. There are many methods to compare two vectors in a multi-dimensional vector space. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^k$ denote the two vectors we want to compare and let the subscript i, where $1 \leq i \leq k$, denote the position in the vector.

Probably the simplest vector distance metrics come from the Minkowski family of distance metrics. *Euclidean* ($r = 2$) and *city-block* ($r = 1$) distances are defined as

$$S_{mink}(\boldsymbol{a}, \boldsymbol{b}) = \sqrt[r]{\sum |a_i - b_i|^r}. \tag{3.1}$$

Another metric that is often used, and is probably the most sensible from a geometrical point of view, is called *cosine similarity*. It characterizes the similarity between two vectors as the cosine of the angle between them:

$$S_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2 \sum b_i^2}}. \tag{3.2}$$

In statistics, the *Pearson product-moment correlation coefficient* (PPMCC) is a measure of correlation (linear dependence) between two variables, X and Y, giving a value between +1 and –1 inclusive. Pearson's correlation coefficient between two variables is defined as the covariance of the variables divided by the product of their standard deviations

$$S_{corr}(\boldsymbol{a}, \boldsymbol{b}) = \frac{E[(\boldsymbol{a} - \mu_a)(\boldsymbol{b} - \mu_b)]}{\sigma_a \sigma_b} = \frac{\sum (a_i - \mu_a)(b_i - \mu_b)}{\sqrt{\sum (a_i - \mu_a)^2 \sum (b_i - \mu_b)^2}}, \tag{3.3}$$

where μ_a is the mean value of the vector $\boldsymbol{a}$ and σ_a is the standard deviation of the vector $\boldsymbol{a}$ (analogically for $\boldsymbol{b}$).

Chapter 4

Presentation of the thesis publications

"I love fools' experiments. I am always making them."

— Charles Darwin

This section gives an overview of the published articles I wrote or took part in writing. Full texts of these publications are listed in Appendix A.

Our articles study three different unsupervised ways to extract hidden information from large unlabeled corpora and how to use them to improve the performance of language models.

The first article (Brychcín and Konopík, 2015a) (see Section 4.1) introduces a new unsupervised stemmer and incorporates the morphological information into language modeling. Our stemmer provides an exceptional performance in the modeling of inflectional languages.

The second article (Brychcín and Konopík, 2014) (see Section 4.2) investigates the distributional semantics models based upon the local context and their application into the language modeling task. A massive improvement is achieved especially for inflectional languages.

The third article (Brychcín and Konopík, 2015b) (see Section 4.3) uses the research from the previous two articles. Moreover, it presents an additional source of information, the global semantics model. All three sources of information are combined and proved to enrich each other. This combination dramatically improves the language modeling. These improvements were verified in the context of a real-world application (a machine translation task).

4.1 HPS: High Precision Stemmer

This section gives an overview of our article entitled *"HPS: High Precision Stemmer"* that has been published in the *Information processing & management* journal in 2015.

4.1.1 Motivation

A large class of languages (that are called *synthetic* languages in the linguistic typology) 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. Word stemming, as one of the basic preprocessing techniques in NLP, strips off these affixes. Many applications thus benefit from applying a stemming method because it leads to reduction of the vocabulary size.

The goal of stemming is related to lemmatization. In NLP, both of these methods are often used for similar purposes: to reduce the number of word forms in a text. The fundamental difference lies in the different kind of results. The outcome of lemmatization is a *lemma* which is a valid linguistic unit. The outcome of stemming is a *stem*. In linguistic sources, the term stem has different definitions (detailed information in our article). We define the stem as a common part of a word that carries the same meaning for all its lexical variants. Our definition of meaning is task dependent (e.g., for information retrieval it is the part of the word that defines what a user looks for).

Many authors have already proved that the knowledge about the morphology of a language can be very useful for language modeling (Oikonomidis and Digalakis, 2003; Kirchhoff et al., 2006; Arisoy et al., 2006; Oparin, 2008; Brychcín and Konopík, 2011). The smaller vocabulary helps language models to deal with the data sparsity problem better (to smooth the probabilities better).

Unfortunately, lemmatization requires supervised training. The labeled training data are usually not available for poorly-resourced languages. Thus, it is natural to study unsupervised methods for the same purposes. The current unsupervised stemming methods do not perform well in the language modeling task (as supported by our experiments in the article) and we feel there is a large potential for improvements.

4.1.2 Description and contributions

In our article we introduced a new unsupervised stemming algorithm called *high precision stemmer* (HPS). HPS uses two phase stemming principle to improve its performance. HPS assumes that the morphological variants of the same word are similar in meaning and thus they should be similarly distributed in a corpus (distributional hypothesis). The first phase of HPS groups lexically similar words that occur in similar contexts. These groups are then used as a training data for the sec-

ond phase which is a *maximum entropy* classifier with stemming-specific features. We use *Brainy* implementation of the maximum entropy classifier (Konkol, 2014).

HPS has several useful properties over the other competing unsupervised stemmers. It requires only a very small amount of training data to produce satisfactory results. The second stage of HPS (maximum entropy classifier) generalizes patterns seen in training data and thus it is able to efficiently stem previously unseen word forms. Thanks to the two-phase stemming process and the fact that the first phase of HPS takes the semantic information about words into account, the HPS outperforms other unsupervised stemmers.

We tested our stemmer on three different tasks and on 6 different languages (English, Spanish, Czech, Slovak, Polish, and Hungarian).

The first task, the *inflectional removal experiments*, measures the ability of a stemmer to replace a lemmatizer, i.e. how well the stemmer removes the inflectional suffixes of the words (how well does a stem correspond to a lemma). This test helps us to imagine how well the stemmer can help in applications (e.g. language modeling) where lemmatizers work well. HPS achieved the best results among all competing unsupervised stemmers.

The second task, the *information retrieval (IR) experiments*, showed that HPS produces comparable results with the state-of-the-art unsupervised stemmers purely designed for IR.

The third task, the *language modeling experiments*, is the most important for this thesis as it shows how HPS improves the language models. We created three different language models. The baseline based upon the modified Kneser-Ney smoothing (Section 2.4.2), and two class-based models (Section 2.5) also smoothed by the modified Kneser-Ney smoothing. To create word groups for two class-based models we used the stems and the suffixes after stems. All three models were then interpolated by bucketed linear interpolation (Section 2.4.1). Perplexity of the inflectional languages (Czech, Slovak, Polish, and Hungarian) was improved by approximately 11% on average compared to the baseline. These results are comparable with those where lemmatizers are used (see e.g., (Brychcín and Konopík, 2011)).

4.2 Semantic Spaces for Improving Language Modeling

This section gives an overview of our article entitled *"Semantic Spaces for Improving Language Modeling"* that has been published in the *Computer speech & language* journal in 2014.

4.2.1 Motivation

As it was written in Section 2, the language modeling deals with the data sparsity problem. The number of possible word sequences grows exponentially with respect to the size of the vocabulary. N-gram language models partially solve this problem but

the number of possible n-grams is still enormous. This problem is especially evident in languages with a rich morphology, e.g. Czech. The word sequences that have never been seen in the training data receive low probability even if they are made of words that are semantically similar to words forming the already seen sentences.

Incorporating the information about word similarity into language models should distinctly help to deal with the data sparsity. Generally, it should allow each training sentence to inform the model about an exponential number of semantically neighbouring sentences.

4.2.2 Description and contributions

In Section 3, we introduced the research area resulting from distributional hypothesis that studies meaning extraction techniques from large unannotated corpora. In our article we study some of these methods working with local context. We experimented with HAL (Lund and Burgess, 1996), COALS (Rohde et al., 2004), RI (Sahlgren, 2005), BEAGLE (Jones and Mewhort, 2007), and P&P (Purandare and Pedersen, 2004). To our best knowledge, these semantic spaces have never been used in language modeling before. This is the main contribution of our paper.

We created word clusters from each semantic space. These clusters represent the words that are assumed to be mutually substitutable in the given context (clusters are derived by grouping the similarly distributed words in the corpus). The words were clustered into five different depths, i.e. 1,000, 5,000, 10,000, 20,000, and 50,000 clusters.

We choose the class-based language model (Section 2.5) as an appropriate architecture for this kind of information. We took class-based models for each five clustering depths (from 1,000 up to 50,000 clusters) and interpolated them with the baseline language model by the bucketed linear interpolation (see Section 2.4.1). Each stand-alone language model used the modified Kneser-Ney smoothing (Section 2.4.2). Thus, the final interpolated model was a combination of six language models.

Our language models were tested on Czech, Slovak, and English corpora. During our experiments, we found that semantic spaces behave differently when we cluster words into different number of clusters. Some of the semantic spaces (e.g. HAL model) are better for dense clusters (1,000 or 5,000 classes), some of them (e.g. COALS model) are more suitable for sparse clusters (20,000 or 50,000 classes). The combination of the HAL-based language models for small numbers of clusters together with the COALS-based models for greater numbers of clusters gives the best results.

We achieved significant improvements in the prediction ability of the language models. The perplexity was improved by 17.9%, 16.1%, and 10.1% for Czech, Slovak, and English, respectively. We also tested our language models on the machine translation task to prove their ability to improve a real-word application. The quality of translations were enhanced by 0.72, 0.66, and 0.37 of BLEU points for Czech, Slovak, and English, respectively.

4.3 Latent semantics in language models

This section gives an overview of our article entitled *"Latent semantics in language models"* that has been published in the *Computer speech & language* journal in 2015.

4.3.1 Motivation

In previous two sections we introduced two unsupervised approaches to improve language modeling, i.e. the unsupervised stemming and the local context semantic spaces. Both approaches are based upon distributional hypothesis and produce promising results.

In Section 3, we mentioned a group of methods called topic models that study long-range dependencies between words across the documents. The motivation for using such models in language modeling is that documents may differ in domains, topics, and writing styles. This also means that they have different probability distributions of n-grams. This assumption can be used for adapting language models to the long context (domain, topic, style of particular documents). This long-context information can be added to standard n-gram models to adapt them to the global context. Some researches already experimented with using such information in language modeling (Gildea and Hofmann, 1999; Wang et al., 2003; Tam and Schultz, 2006; Watanabe et al., 2011).

All three unsupervised approaches (local semantics, global semantics, and morphology) separately were proved to significantly improve language modeling. However, to the best of our knowledge, no one studied the combination of all three approaches in language modeling at once. Since all three sources of information are used for different purposes, we investigate whether they enrich each other and whether their combination achieves better results.

4.3.2 Description and contributions

In this article we extend our work on the application of semantic spaces in language modeling (Brychcín and Konopík, 2014), where we have achieved significant improvements in perplexity and in machine translation task especially with HAL, COALS and RI models. Thus, these models are investigated more deeply in this article.

We attempt to improve language modeling by adding long-range semantic dependencies. We choose latent Dirichlet allocation (LDA) for that task because it has already been shown by many researches that LDA improves language modeling significantly (Tam and Schultz, 2006; Watanabe et al., 2011).

The performance of these language models is further enhanced by our unsupervised stemming algorithm called high precision stemmer (HPS) introduced in Brychcín and Konopík (2015a). We have already tested our stemmer in language modeling tasks

and the results indicate that HPS performs best compared to other unsupervised stemmers.

To the best of our knowledge, we are the first to try to combine these three sources of information (i.e. local semantics, global semantics, and morphology).

LDA is used as a unigram language model that estimates the probability of a word according to the topic distribution in an actual document (global context). HAL was incorporated in the same way as in the previous article, i.e. as class-based language models (Section 2.5) derived from word clusters. Finally, HPS also uses the class-based language model architecture. Each stand-alone language model is smoothed by the modified Kneser-Ney interpolation (Section 2.4.2). To combine them all, we use the bucketed linear interpolation (Section 2.4.1) again.

We carried out experiments on six languages within three different language families: Czech, Slovenian, Slovak, Polish, Hungarian, and English. Our language models were able to dramatically reduce the perplexities of the language models. By grouping HPS, HAL, and LDA with a baseline we achieved approximately 25% improvement on average for all inflectional languages and 15% for English, which is a very satisfactory result. The same models were also investigated in a machine translation task where we measured BLEU scores. Significant growth of the BLEU score was achieved only by changing the language model (the average improvement among all languages was 0.8 BLEU points).

The most important finding is that the investigated sources of information mutually help each other in terms of improving language modeling.

Chapter 5

Summary

"There are only two things in the world:
nothing and semantics."

— Werner Erhard

5.1 Contributions

The contributions of the thesis are the following:

- We developed a novel unsupervised stemmer called HPS which outperforms other competing stemmers on several languages and tasks.
- We were the first to improve the language modeling by the local context semantic spaces (e.g. HAL, COALS, RI).
- We combined three different sources of information (HPS, semantics spaces, and LDA) and empirically proved they enrich each other in terms of improving language modeling.
- We studied the language modeling task of different languages within different language families. We were focused on inflectional languages that are difficult to model and that have not gained as much attention as for example English.
- We found that the methods presented in this thesis are very useful also in other disciplines of NLP. Our stemmer has been successfully used to improve document classification (Brychcín and Král, 2014), named entity recognition (Konkol and Konopík, 2014; Konkol et al., 2015), or sentiment analysis (Steinberger et al., 2014; Habernal et al., 2013,0; Ptáček et al., 2014). Similarly, the semantic models we used in this work to improve language modeling, can be

also used in document classification (Brychcín and Král, 2014), sentiment analysis (Brychcín et al., 2014; Habernal and Brychcín, 2013), and named entity recognition (Konkol et al., 2015).

5.2 Fulfillment of the thesis goals

The thesis publications meet all the thesis goals. The summary for each goal is given in the list below:

- **Deal with specific properties of Czech language and other inflectional languages. Study the relationship between morphology and language modeling.**

  Czech is a representative of inflectional languages that are characterized by a rich morphology. This property complicates the language modeling task because it leads to a huge number of possible word sequences.

  The absence of enough efficient unsupervised methods for word normalization led us to development of the new multilingual stemming algorithm called HPS (Section 4.1). HPS outperforms all other competing unsupervised stemmers and significantly improves the language modeling of several inflectional languages (including Czech).

- **Use semantic information to improve language modeling. Unsupervised training is preferred.**

  A semantic information and the ways how to extract it is the most important part of this thesis. All the methods we studied in this thesis are based upon the unsupervised training.

  In the first article (Section 4.1) we developed new word stemming algorithm (HPS) that uses semantic information in the stemming of words. HPS is proved to be very useful in the language modeling.

  The second article (Section 4.2) studies the local context semantic information used for word clustering. The class-based language models are derived and significantly improve the language modeling.

  The third article (Section 4.3) extends both previous articles and adds a global semantics model. All three sources of information (morphology, local context semantics, and global context semantics) are combined and proved to enrich each other. This combination dramatically improves the language modeling.

- **Focus on the analysis of out-of-vocabulary words. Explore both the unsupervised morphological analysis and the analysis of the context.**

  The OOV words complicate the language modeling task because it is hard to represent them. The only information we have about the OOV word is this word itself and its direct context. In this thesis, we explore both options.

Our HPS (Section 4.1) exploits the morphological information about the OOV word because there is a chance that this OOV word is only a different morphological variant of a previously seen word (they share the same stem).

All our n-gram language models naturally estimate the probabilities of the OOV words given a short history of words (context) because they use the modified Kneser-Ney smoothing. However, our contribution is in combining different sources of information (morphology, local context semantics, and global context semantics; see Section 4.3). We use the class-based model architecture for incorporating knowledge about the morphology and the semantics of words. The history of classes helps to represent the context better. Thus, the OOV word probability can be more precisely estimated.

Moreover, global semantics model predicts the occurrence of an OOV word according to document topics distribution. In some topics, the OOV words are more likely to occur then in others.

5.3 Future work

As shown in Chapter 3, there is an enormous number of methods for latent semantics discovery that are worth investigating. These methods use various architectures, e.g. co-occurrence matrices, graphical models, or neural networks. It is beyond the scope of this thesis to compare them all. This is the main direction for the future work as we believe that other combinations may produce even better language models.

In this thesis, we test our language models in a machine translation task to prove they are able to improve a real-world application. In future, we want to test our language models in different NLP tasks (e.g. speech recognition, optical character recognition, spelling correction, etc.).

Finally, in this thesis we use the class-based models architecture to incorporate the information about the word semantics and the word morphology. These stand-alone models were then combined via bucketed linear interpolation. In future, we would like to investigate new architectures (e.g. neural networks).

5.4 Conclusion

In this thesis we experimented with three kinds of various information sources. Their application into language modeling yields significant improvements in model prediction ability. The beauty of our method is that the whole information comes directly from the data. Nothing is added externally. The information we used is sometimes called *latent* or *hidden* because an algorithm must be employed to discover it. We used three approaches for the discovery of the hidden information. Topic models discover long semantic relations between the words and the documents in which they are used. Semantic spaces (HAL, COALS, RI) work with short semantic relations

between words and their direct contexts. The stemming studies the information hidden directly in different forms of words.

All these sources separately were proved to improve the language modeling. Moreover, we conclusively proved that their combination enhances the prediction ability of language models even more than the individual models do.

We believe that this thesis carries a potential beyond language models. All the methods investigated in this thesis are based upon the unsupervised training. Thus, they can be easily applied to different NLP tasks.

Appendix A

Thesis publications

"Either I will find a way, or I will make one."

— Philip Sidney

This appendix contains the full text of the thesis publications in the form they were accepted for the publication.

A.1 HPS: High Precision Stemmer

Journal title:	Information Processing and Management
Journal ISSN:	0306-4573
Article DOI:	10.1016/j.ipm.2014.08.006
Received at Editorial Office:	1 November, 2013
Article accepted for publication:	27 August, 2014
Volume/issue:	51/1
Pages:	68–91

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 Trans-

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

lation) 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 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 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

[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.

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 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

[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.

[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`.

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].

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.

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

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

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

the frequencies of words. An 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.

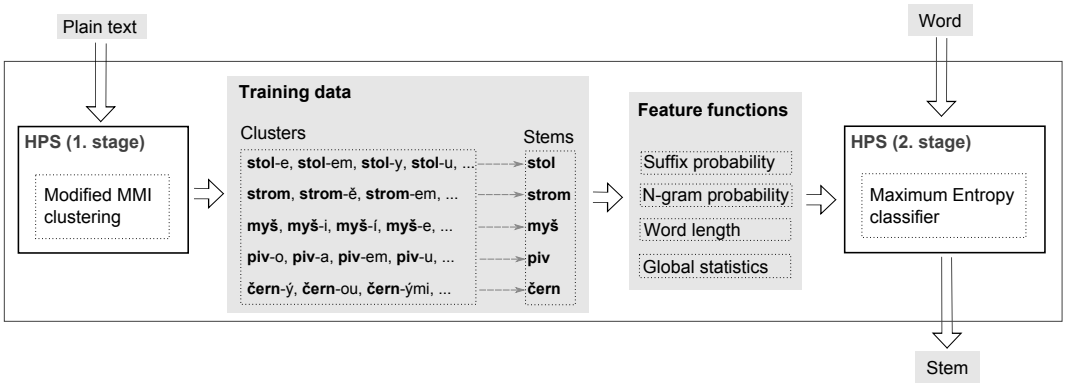


Figure 1: HPS architecture.

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 $I(C^L; C^R)$

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

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 $m_i : W \rightarrow C_i$ 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:

$$m_i = \underset{c_a, c_b}{\operatorname{argmin}} \left[I\left(C_{i-1}^L; C_{i-1}^R\right) - I\left(C_i^L; C_i^R\right) \right], \quad c_a \neq c_b, \quad c_a, c_b \in C_{i-1}, \tag{6}$$

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

$$m_i = \underset{c_a, c_b}{\operatorname{argmax}} \frac{S\left(c_a, c_b\right)}{I\left(C_{i-1}^L; C_{i-1}^R\right) - I\left(C_i^L; C_i^R\right)}, \quad c_a \neq c_b, \quad c_a, c_b \in C_{i-1}, \tag{7}$$

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.

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.

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.

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

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

where $\tilde{E}\left(f_i\left(x, y\right)\right)$ is the expected value of a feature $f_i\left(x, y\right)$ estimated from the training data and $E\left(f_i\left(x, y\right)\right)$ 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 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:

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

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

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

[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].

[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.

4.2.2. 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).

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:

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

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:

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

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

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

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.
- **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].

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

[10] Available at http://langtech.jrc.it/.

[11] Available at http://trec.nist.gov/data/reuters/reuters.html.

- **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

[12]Available from NIST Standard Reference Data Products at http://www.nist.gov/srd/nistsd23.cfm.

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):

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

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.

[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 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.

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.

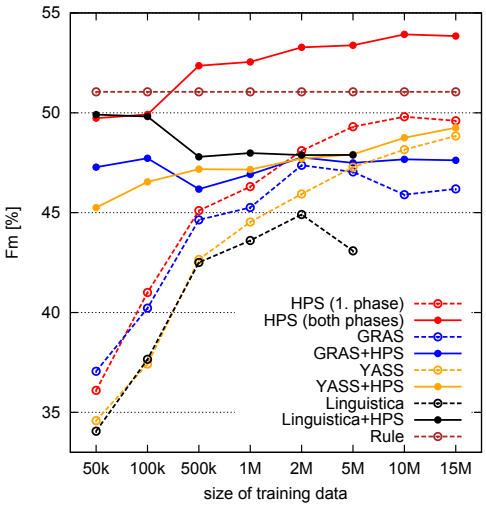
In the tables we also show the results for the stand-alone first phase of HPS to see how large an improvement the second phase of HPS (maximum entropy classifier) produces. The first phase of HPS does not lead to the best results, but has the best precision (except on English).

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.

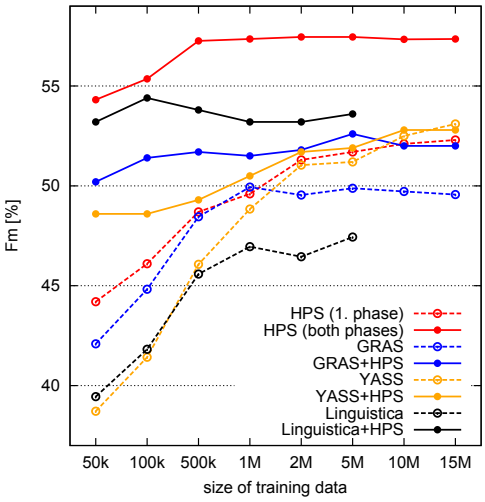
5.4.3. The impact of the size of the training dataset

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,

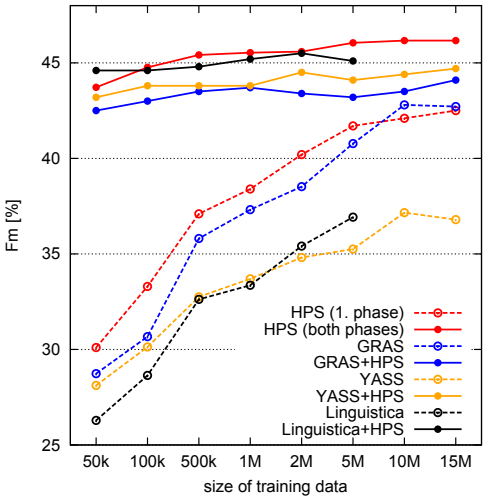
and continue up to 15,000,000 tokens, which we believe is a sufficient amount for all methods. The results are shown in Figure 2.



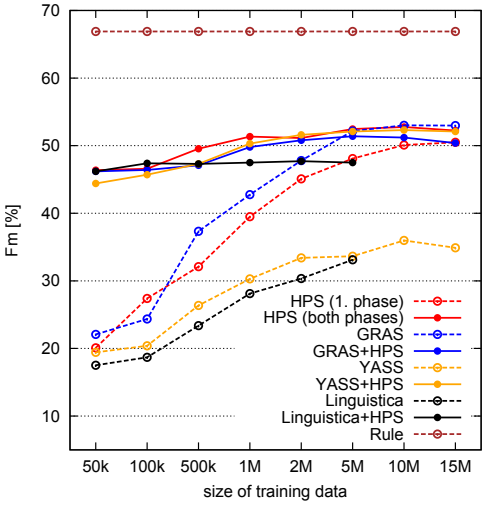
(a) Czech (CZ)



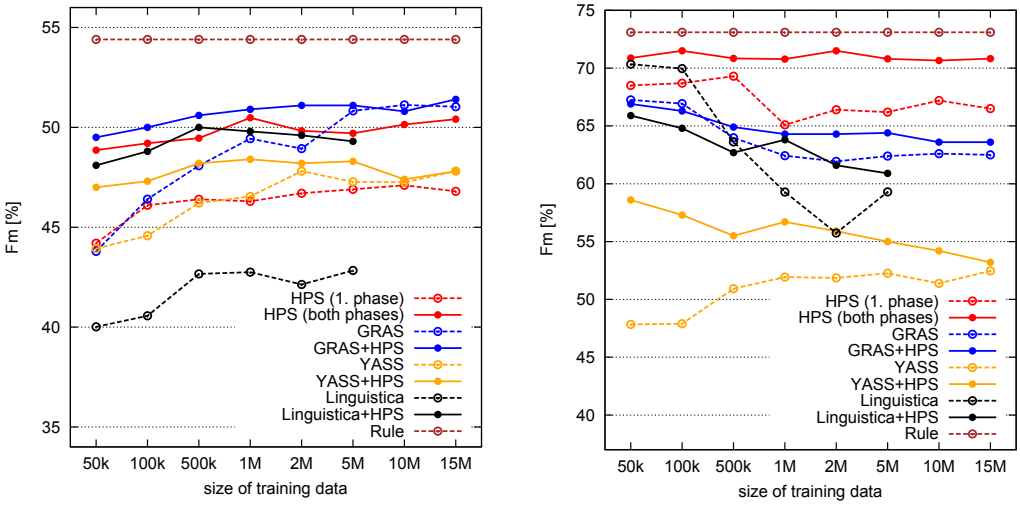
(b) Slovak (SK)



(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 compet-

itive 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.

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

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

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 *R*th 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]:

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

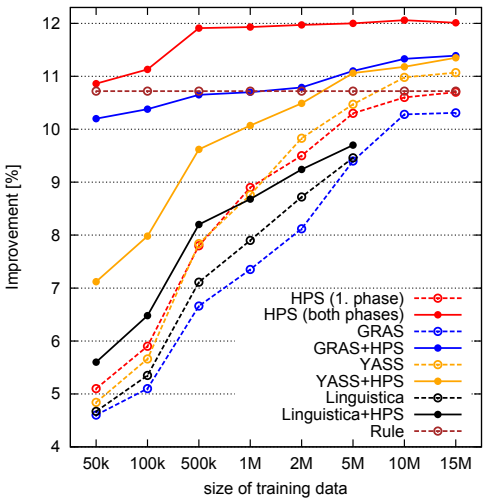
where $\lambda_k()$ is the weight of the kth language model, $P_k()$. We use the EM (Expectation Maximization) algorithm described in [Dempster et al., 1977] to calculate the optimal weights λ_k by maximizing the probability of the held-out data. In bucketed interpolation, the weights are functions of 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

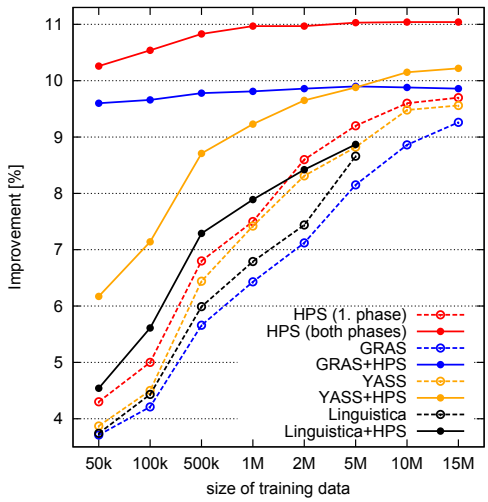
The performance of a language model is typically measured in terms of the perplexity of the model on the unseen test corpus. The perplexity can be seen as the confusion of the model. Lower perplexity means a better prediction ability of language model. It was shown by many authors that the reduction 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.

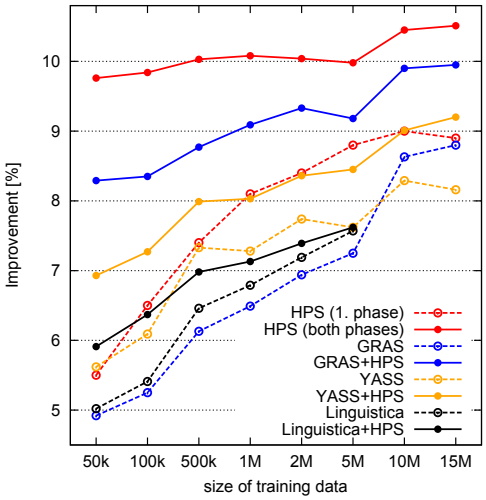
Figure 3 shows the improvement in perplexity when compared to the baseline language model.



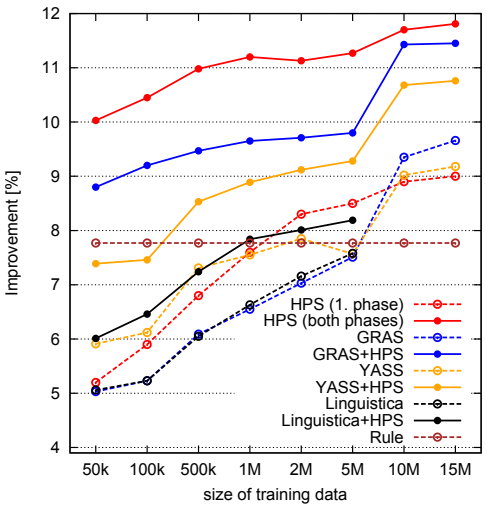
(a) Czech (CZ): baseline perplexity 472.



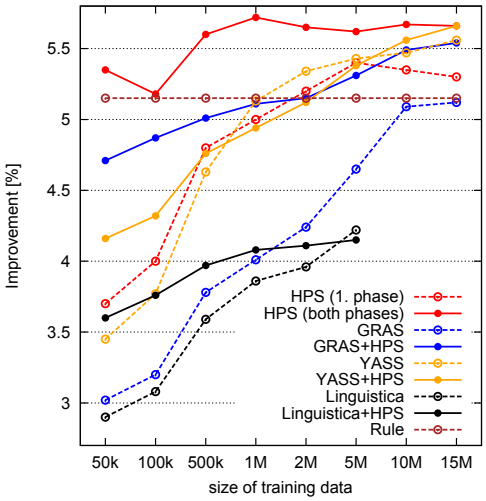
(b) Slovak (SK): baseline perplexity 527.



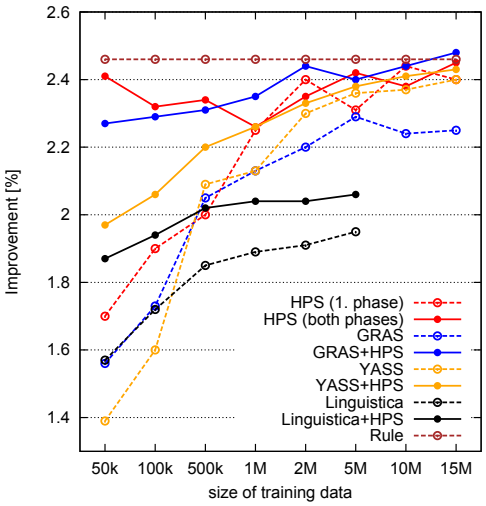
(c) Polish (PL): baseline perplexity 153.



(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.

A.2 Semantic Spaces for Improving Language Modeling

Journal title:	Computer Speech & Language
Journal ISSN:	0885-2308
Article DOI:	10.1016/j.csl.2013.05.001
Received at Editorial Office:	21 March, 2012
Article accepted for publication:	10 May, 2013
Volume/issue:	28/1
Pages:	192–209

Semantic Spaces for Improving Language Modeling[☆]

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

[a] *Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia in Pilsen, Univerzitní 8, 306 14 Plzeň, Czech Republic*
[b] *NTIS – New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia in Pilsen, Univerzitní 8, 306 14 Plzeň, Czech Republic*

Abstract

Language models are crucial for many tasks in NLP (Natural Language Processing) and n-grams are the best way to build them. Huge effort is being invested in improving n-gram language models. By introducing external information (morphology, syntax, partitioning into documents, etc.) into the models a significant improvement can be achieved. The models can however be improved with no external information and smoothing is an excellent example of such an improvement.

In this article we show another way of improving the models that also requires no external information. We examine patterns that can be found in large corpora by building semantic spaces (HAL, COALS, BEAGLE and others described in this article). These semantic spaces have never been tested in language modeling before. Our method uses semantic spaces and clustering to build classes for a class-based language model. The class-based model is then coupled with a standard n-gram model to create a very effective language model.

Our experiments show that our models reduce the perplexity and improve the accuracy of n-gram language models with no external information added. Training of our models is fully unsupervised. Our models are very effective for inflectional languages, which are particularly hard to model. We show results for five different semantic spaces with different settings and different number of classes. The perplexity tests are accompanied with machine translation tests that prove the ability of proposed models to improve performance of a real-world application.

Keywords: Class-based language models, Semantic spaces, HAL, COALS, BEAGLE, Random Indexing, Purandare&Pedersen, Clustering, Inflectional languages, Machine translation.

[☆] This document is a collaborative effort.

[*] Corresponding author

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

1. Introduction

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

Research into language modeling started more than 20 years ago and has evolved into a very mature discipline. Now it is very difficult to outperform the state of the art. Our research is focused on inflectional languages as we believe that these languages offer some room for improvement. We however also provide experiments for English (which is not a very inflectional language). Even in the case of English, we were able to obtain positive results.

Czech and Slovak belong to the Slavic language group. These languages are highly inflectional and have a relatively free word order. Czech has seven cases and three genders. Slovak has six cases and also three genders. The word order is very variable from the syntactic point of view: words in a sentence can usually be ordered in several ways, each carrying a slightly different meaning. These properties of the languages complicate the language modeling task. The great number of word forms and more possible word sequences lead to a greater number of n-grams. Data sparsity is a common problem of language models. In Czech, Slovak and other Slavic languages, this problem is more evident.

Class-based modeling is the most popular technique used for reducing the huge vocabulary-related sparseness of statistical language models [Brown et al., 1992]. Individual words are clustered into a much smaller number of classes. As a result, less data are required to train a robust class-based language model. Both manual and automatic word-clustering techniques are being used. Standalone class-based models usually perform poorly, which is the reason why they are usually combined with other models. Many researchers have demonstrated that the combination of a standalone class-based language model and a standard word n-gram model reduces the model perplexity [Maltese et al., 2001; Whittaker, 2000; Whittaker and Woodland, 2003].

An effective solution for language modeling is to use information about the morphology of the language. In [Oparin, 2008] experiments with morphological random forests in the Czech and Russian language are shown with the conclusion that they can be used effectively for inflectional languages. Authors of [Vaiciunas et al., 2004] describe the language modeling of Lithuanian by means of class-based language models derived by word clustering and morphological word decomposition and their linear interpolation with the baseline word n-gram model. The authors present a perplexity reduction of 8-13% depending on the size of the corpora. A similarly effective solution is to use class-based language models where classes are derived from lemmas and morphological categories [Brychcín and Konopík, 2011]. The article shows a perplexity reduction of 10-30% in corpora in the Czech and Slovak languages. A comparative study of several methods using morphological information for modeling conversational Arabic can be found in [Kirchhoff et al., 2006]. The usage of morphological information seems to be very effective for inflectional

languages; however, it requires a huge number of manually annotated texts.

In [Brown et al., 1992] the MMI (Maximum Mutual Information) clustering algorithm was introduced. This algorithm is based upon the principle of merging a pair of words into one class according to the minimal mutual information loss principle. The algorithm gives very satisfactory results and it is completely unsupervised. Its complexity is however very problematic. This method of word clustering is possible only in very small corpora and is not suitable for large vocabulary applications. The authors in [Yokoyama et al., 2003] used the MMI algorithm to build class-based language models. Their linear interpolation with the word n-gram model was applied to speech recognition of Japanese. The authors showed a 2% absolute improvement in word accuracy but only in very small corpora.

Several authors have tried to approximate the MMI algorithm to reduce computational requirements and to make it more suitable for large vocabulary language models [Bai et al., 1998; Yamamoto and Sagisaka, 1999]. Automatically derived clusters have been used for class-based language models of Japanese and Chinese [Gao et al., 2002]. The authors concentrated on the best way of using the clusters; however, they did not focus on how to get them.

Another way of improving language models is to use semantic information. This idea is based on the assumption that words with lexically different forms usually share similar meanings in cases where they frequently occur in similar contexts. The semantic information can be calculated using the Latent Semantic Analysis (LSA) [Deerwester et al., 1990; Landauer and Dumais, 1997; Landauer et al., 1998] method or its probabilistic variant, the PLSA [Hofmann, 1999]. A similar method to the PLSA is the Latent Dirichlet Allocation (LDA) [Blei et al., 2003] which is essentially the Bayesian version of the PLSA model. Bellegarda and his team were the first to introduce LSA into language modeling [Bellegarda et al., 1996]. Their approach consisted in using LSA to derive word clusters for class-based language models.

The approach then evolved to focus on documents instead of focusing on words. It is assumed that documents may vary in domain, topic and styles, which means that they also differ in the probability distribution of n-grams. This assumption is used for adapting language models to the long context (domain, topic, style of particular documents). LSA (or similar methods) are used to find out which documents are similar and which are not. This long context information is added to standard n-gram models to improve their performance. A very effective group of models (sometimes called topic-based language models) work with this idea for the benefit of language modeling. In [Bellegarda, 2000] a significant reduction in perplexity (down to 33%) and relative reduction in WER[1] (down to 16%) in the WSJ (Wall Street Journal) corpus was shown. Many other authors have obtained good results with PLSA [Gildea and Hofmann, 1999; Wang et al., 2003] and LDA [Tam and Schultz, 2005, 2006] approaches.

In [Liu and Liu, 2007, 2008], the named entity recognition technique was applied

[1]The Word Error Rate (WER) measure is often used in Speech recognition.

to topic modeling. The topic modes were based upon LDA and clustering. The authors tested the hypothesis that named entities carry valuable information which can be useful for latent topic analysis. The authors presented a 14% perplexity reduction as their best result.

Some comparisons between PLSA and LDA as well as some clustering methods can be found in [Hahn et al., 2008]. The authors present their results in English and Arabic broadcast news.

An investigation into Topic Tracing Language Models (TTLM) and their application in speech recognition is presented in [Watanabe et al., 2011]. The TTLM is based on LDA and PLSA and integrates the ability to dynamically track changes in topics. The tracking is based upon focused text information and previously estimated topics.

To put our approach into the context of the above-mentioned methods, we can state that our method also focuses on inflectional languages similarly to methods that use morphology. Our approach, however, is unsupervised. Our method also uses semantic information. We do not rely however on LSA, PLSA or LDA but on different methods (HAL, COALS, BEAGLE and others) that were not tested in the language modeling task before. These methods do not require the text to be partitioned into documents as in the case of LSA, PLSA or LDA. Our approach is in many ways similar to the approach of MMI clustering but the methods we use can deal with much larger data.

The rest of the article is organized as follows. In the following section, we give an overview of the statistical modeling of semantic information. In section 3 we explore the application of semantic information in language modeling. Section 4 shows various results of our experiment in detail. In the last sections we discuss the results and we conclude the article.

2. Semantic spaces

The semantic models investigated in this work are based upon the idea that the word meaning is related to the context in which the word is usually used. The assumption is that two (lexically) different words share a similar meaning if they occur in similar contexts. Some studies [Rubenstein and Goodenough, 1965; Charles, 2000] have confirmed this assumption 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 this article we use the assumption and its implication to improve language models. Before we introduce the method of incorporating semantic similarity into language modeling we briefly explain several methods for calculating word similarity.

In semantic spaces, each word is represented as a highly dimensional vector. The vectors are derived from the statistical properties of words and their contexts in a plain text corpus. The vectors are constructed in such a way that words similar in meaning should have a similar vector. The methods to calculate the vectors differ.

In the following sub-sections we briefly explain the methods that were tested in this article.

2.1. HAL

Hyperspace Analogue to Language (HAL) [Lund and Burgess, 1996; Burgess and Lund, 1997] creates a semantic space from word co-occurrences. Each word in the training data is examined and those words nearer than a fixed distance are recorded as co-occurring. Such a group of words is called a "window". The words in the window are weighted according to the distance from the examined word. It is assumed that the closer the word is, the greater the impact it has on the focused word semantics. The co-occurring words are therefore inversely weighted according to their distance from the examined word.

These windows are used to construct the $|W| \times |W|$ co-occurrence matrix $\mathbb{M}$ ($|W|$ is the number of words being analyzed) in the following way. When a word w_j is found in the window of the examined word w_i then a value is added to the $m_{i,j}$ element in the matrix $\mathbb{M}$. The value depends on the distance of the word w_j from the word w_i. The exact formula to calculate the value is defined in [Lund and Burgess, 1996]. The row and column vectors of the matrix $\mathbb{M}$ contain co-occurrence information on words that appeared before and after respectively. HAL therefore also records simple word-ordering information. Naturally, many words do not appear in the vicinity of each other and the matrix $\mathbb{M}$ tends to be very sparse.

For each word (meaning), certainly not all columns (co-occurred words) provide an equal amount of information. The entropy can be used to retain only a given number of significant columns. In this way, the dimensionality of the matrix $\mathbb{M}$ can be reduced.

2.2. COALS

Correlated Occurrence Analogue to Lexical Semantic (COALS) [Rohde et al., 2004] is a semantic space model based upon HAL and LSA ideas. The process of building the matrix starts almost identically to the HAL methods but COALS adds some tweaks.

The algorithm constructs the matrix $\mathbb{M}$ in a similar way as HAL does. It however does not distinguish whether a co-occurred word comes before or after the focused word. The window that is used has the same length in both directions. The matrix is also normalized using correlation. Any negative values are set to zero and all other values are replaced by the square root.

The final part of the algorithm is inspired by the LSA method. Singular Value Decomposition (SVD) is applied to the matrix $\mathbb{M}$ in order to reduce the dimensionality of the vector space (typically, dimension about 800 is used). The SVD reduction has the effect of bringing out the latent semantic relationships between words (as in LSA) so it can discover transitive relations between words. This final stage is not mandatory for the COALS method and is sometimes skipped.

2.3. Random Indexing

Random Indexing (RI) [Sahlgren, 2005] is based on the process of the accumulation of context vectors of words that are co-occurring. This incremental technique is used to construct the semantic space in a completely different way from the above-described models. Instead of constructing a word by word matrix and then deriving the context vectors, the process is reversed. First, vectors are generated and then the matrix is calculated.

The Random Indexing method can be described in two step. In the first step, a randomly generated high-dimensional vector is assigned to each word. The dimensionality of vectors typically reaches thousands of dimensions. The vectors consist of a small number of randomly distributed nonzero vector values (-1, +1). In this way it is ensured that two vectors do not overlap very often. The generated vector is known as the index vector. During the second step, the algorithm scans the text and updates the context vectors by summing up all the index vectors of co-occurring words.

This method does not require the dimension reduction phase as in the case of HAL and COALS. Here the dimension is set at the beginning and is much lower that in the case of HAL and COALS.

In [Sahlgren et al., 2008] the Random Indexing method is extended to keep the word-order information. The modification is inspired by the BEAGLE method (subsection 2.4), but instead of using convolution operation this method is based upon the permutation of vector coordinates. While both methods are approximative, the permutation is more simple to calculate.

2.4. BEAGLE

Bound Encoding of the AggreGate Language Environment (BEAGLE) [Jones and Mewhort, 2007] is a computational model that builds a semantic space in a similar way to Random Indexing (subsection 2.3). During the first phase, a high-dimensional index vector is also randomly generated; however, the values are given according to the Gaussian distribution. The mean value is set to 0 and the variance is set to $1/D$, where D is the dimension (by default, $D = 1024$).

The meaning of words in the final semantic space is compounded from the co-occurrence information and word order information. The co-occurrence information is calculated as in Random Indexing by summing the vectors of the co-occurring words to the vector of the focused word. The word order information is calculated by convolution of the n-gram vectors that contain the focused word. The final semantic vector is then constructed as a combination of the co-occurrence vector and the word order vector and is sensitive to both the neighboring words and word order.

2.5. Purandare and Pedersen

The Purandare and Pedersen (P&P) [Purandare and Pedersen, 2004] model is another form of word sense induction, in which word meaning is inducted from different usages in training data.

The process of building the semantic space is divided into two stages. First, the training data are processed and features most likely correlated with focused words are identified. The model uses two kinds of features: co-occurring words (similarly to HAL (subsection 2.1) or the COALS model (subsection 2.2)) and co-occurring bigrams. The features are selected from a close distance to the focused word (e.g. five-word distance). Features which are statistically significant are kept, others are removed. This leads to the removal of words which possibly frequently co-occur with the focused word but which do not have a significant impact on the semantics of the focused word.

In the second stage, the algorithm tries to construct the meaning of words from longer contexts (e.g. 20 words on both sides of the focused word). Only words that are in the features of the focused word are included in the longer context vectors, while others are removed. It is expected that these context vectors represent different usages of the focused word in the corpus. These vectors (usages) are then clustered into a predefined number of clusters. Each of the final clusters receives its own semantic vector and represent one of the meaning of the word. In the final semantic space, each word is described by n meanings. Its final semantic vector is created as a combination of clustered vectors.

2.6. Clustering

The fact that a word is characterized by a vector opens up the opportunity to easily compare two words. The more similar two words are in meaning, the more similar their vectors should be. The ability to compare two words enables us to use a clustering method. Similar words are clustered into bigger groups of words (clusters).

2.6.1. Vector similarity metrics

The distance (similarity) between two words can be calculated by a vector similarity function. Let $\vec{a}$ and $\vec{b}$ denote the two vectors to be compared and $S\left(\vec{a},\vec{b}\right)$ denote their similarity measure. Such a metric needs to be symmetric: $S\left(\vec{a},\vec{b}\right) = S\left(\vec{b},\vec{a}\right)$.

There are many methods to compare two vectors in a multi-dimensional vector space. Probably the simplest vector similarity metrics are the familiar Euclidean ($r = 2$) and city-block ($r = 1$) metrics

$$S_{mink}\left(\vec{a},\vec{b}\right) = \sqrt[r]{\sum |a_i - b_i|^r}, \tag{1}$$

that come from the Minkowski family of distance metrics.

Another often used metric characterizes the similarity between two vectors as the cosine of the angle between them. The cosine similarity is defined as follows:

$$S_{\cos}\left(\vec{a},\vec{b}\right) = \cos\left(\theta\right) = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\cdot\left\|\vec{b}\right\|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2 \sum b_i^2}}. \tag{2}$$

In statistics, the Pearson product-moment correlation coefficient (sometimes referred to as the PPMCC) is a measure of the correlation (linear dependence) between two variables, X and Y, giving a value between +1 and –1 inclusive. Pearson's correlation coefficient between two variables is defined as the covariance of the variables divided by the product of their standard deviations

$$S_{corr}\left(\vec{a},\vec{b}\right) = \frac{E\left[\left(\vec{a}-\mu_a\right)\left(\vec{b}-\mu_b\right)\right]}{\sigma_a\sigma_b} = \frac{\sum\left(a_i-\mu_a\right)\left(b_i-\mu_b\right)}{\sqrt{\sum\left(a_i-\mu_a\right)^2\sum\left(b_i-\mu_b\right)^2}}, \tag{3}$$

where μ_a is the mean value of the vector $\vec{a}$ and σ_a is the standard deviation of the vector $\vec{a}$.

This metric is often used in semantic spaces for dense matrices, while the cosine metric is used for sparse matrices.

2.6.2. Clustering algorithm

The clustering of words with similar meaning is the most important and simultaneously the most time consuming part in our class-based language models.

The goal of clustering is simple; to find an optimal grouping in a set of unlabeled data. There are, however, two problems. Firstly, the optimality criterion must be defined. This criterion depends on the task that is being solved. The second problem is the complexity of the problem. The number of possible partitioning rises exponentially[2] with the number of elements in the set. It is therefore impossible to examine every possible partitioning of even a decently large set. The task is then to find a computationally feasible algorithm that would be as close to the optimal partitioning as possible.

In our case, the optimality criterion is supplied from a semantic space and an appropriate similarity metric. The clustering in our case is complicated even more by the fact that we are dealing with really large quantities of data (in the order of hundreds of thousands).

The type of algorithm plays a key role. Hierarchical methods go from the bottom (they start with many classes and join them together) and that is problematic in our case. We need relatively few classes and so a lot of joining operations must be executed. On the contrary, the partitioning methods go from the top (they start with one class and split it repeatedly). These methods are more suitable for us since much fewer operations of splitting than joining is required.

Even with the partitioning clustering method we struggled with the computation complexity of the algorithms. It was soon apparent that a very efficient algorithm would be necessary. Our task was to experiment with approximative partitioning clustering methods. A useful guide was found in the article by [Zhao and Karypis,

[2]To be exact, the number of possible partitioning of a n-element set is given by the Bell number which is defined recursively: $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$.

2002] where a comparison of clustering methods was presented. We were able to conclude that the Repeated Bisection algorithm gave satisfactory results with acceptable computational requirements. We use the implementation of the algorithm from the CLUTO software package [Karypis, 2003].

3. Language models

3.1. Class-based n-gram Language Models

Class-based language models are the state-of-the-art approaches for language modeling. The main task of the approach is to replace the statistical dependencies between words with dependencies between a much lower number of word classes, thus reducing the data sparsity problem.

Let W denote the set of possible words (word vocabulary) and C denote a class vocabulary. Then we can define a mapping function $m : W \to C$, which maps every word $w_i \in W$ to some $c_i \in C$. In our case, the classes are the word clusters derived from particular semantic spaces.

The probability estimation of word w_i conditioned by its history w_{i-n+1}^{i-1} (where n is the length of the n-gram) is given by the following formula

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = P\left(w_i|c_i\right) \cdot P\left(c_i|c_{i-n+1}^{i-1}\right). \tag{4}$$

We are using the Modified Kneser-Ney interpolation (introduced in [Chen and Goodman, 1998]) which at present is the state-of-the-art approach for smoothing methods. The formula for smoothing of word probabilities is

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{cnt\left(w_{i-n+1}^{i}\right) - D\left(cnt\left(w_{i-n+1}^{i}\right)\right)}{\sum\limits_{w_i} cnt\left(w_{i-n+1}^{i}\right)} + \gamma\left(w_{i-n+1}^{i-1}\right) P\left(w_i|w_{i-n+2}^{i-1}\right), \tag{5}$$

where $P()$ is the probability given by the Modified Kneser-Ney interpolation model and $cnt()$ is the count of n-gram. The goal of discounting function $D(cnt)$ is to save some probability mass for lower-order models. The normalization function $\gamma(w_{i-n+1}^{i}) \in (0, 1)$ makes the probability distribution sum up to 1. The definitions and derivations of this functions can be found in the original paper.

The main advantage of Modified Kneser-Ney smoothing is the clever way it calculates the unigram probability distribution

$$P\left(w_i\right) = \frac{N_{1+}\left(\bullet w_i\right)}{N_{1+}\left(\bullet\bullet\right)}, \tag{6}$$

where symbol $\bullet$ means an arbitrary word (class) and $N_r(w_{i-n+1}^{i})$ is the number of n-grams with frequency r (i.e. the number of such n-grams, where $cnt(w_{i-n+1}^{i}) = r$). In different words, the unigram probability of w_i is given by the number of different bigrams ending in w_i divided by the total number of different bigrams.

3.2. Combining Language Models

We use a simple but very effective linear interpolation for combining different language models

$$P^{LI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right), \tag{7}$$

where λ_k is the weight of the k-th language model $P_k()$. We use the *Expectation Maximization (EM)* algorithm described in [Dempster et al., 1977] to calculate optimal weights λ_k by way of maximization of the probability of the held-out data.

The linear interpolation can be extended to a method called bucketed linear interpolation, where weights become the function of the frequency of word history [Bahl et al., 1983]. The main idea is that the weights λ_k should be different for words with histories of varying frequencies. The formula then transforms to

$$P^{BLI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k\left(w_{i-n+1}^{i-1}\right) \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right). \tag{8}$$

The weights $\lambda_k()$ certainly cannot be different for each possible frequency of history. Too much weights would be created and too little data would be available to train them. Instead, the whole frequency spectrum is divided into buckets, where each bucket holds some range of frequencies. Histories in buckets have the same weights. The number of buckets can be tuned in but it generally depends on the amount of training data available. The more training data are available, the more buckets can be used.

4. Experimental results

In this section we closely describe various results of our experiments. The first subsection 4.1 gives an overview of the corpora we use. Subsection 4.2 describes the configuration of the tested language models. Perplexity results, as the most commonly used metric for language models, are shown in the next section 4.3. Finally, two additional tests of the performance of the language models are presented (sections 4.4 and 4.5).

4.1. Text Corpora

Language models in our experiments were trained on three unlabeled corpora in Czech, Slovak and English. Czech and Slovak belong to the group of Slavic languages and are representatives of highly inflectional languages. We also show tests on an English corpus as a representative of a language with low inflection. English is also used to compare the state-of-the-art approaches with our model.

- **CZ**: Contains news on many topics such as political, business, sports, international and other news gathered from one year in the Czech language. Data in this corpus are provided by the Czech News Agency (CNA).

- **SK**: A huge number of texts from the Slovak National Corpus[3] oriented also towards the artistic, publicist and professional area.
- **EN**[4]: 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 – December 31, 1990.

The data from all corpora were divided into training, held-out and testing sets in proportions of 65%, 15% and 20%. The parameters of these corpora are shown in table 1.

Table 1: Parameters of corpora used for experiments.

	CZ			
	train	held-out	test	full
number of tokens	23.1M	5.5M	5.2M	33.8M
words occurred min. 1	472.4k	237.1k	231.1k	650.5k
words occurred min. 5	149.6k	64.4k	62.8k	183.2k
words occurred min. 10	96k	38.5k	37.6k	119.7k
	SK			
	train	held-out	test	full
number of tokens	60.9M	8.7M	17.4M	86.9M
words occurred min. 1	883.8k	361.3k	503k	1,029k
words occurred min. 5	277.9k	92.8k	140.2k	333.5k
words occurred min. 10	183.8k	56.1k	88.1k	223.1k
	EN			
	train	held-out	test	full
number of tokens	51M	10.5M	14M	75.5M
words occurred min. 1	352.6k	157.5k	182.9k	434k
words occurred min. 5	115.2k	53.3k	61.5k	139.3k
words occurred min. 10	79.8k	35.2k	41.2k	96.6k

4.2. Language models

All of testing algorithms for building semantic spaces are implemented with the open source package called S-Space [Jurgens and Stevens, 2010].

As a *baseline model* we use the 4-gram word language model smoothed by the Modified Kneser-Ney smoothing described in subsection 3.1. The higher order models prove not to be computationally feasible due their high memory requirements. The baseline language model as well as the semantic spaces are trained on the *training* parts of corpora. Only words with a reasonable frequency in the training data (5 or more for all corpora) are taken into account during building all of our models.

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

[4]Material is available from NIST Standard Reference Data Products.

The vocabulary size for each language is shown in table 1. Corpora are tokenized with our language-independent tokenizer based upon regular expressions. The word dictionary is generated also from the training parts and it is kept fixed for all tests.

To build a *class-based model*, we use equation 4 and the Modified Kneser-Ney smoothing for the classes. The output of semantic spaces (the vectors for each word in the vocabulary) is kept unchanged. We do not use normalization because the principle of semantic space methods ensures that all vectors are normalized. The classes are generated by the Repeated Bisection clustering algorithm (described in section 2.6.2) with the cosine metric (equation 2) – **COS** and the Pearson product-moment correlation coefficient (equation 3) – **CORR**. The data are clustered into 5 different numbers of clusters: 1,000, 5,000, 10,000, 20,000 and 50,000. Clustering into more than 50,000 classes would be computationally too expensive. Moreover such a high number of classes would mean that the classes would be very small containing often one word per class. The input data for clustering are the vectors from semantic spaces. We test 5 different semantic spaces with two settings for each – see table 2. We used the recommended settings given by authors of particular methods. In addition to the recommended setting, one more setting per method was added. The settings were based upon our understanding of method principles.

The *final language models* are created as the linear interpolation of the baseline model and class-based models. We always interpolate all five class-based models (different number of classes, same semantic space) and the baseline n-gram model. To improve the interpolation, we use 20 buckets. The weights of interpolation were estimated on the *held-out* parts of corpora.

During our experiments we found out that the HAL method is better for dense clusters (1k, 5k and 10k classes) and the COALS method performs better for sparse clusters (20k and 50k classes). We therefore tried to interpolate HAL for dense clusters and COALS for sparse clusters together. The results are denoted by the following notation *HAL_w4+COALS_noSVD* and presented at the end of all tables.

Table 2: Semantic spaces settings. The symbol $|W|$ denotes the number of distinct words in the model. The first order context vectors are used in both configuration of Purandare and Pedersen (PP) model. For Random Indexing model the word order information was not used.

semantic space	number of columns	window size	similarity metric	other settings
HAL_w4	50,000	4	cos	
HAL_w8	50,000	8	cos	
COALS_noSVD	14,000	4	cos	without SVD reduction
COALS_SVD	1,024	4	corr	SVD reduction
BEAGLE_512	512	3	corr	
BEAGLE_1024	1,024	3	corr	
RI_w4	1,024	4	cos	word-order info unset
RI_w6	1,024	6	cos	word-order info unset
PP_m1	$\|W\|$	5	cos	1 meaning for each word
PP_m3	$\|W\|$	5	cos	3 meanings for each word

4.3. Perplexity results

In this subsection, we present the perplexities of several class-based language models created from semantic spaces and their linear interpolations with the baseline model. The results for 4-gram language models are presented in tables 3a, 3b, 3c. The numbers in bold show the best result for the given number of classes. The numbers in brackets are the relative improvements against the baseline. For comparison we present the results given by both the *linear interpolation* as well as the *bucketed linear interpolation*.

Table 3: 4-gram perplexity results on CZ, SK and EN corpus.

(a) CZ

Baseline	268						
	Number of classes					Linear	Bucketed
	1k	5k	10k	20k	50k	interpolation	interpolation
HAL_w4	**1,091**	**644**	**526**	493	444	231 (-13.8%)	229 (-14.6%)
HAL_w8	1,199	721	614	550	521	237 (-11.6%)	234 (-12.7%)
COALS_noSVD	1,515	771	528	**389**	**331**	235 (-12.3%)	231 (-13.8%)
COALS_SVD	1,811	805	587	454	391	261 (-2.6%)	260 (-3.0%)
BEAGLE_512	1,346	912	800	686	562	239 (-10.8%)	236 (-11.9%)
BEAGLE_1024	1,322	884	750	657	544	237 (-11.6%)	234 (-12.7%)
RI_w4	1,127	710	600	503	421	233 (-13.1%)	230 (-14.2%)
RI_w6	1,205	712	619	530	436	235 (-12.3%)	232 (-13.4%)
P&P_m1	1,351	698	540	496	405	263 (-1.9%)	263 (-1.9%)
P&P_m3	1,438	733	541	480	399	263 (-1.9%)	262 (-2.2%)
HAL_w4+COALS_noSVD	**1,091**	**644**	**526**	**389**	**331**	**225 (-16.0%)**	**220 (-17.9%)**

(b) SK

Baseline	279						
	Number of classes					Linear	Bucketed
	1k	5k	10k	20k	50k	interpolation	interpolation
HAL_w4	**1,482**	924	782	701	580	243 (-12.9%)	240 (-14.0%)
HAL_w8	1,895	1,221	835	755	629	250 (-10.4%)	245 (-12.2%)
COALS_noSVD	2,037	1,118	837	628	**463**	242 (-13.3%)	238 (-14.7%)
COALS_SVD	2,256	1,226	919	711	497	272 (-2.5%)	270 (-3.2%)
BEAGLE_512	1,651	1,166	1,063	935	841	260 (-6.8%)	258 (-7.5%)
BEAGLE_1024	1,512	1,088	990	892	799	256 (-8.2%)	253 (-9.3%)
RI_w4	1,824	1,023	966	774	631	255 (-8.6%)	252 (-9.7%)
RI_w6	1,911	1,210	1,058	892	493	257 (-7.9%)	254 (-9.0%)
P&P_m1	1,755	**897**	**721**	596	510	275 (-1.4%)	274 (-1.8%)
P&P_m3	1,760	902	729	**589**	503	273 (-2.2%)	272 (-2.5%)
HAL_w4+COALS_noSVD	**1482**	924	782	628	**463**	**238 (-14.7%)**	**234 (-16.1%)**

(c) EN

Baseline	189						
	Number of classes					Linear	Bucketed
	1k	5k	10k	20k	50k	interpolation	interpolation
HAL_w4	**550**	496	423	409	378	173 (-8.5%)	172 (-9.0%)
HAL_w8	760	622	549	521	494	177 (-6.3%)	176 (-6.9%)
COALS_noSVD	791	468	346	280	**214**	175 (-7.4%)	175 (-7.4%)
COALS_SVD	665	344	296	269	218	178 (-5.8%)	177 (-6.3%)
BEAGLE_512	577	459	434	398	376	176 (-6.9%)	175 (-7.4%)
BEAGLE_1024	556	454	439	426	365	175 (-7.4%)	174 (-7.9%)
RI_w4	621	516	475	435	349	175 (-7.4%)	174 (-7.9%)
RI_w6	663	527	498	435	376	175 (-7.4%)	175 (-7.4%)
P&P_m1	584	**330**	274	**238**	215	183 (-3.2%)	183 (-3.2%)
P&P_m3	576	342	**272**	245	216	184 (-2.6%)	184 (-2.6%)
HAL_w4+COALS_noSVD	**550**	496	423	280	**214**	**171 (-9.5%)**	**170 (-10.1%)**

The numbers in the tables above show that almost every semantic space gives at least some improvement. In particular, we can see that for inflectional languages (CZ and SK corpora) the improvements in perplexities are more significant than for the English corpus (EN) (a representative of a low inflectional language).

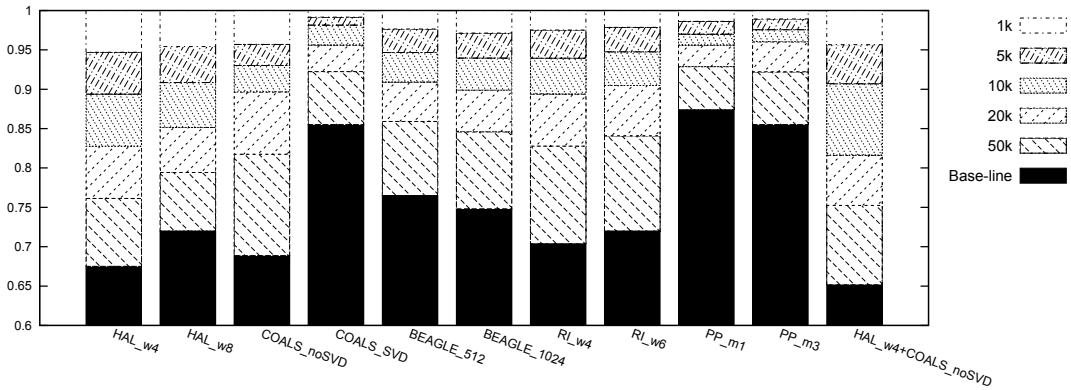
As we indicated before, the HAL-based models are efficient in case of dense clusters (1,000 or 5,000 classes) and the COALS model looks more suitable for sparse clusters (20,000 or 50,000 classes). The combination of both (the last row of the tables) provides the best perplexity results for each corpus. In following subsections, the HAL+COALS-based language model will be tested in order to verify its performance on other tests.

It is also interesting that the linear interpolation of class-based models created from the P&P model does not improve perplexity against baseline, however, each of the class-based language models alone gives very low perplexity in comparison to other semantic spaces. In many cases these class-based models even give the lowest perplexity when they are not interpolated with the standard n-gram model.

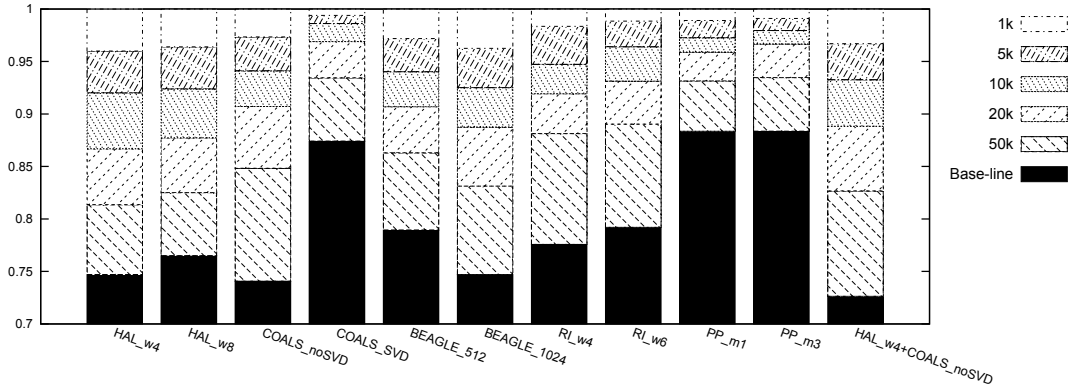
4.3.1. Interpolation weights

In this subsection we show the interpolation weights of particular sub-models that form the final model. Only the simple interpolation is depicted (single weight for each sub-model). Figures with bucketed interpolation would not be readable due to high number of weights. Weights are depicted in figures 1a, 2a and 3a. The impact of each sub-model on the final probability distribution can be easily seen from the length of particular sub-bars.

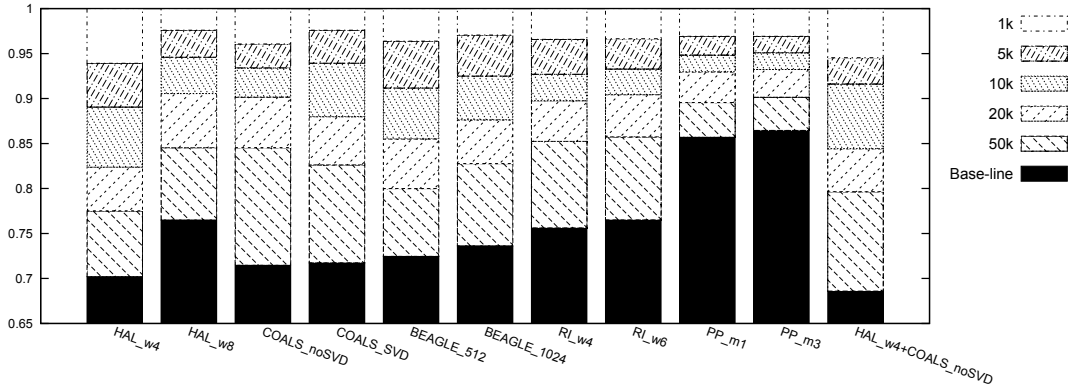
Figures show a trend where the sparse clusters (20k or 50k classes) received bigger weights (immediately after the baseline) and the weights decrease with decreasing number of classes. A direct correlation between weights and perplexity results (section 4.3) can be seen. The bigger the weights allocated by the EM algorithm are, the higher the improvement in perplexity is achieved.



(a) CZ



(a) SK



(a) EN

Figure 3: Interpolation weights of 4-gram language models on CZ, SK and EN corpus.

4.4. Word estimation test

The perplexity sometimes does not correspond with results from real-world applications. Here we introduce another test of our language models called the *word estimation test*.

During the test, our models try to find a missing word in the sentence. Since the selection of a correct word from a full vocabulary would be computationally very expensive, we use the list of the most frequent words (in our tests 1000 words) from which the language model tries to find the correct word. Of course, the list is constructed in such a way that it always contains the correct word (the word that was originally in the focused place in the sentence). The stop words were skipped during this test.

During the test, the words are sorted according to the probability given by the language model. The test then measures the average order of where the model placed the correct word. The best result is when the model places the word as the

first word. If the wrong word is chosen, it matters how far the correct word from the top of the list is. We assume that the closer to the top the correct word is the easier the error can be corrected by the system.

The test is formalized as follows. During the m-th test, let O_m denote the order of the correct answer in the list of possibilities, which is sorted according to the language model. The expectation value of order $E(O)$ is then defined as

$$E(O) = \frac{1}{M} \sum_{m=1}^{M} O_m, \qquad 1 \leq m \leq M, \tag{9}$$

where M means the total number of words in progress.

Similarly to the previous subsection, the numbers in bold in the tables below (4a, 4b, 5a) are the best from all semantic spaces, and the numbers in brackets are relative improvements against the baseline.

Although this is not a standard test, we believe that it may provide more fine-grained performance evaluation than other tests. The test distinguishes between really wrong answers (the correct word is far from the beginning of the list) and almost correct answers (the correct word is close to the beginning). The expectation value helps to create a reasonable idea about the probability distributions of different models.

Table 4: Word estimation test results with 4-gram language model on CZ, SK and EN corpus.

(a) CZ

Baseline	69.7					
	Number of classes					Bucketed
	1k	5k	10k	20k	50k	interpolation
HAL_w4	**101.7**	**78.4**	**72.7**	**71.3**	70.4	62.0 (-11.0%)
HAL_w8	109.2	90.3	78.7	76.1	74.6	63.9 (-8.3%)
COALS_noSVD	128.3	95.7	83.2	73.4	**70.0**	65.5 (-6.0%)
COALS_SVD	162.0	117.3	100.4	84.1	76.2	67.6 (-3.0%)
BEAGLE_512	132.3	105.2	98.4	90.6	82.3	67.8 (-2.7%)
BEAGLE_1024	129.1	102.3	93.9	87.6	80.3	66.7 (-4.3%)
RI_w4	114.5	89.3	82.9	75.8	70.8	63.6 (-8.8%)
RI_w6	115.4	87.7	83.8	78.1	71.6	63.7 (-8.6%)
P&P_m1	155.4	118.3	103.9	95.6	87.2	69.1 (-0.9%)
P&P_m3	144.8	112.2	101.5	92.8	84.6	68.3 (-2.0%)
HAL_w4+COALS_noSVD	**101.7**	**78.4**	**72.7**	73.4	**70.0**	**60.2 (-13.6%)**

(b) SK

Baseline	61.5					
	Number of classes					Bucketed
	1k	5k	10k	20k	50k	interpolation
HAL_w4	**102.8**	76.7	70.6	67.8	61.3	53.2 (-13.5%)
HAL_w8	113.9	83.4	76.0	71.8	64.8	54.8 (-10.9%)
COALS_noSVD	113.8	**75.9**	**66.4**	**60.8**	**57.8**	52.2 (-15.1%)
COALS_SVD	145.9	100.0	82.9	68.2	65.2	59.8 (-2.8%)
BEAGLE_512	141.2	110.4	103.1	97.3	86.9	60.6 (-1.5%)
BEAGLE_1024	135.5	107.3	100.0	94.8	84.8	59.9 (-2.6%)
RI_w4	122.8	88.0	83.8	76.5	67.7	57.1 (-7.2%)
RI_w6	123.9	88.4	84.6	77.0	68.7	57.3 (-6.8%)
P&P_m1	160.5	117.0	101.1	92.6	84.4	60.9 (-1.0%)
P&P_m3	157.4	113.3	99.5	87.9	82.1	60.4 (-1.8%)
HAL_w4+COALS_noSVD	**102.8**	76.7	70.6	**60.8**	**57.8**	**51.9 (-15.6%)**

(a) EN

Baseline	42.1					
	Number of classes					Bucketed
	1k	5k	10k	20k	50k	interpolation
HAL_w4	**70.0**	**60.9**	56.3	55.4	49.4	39.1 (-7.1%)
HAL_w8	83.6	71.4	64.8	64.0	59.3	40.2 (-4.5%)
COALS_noSVD	87.7	62.8	52.9	**46.8**	**42.8**	40.5 (-3.8%)
COALS_SVD	91.5	62.1	**52.6**	48.8	44.6	41.7 (-1.0%)
BEAGLE_512	81.5	70.9	66.7	63.8	59.6	40.6 (-3.6%)
BEAGLE_1024	82.7	71.2	66.9	64.9	58.9	40.7 (-3.3%)
RI_w4	81.2	65.3	63.1	60.3	53.9	40.1 (-4.8%)
RI_w6	81.8	66.7	64.4	61.3	55.1	40.6 (-3.6%)
P&P_m1	103.6	73.2	63.4	54.7	47.5	41.8 (-0.7%)
P&P_m3	98.9	71.3	62.0	53.8	46.9	41.9 (-0.5%)
HAL_w4+COALS_noSVD	**70.0**	**60.9**	56.3	**46.8**	**42.8**	**38.6 (-8.3%)**

From the tables above it is easy to see that the HAL and COALS models absolutely lead the word estimation test for all tested languages. Their combination created the best improvement when compared to the baseline (for Czech and Slovak more than 10% improvement in the average order of the correct word).

4.5. Machine translation

This section describes the performance of proposed language models in a machine translation task. The success in this task should verify the ability of the models to improve performance of a real-world application. The system used in this

test is based upon the statistical machine translation toolkit called Moses[5], briefly described in [Koehn et al., 2007]. To train the system the instructions for building a baseline system were accurately followed. The parallel corpora used for training consisted of texts in Czech, Slovak and English languages from European Parliament proceedings corpus (EuroParl)[6] version 6. Statistics of used corpora are shown in table 5. Training parameters are in table 6. Machine translation experiments with the EuroParl corpus are presented e.g. in [Koehn, 2005].

Table 5: Statistics of the EuroParl corpus version 6. The numbers are calculated after tokenization and sentence alignment. The *train* part of corpus was used for training Moses. The *held-out* part was used for tuning the weights of particular modules of Moses. The *test* part was used for evaluating experiments.

	from EN to SK		from EN to CZ	
	EN	SK	EN	CZ
train sentences	300,000	300,000	300,000	300,000
train tokens	8,004,022	6,933,404	7,967,658	6,849,087
train words	45,864	144,696	53,511	141,753
held-out sentences	50,000	50,000	50,000	50,000
held-out tokens	1,351,147	1,155,286	1,342,095	1,143,846
held-out words	22,165	59,478	23,727	58,126
test sentences	110,779	110,779	112,351	112,351
test tokens	2,992,433	2,549,971	3,025,821	2,598,880
test words	29,163	84,407	32,402	84,755

Table 6: Parameters used for training the machine translation system.

Casing normalization	Off
Minimum-maximum tokes per sentence	1-80
Language model order	3
Language model smoothing	Modified Kneser-Ney
Alignment heuristic	grow-diag-final-and
Reordering model	msd-bidirectional-fe

In our experiments the translation decoding was done in two steps. In the first step the Moses n-best decoder was used for generating 500 best hypotheses. The Moses standard 3-gram language model was used for speeding up the decoding phase. In the second step the hypotheses were re-scored by our 4-gram language models by replacing the Moses language models scores with the scores of our models.

The results are evaluated by the widely used BLEU metric [Papineni et al., 2002], which measures n-gram overlap with reference translations. Results are presented in tables 7a, 7b and 8a.

[5]Available at `http://www.statmt.org/moses/`.
[6]Available at `http://www.statmt.org/europarl/`.

Table 7: BLEU scores achieved by using 4-gram language models in machine translation. The bold numbers are the bests for the given number of classes. The numbers in brackets are absolute improvements in BLEU points against the baseline.

(a) translation from EN to CZ

Baseline	21.91					
	Number of classes					Bucketed
	1k	5k	10k	20k	50k	interpolation
HAL_w4	**19.44**	**20.28**	**20.63**	20.75	20.92	22.46 (0.55)
HAL_w8	19.26	20.24	20.55	20.72	20.81	22.39 (0.48)
COALS_noSVD	18.14	19.59	20.61	**21.28**	**21.52**	22.42 (0.51)
COALS_SVD	18.06	19.41	20.22	20.93	21.26	22.01 (0.10)
BEAGLE_512	19.13	19.66	19.81	20.08	20.73	22.19 (0.28)
BEAGLE_1024	19.19	19.71	19.86	20.10	20.78	22.28 (0.37)
RI_w4	19.41	19.74	20.10	20.49	20.67	22.45 (0.54)
RI_w6	19.35	19.71	20.11	20.44	20.71	22.39 (0.48)
P&P_m1	18.74	19.73	20.02	20.34	21.18	21.95 (0.04)
P&P_m3	18.65	19.69	20.07	20.51	21.23	21.98 (0.07)
HAL_w4+COALS_noSVD	**19.44**	**20.28**	**20.63**	**21.28**	**21.52**	**22.63 (0.72)**

(b) translation from EN to SK

Baseline	24.46					
	Number of classes					Bucketed
	1k	5k	10k	20k	50k	interpolation
HAL_w4	**22.75**	23.28	23.56	23.89	24.01	25.05 (0.59)
HAL_w8	22.32	23.19	23.47	23.84	23.92	24.89 (0.43)
COALS_noSVD	22.14	22.92	23.38	24.06	**24.25**	25.08 (0.62)
COALS_SVD	21.22	21.93	23.15	23.86	24.17	24.61 (0.15)
BEAGLE_512	21.75	22.20	22.82	23.52	23.63	24.65 (0.19)
BEAGLE_1024	21.87	22.34	22.96	23.66	23.71	24.75 (0.29)
RI_w4	21.66	23.19	23.43	23.78	24.05	24.82 (0.36)
RI_w6	21.62	23.21	23.40	23.77	24.21	24.80 (0.34)
P&P_m1	21.85	23.41	**23.63**	24.11	24.15	24.45 (-0.01)
P&P_m3	21.91	**23.46**	23.59	**24.19**	24.17	24.49 (0.03)
HAL_w4+COALS_noSVD	**22.75**	23.28	23.56	24.06	**24.25**	**25.12 (0.66)**

(a) translation from CZ to EN

Baseline	32.59					
	Number of classes					Bucketed
	1k	5k	10k	20k	50k	interpolation
HAL_w4	30.21	30.43	30.60	31.10	31.48	32.89 (0.30)
HAL_w8	30.03	30.21	30.35	30.69	31.11	32.73 (0.14)
COALS_noSVD	29.42	30.45	31.24	32.10	**32.36**	32.84 (0.25)
COALS_SVD	30.14	30.84	31.89	32.16	32.31	32.77 (0.18)
BEAGLE_512	30.19	30.98	31.02	31.24	31.50	32.83 (0.24)
BEAGLE_1024	**30.30**	31.04	30.95	31.24	31.51	32.88 (0.29)
RI_w4	29.87	30.26	30.47	30.53	31.33	32.85 (0.26)
RI_w6	29.75	30.20	30.45	30.61	31.29	32.81 (0.22)
P&P_m1	30.01	**31.60**	32.10	32.20	32.32	32.64 (0.05)
P&P_m3	30.05	31.49	**32.14**	**32.27**	32.28	32.61 (0.02)
HAL_w4+COALS_noSVD	30.21	30.43	30.60	32.10	**32.36**	**32.96 (0.37)**

The results clearly show a significant performance gain in machine translation from English into Czech and Slovak languages. Even translation from Czech and Slovak into English showed some increase in BLEU points. The HAL and COALS combined model is again the best performing one. The performance of other models indicate the same trends as in previous tests. This test is a solid proof that the proposed language models are usable in a real-world application.

5. Discussion

In the previous subsections we showed several experiments that tried to compare and test our language models in different tasks. Perplexities, machine translation performance and average word orders describe the behavior of language models and help to give us an idea about the probability distribution of language models. However, the semantic spaces tested in this work may perform differently in different applications in NLP; the results of their usage in language modeling are summarized below.

During our experiments, we found that semantic spaces have different behavior when we cluster words into differing numbers of clusters. Some of the semantic spaces (e.g. HAL model) are better for dense clusters (1,000 or 5,000 classes), some of them (e.g. COALS model) are more suitable for sparse clusters (20,000 or 50,000 classes). The combination of HAL-based language models for small number of clusters together with COALS-based models for greater number of clusters gave the best results.

Some thoughts were given into the analysis of why HAL model perform better on dense clusters and COALS is better for sparse clusters. It has to be noted that the following ideas can be hardly proven and have to be considered only as speculations. The basic difference between HAL and COALS in our experiments is that HAL creates vectors with dimension 50000. COALS uses dimension only 14000

and modifies vectors by correlation, zeroing the negative numbers and enhancing the positive ones. We believe that the impact of these differences is that COALS can capture more precisely very similar words. The HAL on the other hand benefits from a larger vector and describes more precisely no so much similar words. The results is that COALS is better to create sparse clusters (consisted of more similar words) and HAL is better for dense clusters (not so similar words).

As expected we achieved better results for inflectional languages. For Czech and Slovak, the best improvement in perplexity was 17.9% and 16.1% respectively. For the average order of the correct word in the word estimation test, we achieved a 13.6% and 15.6% relative reduction. The improvement in the machine translation was 0.72 and 0.66 of BLEU points. For English, the improvement was not so significant. Perplexity was reduced by about 10.1%. The word estimation test was improved by about 8.3% and the increase in BLEU metrics of 0.37 points was achieved.

Our explanation is as follows. We discovered that during word clustering, the semantic spaces have a tendency to connect together words with similar morphological forms. In an ideal case, the semantic space should merge together those words that have the same meaning and simultaneously the same morphological form (the same morphological tag). According to several studies in morphological-based language models [Vaiciunas et al., 2004; Kirchhoff et al., 2006; Oparin, 2008; Brychcín and Konopík, 2011], the impact of morphological analysis for modeling of inflectional languages is much more significant than for modeling of low inflection languages. We believe that this is the main reason why the techniques described in this article are more suitable for inflectional languages.

In our experiments we also uncovered several interesting facts. The COALS model with SVD reduction performs very badly for Czech and Slovak. We believe this is caused by the data sparsity problem. For English, the results of the model are better, however, its combination with the baseline model gives no improvement. Similarly, the P&P model performs very well when we investigate the stand-alone class-based language models, but improvement against baseline is negligible. What is noteworthy is that the Random Indexing gives each of the tested languages some improvement but it is worse than the HAL or COALS model. Finally, the BEAGLE model (a similar approach to word meaning modeling as Random Indexing) also produces negligible improvement when compared to the baseline.

6. Summary

6.1. Future work

In this work, we concentrated solely on a way to improve the probability estimates with the information that is already in the data. We wanted to use no external information. We see yet another possibility to improve the models without external information. The work will focus on an analysis of OOV[7] words. The idea is based

[7]Out of vocabulary (OOV) word means an unknown (previously unseen) word for language model.

on an estimation of the semantic vectors of OOV words from their contexts and by using an unsupervised approach to morphological analysis (morpheme segmentation, stemming, etc.). Our findings lead us to believe that OOV words are the central weakness of many language models. We believe that by focusing on OOV words we may achieve a very significant performance boost.

The experiments with the Moses machine translation framework revealed that it is an extraordinary framework with great extension capabilities. In the future we plan to experiment with factored translation allowing the usage of class information directly in the training / decoding phase. We expect that by the tighter link between Moses and our language models, even better results can be achieved.

6.2. Conclusion

In this article we tried to reach a really hard target. We tried to beat n-gram language models merely with better probability estimates and without any external knowledge (morphology, syntax, partitioning into documents). We choose not to do this in a simpler way where n-gram models are weak, we ignored low occurring words and we instead concentrated on the modeling of words with reasonable frequency in the data (five or more). By using semantic spaces and clustering, we were able to improve the probability estimates and achieve significant improvements. The improvement was detected in both synthetic and real-world tests.

To be completely honest, we must note that our models in their current state are not very suitable for practical use. The clustering phase (the most computationally intensive phase) may take up to one week in a powerful computational unit. The models are also very memory demanding since the final language model is a combination of the 4-gram model and several class based models. We believe that there is great room for optimization as there is much redundancy in all the models. By clever pruning, a huge memory saving may be achieved. This was however not the point of our efforts. Our work was rather the proof-of-concept effort. We concentrated on finding a new possibility for improving language modeling. We can state that the semantic class based language models are suitable for building large corpora language models.

We think that our article may also be useful in another way. The semantic spaces presented in the article are quite a new branch of corpus statistics. It is challenging to measure the performance of semantic spaces since it is hard to find an objective criterion. We believe that application in language modeling may be used as an objective criterion that helps to compare semantic spaces with one another. Taking into account the results and our findings during testing, we can recommend the HAL and COALS models. Especially the combination HAL and COALS models seems to provide very good results. We can also recommend the RI models even though they did not win any test. They were, however, able to provide consistently very good results and are computationally very undemanding.

Acknowledgement

This work was supported by grant no. SGS-2010-028, 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 (CNA) for providing a huge number of texts in Czech. We would like to thank David Jurgens and Dr. Keith Stevens for the implementation methods for building semantic spaces [Jurgens and Stevens, 2010] we use in this work.

References

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

Bai, S., Li, H., Lin, Z., Yuan, B., 1998. Building class-based language models with contextual statistics. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 173–176.

Bellegarda, J. R., Aug. 2000. Exploiting latent semantic information in statistical language modeling. Proceedings of the IEEE 88 (8), 1279 –1296.

Bellegarda, J. R., Butzberger, J. W., Chow, Y. L., Coccaro, N. B., Naik, D., 1996. A novel word clustering algorithm based on latent semantic analysis. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 172–175.

Blei, D. M., Ng, A. Y., Jordan, M. I., Lafferty, J., 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3, 2003.

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.

Burgess, C., Lund, K., 1997. Modelling parsing constraints with high-dimensional context space. Language and Cognitive Processes 12, 177–210.

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.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407.

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.

Gao, J., Goodman, J. T., Miao, J., 2002. The use of clustering techniques for language modeling application to asian languages. Computational Linguistics.

Gildea, D., Hofmann, T., 1999. Topic-based language models using em. In: Proceedings of Eurospeech. pp. 2167–2170.

Hahn, S., Sethy, A., Kuo, H. J., Ramabhadran, B., 2008. A study of unsupervised clustering techniques for language modeling. Proceedings of Interspeech, 1598–1601.

Hofmann, T., 1999. Probabilistic latent semantic analysis. In: Proceedings of 15th Conference on Uncertainty in Artificial Intelligence. pp. 289–296.

Jones, M. N., Mewhort, D. J. K., 2007. Representing word meaning and order information in a composite holographic lexicon. Psychological Review 114, 1–37.

Jurgens, D., Stevens, K., 2010. The s-space package: An open source package for word space models. In System Papers of the Association of Computational Linguistics.

Karypis, G., 2003. Cluto - a clustering toolkit.
URL www.cs.umn.edu/~karypis/cluto

Kirchhoff, K., Vergyri, D., Bilmes, J., Duh, K., Stolcke, A., Oct. 2006. Morphology-based language modeling for conversational Arabic speech recognition. Computer Speech and Language 20 (4), 589–608.

Koehn, P., 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In: Machine Translation Summit X. Phuket, Thailand, pp. 79–86.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E., 2007. Moses: open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. ACL '07. Association for Computational Linguistics, pp. 177–180.

Landauer, T. K., Dumais, S. T., 1997. Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. Psychological Review (104).

Landauer, T. K., Foltz, P., Laham, D., 1998. An Introduction to Latent Semantic Analysis. Discourse Processes (25), 259–284.

Liu, F., Liu, Y., 2007. Unsupervised language model adaptation incorporating named entity information. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics. Association for Computational Linguistics, pp. 672–679.

Liu, Y., Liu, F., 2008. Unsupervised language model adaptation via topic modeling based on named entity hypotheses. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 4921–4924.

Lund, K., Burgess, C., 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods Instruments and Computers 28 (2), 203–208.

Maltese, G., Bravetti, P., Crepy, H., Grainger, B. J., Herzog, M., Palou, F., 2001. Combining word and class-based language models: a comparative study in several languages using automatic and manual wordclustering techniques. In: Proceedings of 7th European Conference on Speech Communication and Technology. Eurospeech, pp. 21–24.

Oparin, I., 2008. Language models for automatic speech recognition of inflectional languages. Ph.D. thesis, University of West Bohemia, Pilsen.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. ACL '02. Association for Computational Linguistics, pp. 311–318.

Purandare, A., Pedersen, T., 2004. Word sense discrimination by clustering contexts in vector and similarity spaces. Proceedings of 8th Conference on Computational Natural Language Learning, 41–48.

Rohde, D. L. T., Gonnerman, L. M., Plaut, D. C., 2004. An improved method for deriving word meaning from lexical co-occurrence. Cognitive Psychology 7, 573–605.

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

Sahlgren, M., 2005. An Introduction to Random Indexing. Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005.

Sahlgren, M., Holst, A., Kanerva, P., 2008. Permutations as a means to encode order in word space. Proceedings of the 30th Annual Conference of the Cognitive Science Society, 1300–1305.

Tam, Y., Schultz, T., 2005. Dynamic language model adaptation using variational bayes inference.

In: Proceedings of Interspeech. pp. 5–8.

Tam, Y., Schultz, T., 2006. Unsupervised language model adaptation using latent semantic marginals. In: Proceedings of Interspeech.

Vaiciunas, A., Kaminskas, V., Raškinis, G., 2004. Statistical language models of lithuanian based on word clustering and morphological decomposition. Informatica 15 (4), 565–580.

Wang, S., Schuurmans, D., Peng, F., Zhao, Y., 2003. Semantic n-gram language modeling with the latent maximum entropy principle. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP03).

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

Whittaker, E. W. D., 2000. Statistical language modelling for automatic speech recognition of russian and english. Ph.D. thesis, Cambridge University, Cambridge, MA, USA.

Whittaker, E. W. D., Woodland, P. C., 2003. Language modelling for Russian and English using words and classes. Computer Speech and Language 17, 87–104.

Yamamoto, H., Sagisaka, Y., 1999. Multi-class composite n-gram based on connection direction. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing.

Yokoyama, T., Shinozaki, T., Iwano, K., Furui, S., 2003. Unsupervised language model adaptation using word classes for spontaneous speech recognition. In: Proceedings of IEEE-ISCA Workshop on Spontaneous Speech Processing and Recognition. pp. 71–74.

Zhao, Y., Karypis, G., 2002. Criterion functions for document clustering: Experiments and analysis. Tech. rep., Department of Computer Science, University of Minnesota, Minneapolis.

A.3 Latent semantics in language models

Journal title:	Computer Speech & Language
Journal ISSN:	0885-2308
Article DOI:	10.1016/j.csl.2015.01.004
Received at Editorial Office:	5 December, 2013
Article accepted for publication:	15 January, 2015
Volume/issue:	33/1
Pages:	88–108

Latent semantics in language models

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

This paper investigates three different sources of information and their integration into language modelling. Global semantics is modelled by Latent Dirichlet Allocation and brings long range dependencies into language models. Word clusters given by semantic spaces enrich these language models with short range semantics. Finally, our own stemming algorithm is used to further enhance the performance of language modelling for inflectional languages.

Our research shows that these three sources of information enrich each other and their combination dramatically improves language modelling. All investigated models are acquired in a fully unsupervised manner.

We show the efficiency of our methods for several languages such as Czech, Slovenian, Slovak, Polish, Hungarian, and English, proving their multilingualism. The perplexity tests are accompanied by machine translation tests that prove the ability of the proposed models to improve the performance of a real-world application.

Keywords: Language Models, Latent Dirichlet Allocation, Semantic spaces, Stemming, HAL, COALS, Random Indexing, HPS, LDA, Machine translation, Moses

1. Introduction

Language modelling is an essential part in many tasks of natural language processing (NLP). Speech recognition, machine translation, optical character recognition, and many other disciplines strongly depend on the language model and thus every improvement in language modelling can also improve the performance of the whole system.

In this paper we explore fully unsupervised methods for language modelling (which require no labelled data and no information about language itself). To prove

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

their multilingualism we experiment with several languages including highly inflectional as well as low-inflection languages. We incorporate three different families of languages (Slavic, Uralic, and Germanic) into our experiments. As representatives of Slavic languages we experiment with Czech, Slovenian, Slovak, and Polish. Uralic languages are represented by Hungarian, and Germanic languages by English. All languages we investigate in this paper except English are characterized by a high level of inflection and relatively free word order (from the syntactic point of view, the words in a sentence can usually be reordered in several ways to carry a slightly different meaning). Properties of these languages complicate the language modelling task. The great number of word forms and large number of possible word sequences lead to a much higher number of n-grams. Data sparsity is a common problem of language models, but for highly inflected languages this problem is even more evident.

The highly inflected languages in this paper belong rather among non-mainstream languages, for which the language modelling task has not gained as much attention as it has for English, for example. We thus believe there is considerable potential for improvements. However we provide experiments also for English to compare our methods with the state of the art.

In this paper we extend our work on the application of semantic spaces in language modelling [Brychcín and Konopík, 2014], where we have achieved significant improvements in perplexity and in machine translation task especially with HAL, COALS and RI models. Thus, these models are investigated more deeply in this paper.

We attempt to improve language modelling by adding long-range semantic dependencies. We choose Latent Dirichlet Allocation (LDA) [Blei et al., 2003] for that task, because it has already been shown by many researchers that LDA improves language modelling (see, e.g., [Tam and Schultz, 2005, 2006; Watanabe et al., 2011]).

The performance of these language models is further enhanced by our unsupervised stemming algorithm called High Precision Stemmer (HPS)[1] introduced in [Brychcín and Konopík, 2015]. We have already tested our stemmer in language modelling tasks and the results indicate that HPS performs best compared to other unsupervised stemmers.

To the best of our knowledge we are first to try to combine these three sources of information (i.e. local semantics, global semantics, and morphology).

2. State of the art in latent semantics

In the context of this article, we work with various methods for modelling semantic relations between words, and use them to improve our language models. The backbone principle of methods for discovering hidden meaning from a plain text is the formulation of Distributional Hypothesis in [Firth, 1957] that says *"a word is*

[1] A description of the algorithm and its implementation is available at http://liks.fav.zcu.cz/HPS.

characterized by the company it keeps". The direct implication of this hypothesis is that the word meaning is related to the context where it usually occurs and thus it is possible to compare the meanings of two words by statistical comparisons of their contexts. This implication was confirmed by empirical tests carried out on human groups in [Charles, 2000].

Several authors have made huge efforts to give an overview of the current state of the art in computational methods for extracting meaning from text [Turney and Pantel, 2010; Riordan and Jones, 2011; McNamara, 2011].

All the methods for extracting meaning can be approximately summarized in two categories. Authors [Riordan and Jones, 2011; McNamara, 2011] categorize these methods into *context-word* and *context-region* approaches. In this paper we use the notation *local context* and *global context*, respectively, because we think this notation describes the principle of meaning extraction better. These two categories are briefly described in the following subsections: 2.1 and 2.2. Additionally, to give a better idea of how these two approaches differ, Figure 1 shows an example of global context and local context semantics of words.

`hockey, Pittsburgh, Jágr, 68`	`hockey, football, soccer, tennis`
`win, champion, medal, gold`	`win, lose, play, wager, defeat`
(a) Global context semantics	(b) Local context semantics

Figure 1: Examples of semantically similar words. Each row represents semantically similar words according to (a) global semantics with long range dependencies (words in the same line are likely to occur in similar contexts, but not at the same position) and (b) local semantics with short range dependencies (words in the same line should be mutually substitutable at the same position in the appropriate context).

Models based upon the distributional hypothesis usually represent the meaning as a point in a multi-dimensional space. Thus, one meaning is represented as a single vector. These models are then referred to as the vector-space models (VSM). The vector representation enables an easy comparison of word meanings by computing distances between the vectors.

2.1. Global context

Global semantics models assume that words that are close in meaning will occur in similar pieces of text (documents). These methods are usually based on the bag-of-words hypothesis, which assumes the word order has no meaning.

LSA (Latent Semantic Analysis) [Deerwester et al., 1990] is a model for discovering global semantic relationships between words. A term-document matrix is constructed and then the SVD (Singular Value Decomposition) is used to reduce the dimension of the matrix and to smooth the values of the matrix.

In [Hofmann, 1999] the PLSA (Probabilistic Latent Semantic Analysis) model was introduced, which is in fact the Bayesian version of LSA. PLSA assumes that

the document is a mixture of topics and a topic is simultaneously a mixture of words. The probabilistic output makes this model more easily applicable in many tasks.

PLSA was later extended to LDA [Blei et al., 2003], which places the Dirichlet prior to the document-topic distribution as well as the topic-word distribution. LDA is thus the proper model even for unseen documents that has often been criticized in the case of PLSA.

The motivation for using such methods in language modelling is the fact that text can continuously change in domain, topics, writing style, and so on and it is not possible to recognize these changes only from a short history of words (let us say three words if we use four-gram language models). A much larger history is needed to observe these changes, where of course the data sparsity problem is much more evident. Inhibition of word order thus leads to far fewer possible combinations of histories.

LDA is used for boosting the probabilities of words that are likely to co-occur in the same document (as will be described in Subsection 4.3). This is independent of the position of words in the document (the document is a bag of words). These word probabilities only depend on the document itself (global context). This brings long-range dependencies, and language models are thus adapted to the current domain of text.

2.2. Local context

The second approach to modelling the semantics of words is to use only their local context. Local semantic models assume that the meaning of a word is related to the short context around the word. Methods based on this assumption usually use a small context window (let us say four words in both directions). These methods do not require text that is naturally divided into documents, which can be found advantageous compared to the methods mentioned in the previous section (LSA, PLSA, LDA).

Because of the short context, these methods can take word order into account, so the methods model semantic as well as syntactic relations between words. There are a lot of methods for deriving word meaning from the local context. We have already experimented with five of them in [Brychcín and Konopík, 2014]. In this paper we continue our research and use only the three best performing methods in language modelling (HAL, COALS, and RI).

HAL (Hyperspace Analogue to Language) [Lund and Burgess, 1996] is a very simple method for building semantic space. HAL goes through the corpus and records the co-occurring words around the target word (in some small context window – typically four words in both directions). HAL distinguishes between left and right context of the target word and records them separately. Co-occurring words are weighted inversely to the distance from the target word. This results in the co-occurrence matrix $\mathbb{M} = |W| \times 2|W|$, where $|W|$ is the vocabulary size. The word vector dimension is $2|W|$ because of distinction between left and right context.

COALS (Correlated Occurrence Analogue to Lexical Semantics) [Rohde et al., 2004] is an extension of the HAL model. It starts almost identically to HAL, but

it does not distinguish between left and right context. After recording the co-occurrence information, the raw counts of the matrix $\mathbb{M}$ are converted into the Pearson's correlations. Negative values are zeroed and other values are replaced by their square roots. The optional final step, inspired by LSA [Deerwester et al., 1990], is to apply the SVD reduction to the matrix $\mathbb{M}$, resulting in the smoothing of values and also the discovery of latent semantic relationships between words.

RI (Random Indexing) [Sahlgren, 2005] uses a completely different approach to recording co-occurrence statistics compared to HAL and COALS. For each word in the vocabulary, RI starts by creating high-dimensional index vectors randomly filled with few 1s and -1s (Sahlgren et al. [2008] recommend to fill index vectors with two 1s and two -1s). The dimension is typically of the order of thousands. Such vectors are very sparse and thus unlikely to overlap. The index vectors are assumed to be nearly orthogonal. The algorithm then iterates over the corpus and for each target word it sums all the co-occurring words' index vectors. These sums are recorded in the matrix $\mathbb{M}$. Even though, RI performs a little worse than the other two methods in language modelling, we use it again because it is computationally very undemanding.

In [Brychcín and Konopík, 2014] we also tested BEAGLE [Jones and Mewhort, 2007] and P&P [Purandare and Pedersen, 2004] models, but these models did not perform as well in language modelling as the above mentioned methods.

There are several interesting methods which we have not investigated in language modelling yet, but which are certainly worth mentioning.

Similarly to [Purandare and Pedersen, 2004], the authors of [Reisinger and Mooney, 2010a,b] address the common problem of VSMs, where each word is represented with a single vector, which clearly fails to capture homonymy and polysemy. In their approach, contexts for a single word are clustered to create several meanings of the word. Similar studies, where the word meaning is disambiguated according to the context, can be found in [Dinu and Lapata, 2010; Erk and Padó, 2010; Huang et al., 2012].

Recently, the neural-network models for learning word representations get attention from many researchers. Artificial neural networks hold an implicit ability to store all seen data (words) in their weights. In theory, it should be enough to infer word meanings. However, many researcher specifically design network architectures to support inference of semantic information. For example, recurrent neural networks can use a memory to see sequences and the context.

In [Mikolov et al., 2013], authors introduce skip-gram and continuous bag-of-words (CBOW) models based on a simple single-layer architecture. They proved that even such a simple neural-network architecture can bring promising results. Huang et al. [2012] introduced a model based on neural network, which uses both local and global context via a joint training objective for modelling semantics of the words. They outperform models that use only local context on the word similarity tasks.

In other papers, words are usually regarded to as an independent entities without any relationship between morphologically related word forms. Luong et al. [2013]

came with an idea to represent words as a composition of morphemes using the recursive neural network (RNN). The word semantics is than learned by neural language models (NLM).

3. State of the art in language modelling

Over the past few years, great attention has been concentrated on the exploration of semantic information in language modelling. Further, discovering latent semantic relations between words is more interesting because there are many methods that work in an unsupervised manner. These methods are usually based on assumptions introduced in the previous section (i.e. the distributional hypothesis and the bag-of-words hypothesis). In the context of this article we distinguish three directions of improvements in language modelling (using global context semantics: Subsection 3.1; using local context semantics: Subsection 3.2; and using morphological information: Subsection 3.3).

3.1. Global semantics language models

The use of global semantics in language modelling is motivated by the assumption that documents (long contexts) may differ in domains, topics, and writing styles. This also means that they have a different probability distribution of words. This assumption is used for adapting language models to the long context (domain, topic, style of particular documents). A method such as LSA, PLSA, or LDA is used to find out which documents (which global contexts) are similar and which are not. This long-context information is added to standard n-gram models to adapt them to the global context. The group of language models that benefits from this idea is sometimes called *topic-based language models*.

An important study on the application of LSA to language modelling was presented in [Bellegarda, 2000]. Significant reductions in perplexity (down to 33%) and improvements in speech recognition of English [word error rate (WER) was decreased by 16%] were achieved in this paper. Several authors later obtained good results with PLSA [Gildea and Hofmann, 1999; Wang et al., 2003] and LDA [Tam and Schultz, 2005, 2006] approaches.

Topic Tracing Language Models (TTLM) are investigated in [Watanabe et al., 2011]. These models are based on LDA and PLSA and integrate the ability to dynamically track changes in topics. The tracking is based upon focused text information and previously estimated topics. This research proves that TTLM significantly improves speech recognition of English.

The language models based on semantic composition are described in [Mitchell and Lapata, 2009]. Word vectors are constructed from LDA (word distribution over topics) or SSM (simple semantic space based on word co-occurrence statistics). Different vector compositions are investigated to represent the history of upcoming words in the language model. Their composition models reduce the perplexity of English corpora when combined with a baseline.

3.2. Local semantics language models

The second direction is to use local context semantics for language modelling, where usually class-based language models or similar architectures are used. Individual words are clustered into much smaller number of classes, which reduces the data sparsity problem that the language models try to tackle.

One of the pilot studies in unsupervised language modelling methods was [Brown et al., 1992], where class-based language models of English were introduced. The clustering algorithm builds clusters in a way that will minimize the entropy of the bigram class-based language model. This problem was reformulated so as to maximize the average mutual information between word clusters in whole training corpora [Maximum Mutual Information (MMI) clustering]. To achieve this, classes keep the words that are most probable in the given context (one word window in both directions), which is essentially similar to the distributional hypothesis on which the methods investigated in this paper are based (words occurring in similar contexts are likely to have similar meanings). The MMI algorithm gives very satisfactory results but its computational complexity is very problematic.

This approach was later extended by [Martin et al., 1998] in order to improve the complexity and to work with trigrams (not only bigrams as in Brown's MMI clustering). This clustering algorithm was also used to create class-based language models of Russian and English in [Whittaker and Woodland, 2003]. Linear interpolation with a baseline model improved the perplexity of Russian by 23% and that of English by 7.9%. The authors also present a 2.2% relative reduction of WER in speech recognition of English.

In [Deschacht et al., 2012], the Latent words language model (LWLM) was introduced. This model uses a very similar idea to the methods in [Brown et al., 1992; Martin et al., 1998], but the solutions differ. LWLM represents the word clusters as a latent variable in a graphical model and these clusters consist of the words that are most probable in the given context window. Gibbs sampling or the Expectation Maximization (EM) algorithm is used for inference. LWLM improved the perplexity of language modelling by 14–18% on three English corpora. LWLM groups words according to their local contexts and thus models the semantic relationships between words. LWLM provides an efficient framework that is able to work with a wider context than the MMI approach and with lower complexity.

In [Brychcín and Konopík, 2014] we were the first to apply semantic spaces (working with a small context window) to language models. Our clustering method based on semantic spaces build clusters of words that are similarly distributed in the corpus. We experimented with modelling of Czech, Slovak, and English and achieved perplexity reductions ranging between 10 and 18%. These language models were also able to significantly improve the BLEU score in machine translation tests.

Neural networks are becoming more attractive for language modelling in recent years [Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010]. They reach the state-of-the-art performance and successfully compete with n-grams. Neural networks can be designed to capture words contextual meaning which enables them to estimate similarity between words. The word sequences that have never been

seen before receive high probability if they are made of words that are semantically similar to words forming an already seen sentences. Given virtually unlimited word combinations such an ability can dramatically increase model performance.

Some researchers also experiment with enrichment of neural networks with an external source of information. For example in [Mikolov and Zweig, 2012], an approximation of LDA topics is fed into the network input alongside with words to add global semantic information.

3.3. Morphology-based language models

A third important direction for improving language modelling is to use morphological information about language. Many authors have already proved that this kind of information can be very useful for the modelling of inflectional languages, which, as stated above, are important for this paper. These approaches usually use supervised methods (lemmatization, part-of-speech tagging, etc.), but unsupervised methods of stemming also exist.

In [Oikonomidis and Digalakis, 2003], the authors used stems for language modelling of Greek. Class-based language models and maximum entropy language models were investigated. The authors present small but significant improvements in the WER of speech recognition.

Another approach to the modelling of Arabic was investigated in [Kirchhoff et al., 2006]. Several different approaches to incorporating morphological information (stems, roots, affixes, morphemes) into language models were tested, with a special focus on factored language models (FLMs) as an architecture. These models successfully improved the performance of speech recognition of Arabic.

Language modelling and speech recognition of Turkish using stems, endings, and morphemes was also investigated in [Arsoy et al., 2006]. The authors present significant improvements in WER by application of their combined model, which uses information about the morphology of Turkish.

In [Oparin, 2008], experiments with morphological random forests (using information about stems, lemmas, parts-of-speech, etc.) in the Czech and Russian languages were shown, with the conclusion that they can be used effectively for inflectional languages.

In [Brychcín and Konopík, 2011] we studied language modelling of Czech and Slovak. We used lemmatization and part-of-speech tagging to derive word clusters for class-based language models and achieved perplexity improvements ranging between 10 and 30% for both languages depending on the amount of training data.

We outlined three main directions (i.e. local semantics, global semantics, and morphology) in language modelling on which researchers often focus their attention. To put our research into the context of the state-of-the-art we can state that our method is based on all these sources of information. As will be shown later, these sources of information enrich each other; moreover, their combination dramatically improves language modelling.

The rest of the article is organized as follows. Section 4 describes how the latent semantics is discovered in unlabelled corpora and how it is incorporated into the

language models. Our experiments are described in Section 5, which is followed by discussion of the results and the conclusion.

4. Our language models

This section describes our language models and how these models are acquired from an unannotated corpus. The baseline is introduced in the following subsection 4.1. The next subsections present various sources of information, such as the morphology (Subsection 4.2), global semantics (Subsection 4.3), and local semantics (Subsection 4.4), which are used to improve language modelling. Finally, Subsection 4.5 describes how these sources of information are combined together with the baseline.

4.1. Baseline

We are using the Modified Kneser-Ney interpolation (introduced in [Chen and Goodman, 1998]), which is the state-of-the-art approach for smoothing methods. The formula for smoothing of word probabilities is

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{cnt\left(w_{i-n+1}^{i}\right) - D\left(cnt\left(w_{i-n+1}^{i}\right)\right)}{\sum_{w_i} cnt\left(w_{i-n+1}^{i}\right)} + \gamma\left(w_{i-n+1}^{i-1}\right) P\left(w_i|w_{i-n+2}^{i-1}\right), \tag{1}$$

where $P()$ is the probability given by the Modified Kneser-Ney interpolation model and $cnt()$ is the count of the n-gram. The goal of discounting function $D(cnt)$ is to save some probability mass for lower-order models. The normalization function $\gamma(w_{i-n+1}^{i}) \in (0, 1)$ makes the probability distribution sum up to 1. The definitions and derivations of these functions can be found in the original paper.

The main advantage of the Modified Kneser-Ney smoothing is the clever way in which it calculates the unigram probability distribution

$$P\left(w_i\right) = \frac{N_{1+}\left(\bullet w_i\right)}{N_{1+}\left(\bullet\bullet\right)}, \tag{2}$$

where symbol $\bullet$ means an arbitrary word (class) and $N_r(w_{i-n+1}^{i})$ is the number of n-grams with frequency r (i.e. the number of such n-grams, where $cnt(w_{i-n+1}^{i}) = r$). In other words, the unigram probability of w_i is given by the number of different bigrams ending in w_i divided by the total number of different bigrams.

4.2. Stem-based language model

We use HPS (see [Brychcín and Konopík, 2015]) as a stemming algorithm. HPS is a new unsupervised stemming algorithm that uses both lexical and semantic information about words to decide how to stem a particular word. The idea of HPS is to split the stemming into two stages. The first stage is based upon a clustering where stemming candidates are selected. The second stage uses a maximum entropy classifier with stemming-specific features that is trained on these candidates.

In our case, stemming is a mapping function $m^S : w \to s$ that maps words $w \in W$ to stems $s \in S$. Note that stemming is contextually independent (in any context the same word always receives the same stem).

We suppose the stemming should be very helpful for languages with rich morphology. We use three ways of incorporating stemming information into the language modelling. The first way is to use class-based language models with stems representing classes, and the other two ways are used to improve global semantic (Subsection 4.3) and local semantic language models (Subsection 4.4).

Class-based language models are the state-of-the-art approaches to language modelling. The main task of the approach is to replace the statistical dependencies between words with dependencies among a much lower number of word classes, thus reducing the data sparsity problem. In our case the class consists of all words with the same stem.

The probability estimation of a word w_i conditioned by its history w_{i-n+1}^{i-1} (where n is the length of the n-gram) is given by the following formula

$$P^S\left(w_i|w_{i-n+1}^{i-1}\right) = P\left(w_i|s_i\right) P\left(s_i|s_{i-n+1}^{i-1}\right). \tag{3}$$

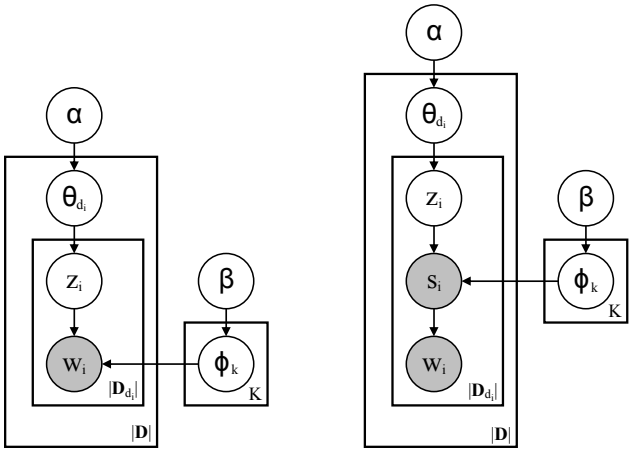
The probability $P\left(s_i|s_{i-n+1}^{i-1}\right)$ is calculated in the same way as in formula 1, but words are replaced with stems. To calculate the probability $P\left(w_i|s_i\right)$, we use Good-Touring smoothing.

4.3. Global semantics language model

For modelling global context properties we use the well-known topic model LDA [Blei et al., 2003] and our extension of LDA enriched with stem information: stem-based LDA (S-LDA).

LDA models documents as a mixture of topics where each topic is simultaneously a mixture of words. Assume we have a set of documents $\boldsymbol{D} = \{\boldsymbol{D}_1, \boldsymbol{D}_2, ..., \boldsymbol{D}_M\}$ each containing a sequence of words. Let w_i denote a word at position i in a corpus, d_i is a document index to which this word belongs and z_i is a hidden topic label for this word that we try to discover. The graphical model representation is depicted in Figure 2a. The generative process of a word corpus in LDA is as follows:

1. For each document $\boldsymbol{D}_m \in \boldsymbol{D}$, sample a distribution $\boldsymbol{\theta}_m \sim \mathit{Dirichlet}\left(\boldsymbol{\alpha}\right)$ over topics $1 \leq z_i \leq K$, where $\boldsymbol{\alpha}$ is a vector of hyper-parameters of Dirichlet distribution.
2. For each topic, sample the word distribution $\boldsymbol{\phi}_k \sim \mathit{Dirichlet}\left(\boldsymbol{\beta}\right)$ over words $1 \leq w_i \leq |W|$, where $\boldsymbol{\beta}$ is a vector of hyper-parameters of Dirichlet distribution.
3. For each position i in a corpus:
   (a) Sample a topic label z_i from the distribution $\boldsymbol{\theta}_{d_i}$.
   (b) Sample the word w_i from the distribution $\boldsymbol{\phi}_{z_i}$.



(a) LDA (b) S-LDA

Figure 2: Graphical model representation of (a) LDA and (b) S-LDA (stem-based LDA). Note that $|\boldsymbol{D}|$ denotes the number of documents and $|\boldsymbol{D}_{d_i}|$ denotes the size of actual document (the number of word tokens).

For inference of topic assignments, we use Gibbs sampling, which needs to compute $P\left(z_i = k|\boldsymbol{z}_{\neg i}, \boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right)$, the probability of topic assignment at position i in the corpus given all other topic assignments for all words. According to [Griffiths and Steyvers, 2004] this leads to a simple formula

$$P\left(z_i = k|\boldsymbol{z}_{\neg i}, \boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right) \propto \frac{cnt_{\neg i,k}^{(w_i)} + \beta_{w_i}}{\sum_{j=1}^{|W|}\left[cnt_{\neg i,k}^{(j)} + \beta_j\right]} \cdot \frac{cnt_{\neg i,k}^{(d_i)} + \alpha_k}{\sum_{l=1}^{K}\left[cnt_{\neg i,l}^{(d_i)} + \alpha_l\right]}, \tag{4}$$

where $cnt_{\neg i,k}^{(w_i)}$ is the number of times the topic k has been assigned to a word w_i, except the position i in the corpus. The $cnt_{\neg i,k}^{(d_i)}$ denotes the count of how many times the topic k occurs in the document d_i again except for the position i in the corpus.

From the topic assignments we can easily obtain estimates of $\boldsymbol{\theta}_m$ and $\boldsymbol{\phi}_k$:

$$P(w_i = j|z_i = k, \boldsymbol{\beta}) = \phi_k^{(j)} \approx \frac{cnt_k^{(j)} + \beta_j}{\sum_{j=1}^{|W|}\left[cnt_k^{(j)} + \beta_j\right]} \tag{5}$$

$$P(z_i = k|d_i, \boldsymbol{\alpha}) = \theta_k^{(d_i)} \approx \frac{cnt_k^{(d_i)} + \alpha_k}{\sum_{l=1}^{K}\left[cnt_l^{(d_i)} + \alpha_l\right]}. \tag{6}$$

Finally, to derive the probability of a word w_i in the context of the whole document (global context) we need to marginalize out the topic variable

$$P^{LDA}(w_i|d_i) = \sum_{k=1}^{K} P(w_i|z_i = k)\, P(z_i|d_i) = \sum_{k=1}^{K} \phi_k^{(w_i)} \theta_k^{(d_i)}. \quad (7)$$

In the second part of this subsection we describe our extension of LDA called S-LDA (shown in Figure 2b). The generative process of S-LDA starts in the same way as the LDA does. Firstly, the topics z_i are generated. For each z_i the stem s_i is sampled from the Dirichlet distribution and finally the word form w_i is selected.

In many languages, especially those with rich morphology, the suffixes contain morpho-syntactic information of the word. In topic models such as LDA based on the bag-of-words approach, the word order has no meaning and the syntactic information is inhibited. We assume the base forms of the words (approximated by stems) contain a satisfactory amount of information to infer topics. Moreover, taking these properties into account, we suppose that this model will deal with the data sparsity problem better.

Because the variables s_i and w_i are both observed in a corpus and $P(w_i|s_i)$ is constant during sampling z_i, the inference process is almost the same as for LDA (in formulas 4 and 5, the variables $\boldsymbol{w}$, w_i, and W are only replaced with $\boldsymbol{s}$, s_i, and S, respectively, where $\boldsymbol{s}$ denote stems on all positions and S is a set of different stems).

Finally, the unigram probability according to S-LDA is given by

$$P^{S-LDA}(w_i|d_i) = P(w_i|s_i) \sum_{k=1}^{K} P(s_i|z_i = k)\, P(z_i|d_i) = P(w_i|s_i) \sum_{k=1}^{K} \phi_k^{(s_i)} \theta_k^{(d_i)}, \quad (8)$$

where $P(w_i|s_i)$ is calculated in the same way as in Subsection 4.2.

4.4. Local semantics language models

According to our previous research [Brychcín and Konopík, 2014], the well-performing semantic spaces in language modelling were discovered to be:

- HAL (Hyperspace Analogue to Language) [Lund and Burgess, 1996]
- COALS (Correlated Occurrence Analogue to Lexical Semantic) [Rohde et al., 2004]
- RI (Random Indexing) [Sahlgren, 2005]

Since words in these semantic spaces are represented as real-valued vectors, we can apply clustering methods. The main assumption is that words within the same cluster should be semantically substitutable (i.e. they make sense at the same position in the appropriate context).

The selection of a suitable clustering algorithm is crucial for such a task. According to the study in [Zhao and Karypis, 2002], we selected the Repeated Bisection algorithm because of its efficiency and acceptable computational requirements. We

use the implementation from the CLUTO software package [Karypis, 2003]. As a measure of the similarity between two words, we use the cosine similarity of word vectors, calculated as the cosine of the angle between corresponding vectors.

Since we already have word clusters, we can easily define the mapping function $m : w \rightarrow c$, where $w \in W$ denotes a word and $c \in C$ denotes a word cluster. Class-based language models seem to be a suitable architecture for applying this kind of information to the language models.

$$P\left(w_i|c_{i-n+1}^{i-1}\right) = P\left(w_i|c_i\right) P\left(c_i|c_{i-n+1}^{i-1}\right). \tag{9}$$

The class (cluster) at position i is given by $c_i = m(w_i)$. The probability $P\left(c_i|c_{i-n+1}^{i-1}\right)$ is calculated in the same way as in formula 1, but words are replaced with word classes. To calculate the probability $P\left(w_i|c_i\right)$ we use Good-Touring smoothing. The mapping function for particular semantic spaces will be denoted as m_{HAL}, m_{COALS}, and m_{RI}.

Similarly to the previous subsection, we also extend these local semantic models with stemming information. During the building of semantic space we can simply use stems instead of word forms and the mapping function becomes $m : w \rightarrow s \rightarrow c$, where $w \in W$, $s \in S$, and $c \in C$ and everything else remain unchanged. The mapping functions using semantic spaces together with stemming will be denoted as m_{S-HAL}, $m_{S-COALS}$, and m_{S-RI}.

4.5. Combination of language models

In this paper we work with two ways of combining language models: linear interpolation (see Subsection 4.5.1) and its extension, bucketed linear interpolation (see Subsection 4.5.2).

4.5.1. Linear interpolation

As in our previous works we use a simple but very effective linear interpolation to combine different language models

$$P^{LI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right), \tag{10}$$

where λ_k is the weight of the k-th language model $P_k()$. We use the *Expectation Maximization (EM)* algorithm described in [Dempster et al., 1977] to calculate optimal weights λ_k, which maximizes the likelihood of the held-out data. Using linear interpolation it is quite straightforward to combine different sources of information such as our global and local context language models.

4.5.2. Bucketed linear interpolation

The linear interpolation can be extended to a method called bucketed linear interpolation, where weights become the function of the frequency of word history [Bahl et al., 1983]. The main idea is that the weights λ_k should be different for words with histories of varying frequencies. For example we expect that the word n-gram

language model would produce the best probability estimates (receive the highest weight) for very often frequented histories. The formula of linear interpolation is transformed to

$$P^{BLI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k\left(w_{i-n+1}^{i-1}\right) \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right). \tag{11}$$

The weights $\lambda_k()$ certainly cannot be different for each possible frequency of history because of data sparsity. Instead, the whole frequency spectrum is divided into buckets, where each bucket holds some range of frequencies. Histories with frequencies in the same bucket receive the same weights. The number of buckets can be tuned but it generally depends on the amount of training data available. The more training data are available, the more buckets can be used.

In our previous research [Brychcín and Konopík, 2014], it was shown that bucketed linear interpolation produces slightly better results compared to simple linear interpolation when combining several language models.

5. Experimental results

In this section we describe various results of our experiments in detail. Firstly, the corpora we use in our experiments are introduced in Subsection 5.1. Perplexity results, as the most commonly used metric for language models, are shown in the next subsection, 5.2. Finally, machine translation tests are shown in Subsection 5.3. We use six languages for our experiments: Czech (CZ), Slovenian (SL), Slovak (SK), Polish (PL), Hungarian (HU), and English (EN).

In all language models, the vocabulary consists of words occurring at least five times in the training data (and is kept fixed for all tests) and n-grams are smoothed by Modified Kneser-Ney interpolation. These language models will be denoted as follows in our experiments:

- Word and stem-based language models:
  - **BL:** Four-gram word-based language model (baseline).
  - **HPS:** Four-gram class-based language model, where classes are stems given by HPS.
- Global semantics language models:
  - **LDA:** Global semantic language models using LDA.
  - **S-LDA:** Global semantic language models using a stemmed version of LDA.
- Local semantics language models:
  - **HAL:** Four-gram class-based language model, where classes are given by clustering HAL.

- **S-HAL:** Four-gram class-based language model, where classes are given by clustering HAL preprocessed by HPS.
- **COALS:** Four-gram class-based language model, where classes are given by clustering COALS.
- **S-COALS:** Four-gram class-based language model, where classes are given by clustering COALS preprocessed by HPS.
- **RI:** Four-gram class-based language model, where classes are given by clustering RI.
- **S-RI:** Four-gram class-based language model, where classes are given by clustering RI preprocessed by HPS.

LDA as well as the semantic spaces (HAL, COALS, and RI as well as their stemmed versions) works with the same vocabulary as the language models do (words occurring at least five times in the training part of the corpus). We use LDA implementation from the MALLET [McCallum, 2002] software package. For each experiment we always train the LDA with 1,000 iterations of Gibbs sampling. The hyperparameters of Dirichlet distributions were initially set to $\alpha = 50/K$, where K is the number of topics and $\beta = 0.1$. This setting is recommended by [Griffiths and Steyvers, 2004]. The number of topics was ranging between 20 and 1,000 to find an optimal configuration for each language (as shown later in experiments, 400 topics LDA performed best among tested languages).

Implementation of the HAL, COALS, and RI algorithms is available in an open source package S-Space [Jurgens and Stevens, 2010]. The parameters of these semantic spaces are set as follows. For each semantic space we use a four-word context window (in both directions). HAL uses a matrix consisting of 50,000 columns, which keeps the largest amount of information. COALS uses a matrix with only 14,000 columns (as recommended by the authors of the algorithm). The SVD reduction was not used in our experiments (according to our previous research [Brychcín and Konopík, 2014], COALS with SVD reduction performed worse). RI uses vectors with a dimension of 1024.

For clustering of words, the Repeated Bisection algorithm with the cosine similarity metric is used. We take the implementation from the CLUTO software package [Karypis, 2003]. For all semantic spaces the word vectors are clustered into four different numbers of clusters: 1,000, 5,000, 10,000, and 20,000. The use of more clusters is meaningless as they will be too sparse and too close to the baseline word model (as is clear from statistics on corpora in Subsection 5.1). From these clusters, appropriate class-based language models are constructed.

For stemming we use our own implementation of HPS available at `http://liks.fav.zcu.cz/HPS`. Already trained stemming models for languages used in this paper are also publicly available there.

We use 50 buckets in linear interpolation for combining our language models.

5.1. Corpora

In our experiments we use the Europarl[2] corpora, version 7, provided by [Koehn, 2005]. These corpora consist of parallel texts in many languages extracted from the proceedings of the European parliament. The texts of each language are aligned with the text in English on sentence level.

We chose to use these corpora for two reasons. Besides the perplexities, we test our language models in a machine translation task (where the parallel corpora are needed to train and evaluate the machine translation system). The second reason is that the Europarl corpora contain texts within the same domain (the same texts on the same topics) in several languages, enabling us to make comparisons of our models among different languages.

Corpora are tokenized with our simple language-independent tokenizer based upon regular expressions. The casing of all word tokens in corpora is normalized using the *true casing* method available within the Moses framework (see Section 5.3). The *true casing* method uses casing statistics collected from a raw text corpus. The statistics contain information about the most frequent casing of all words.

Parallel corpora are used for training machine translation systems (more information is given in Subsection 5.3). Some statistics on these parallel corpora are shown in Table 1.

Table 1: Statistics on parallel corpora after preprocessing and alignment with appropriate English texts for a machine translation. The number of distinct words in corpora is denoted as *words min. 1*. The number of distinct words occurring at least five times in corpora is denoted as *words min. 5*. The full size of the corpora in terms of number of tokens is denoted as *tokens*. The number of the sentences is denoted as *sentences*.

	tokens	sentences	words min. 1	words min. 5
CZ (CZ-EN)	14,971,635	646,605	176,545	63,455
SL (SL-EN)	14,479,624	623,490	145,307	56,352
SK (SK-EN)	14,865,039	640,715	175,676	65,443
PL (PL-EN)	14,685,556	632,565	182,734	67,242
HU (HU-EN)	14,431,768	624,934	327,374	85,802
EN (CZ-EN)	17,430,472	646,605	67,119	27,494

Texts available in the Europarl corpus are not divided into documents that are required for the training of our global context language models based on LDA. There are, however, publicly available source texts of Europarl corpora that are not already mutually aligned on sentence level (monolingual texts). Fortunately, these source texts are annotated with the tag *SPEAKER*, which indicates particular speakers of the texts (the text is about one topic just discussed in parliament). We assume that texts spoken by one speaker at one moment can be taken as documents. To find the boundaries of the documents we trace the points of change of the *SPEAKER* tag. Every change introduces a new document. We use these documents from

[2] Available at http://www.statmt.org/europarl.

these monolingual corpora for training all our language models. Statistics on these monolingual corpora are shown in Table 2. We can see that there are more tokens for all languages, especially for English.

Table 2: Statistics on monolingual corpora after preprocessing. The number of distinct words in corpora is denoted as *words min. 1*. The number of distinct words occurring at least 5 times in corpora is denoted as *words min. 5*. Full size of corpora (the number of tokens) is denoted as *tokens*. The number of distinct documents is denoted as *documents*.

	tokens	documents	words min. 1	words min. 5
CZ	15,100,585	69,927	177,362	63,843
SL	14,547,176	66,600	145,582	56,496
SK	14,967,656	68,669	176,374	65,790
PL	14,873,966	68,444	184,065	67,769
HU	14,548,536	67,160	329,527	86,320
EN	61,260,516	207,946	136,907	49,144

Both parallel and monolingual corpora are split into the training, development (held-out), and test sets in proportions of approximately 70, 10, and 20%, respectively. The held-out set and the test set for both types of corpora are chosen in such a way they contain the same sentences and these sentences are previously unseen for language models as well as for the machine translation model.

5.2. Perplexity results

This subsection presents various experiments with our language models using perplexity as an evaluation measure. Perplexity is the most often used measure of the quality of a language model. The perplexities in the tables below are always calculated on the test part of appropriate corpora (see Section 5.1 about corpora), which is previously unseen for language models.

Baseline perplexities (BL) are shown in all tables in this subsection. Numbers in brackets shown on the right of perplexities of our models denote the relative improvements in perplexity compared to the baseline. We rely on these numbers to be more important than the perplexities themselves. Bold numbers represent the best results for the current language. If it is not written explicitly, then the combination of models is always done by linear interpolation (for more information see Subsection 4.5.1).

Our first way of incorporating morphological information (stemming) into language models is to use stem-based language models (class-based languages model with stems used as classes) and to interpolate them with baseline models. The perplexities are shown in Table 3. We can see that there are significant improvements for all languages except English, even with this simple approach. The stem-based model performs similarly for all Slavic languages, that is, for Czech, Slovenian, Slovak, and Polish. For Hungarian, the representative of Uralic languages, the improvements given by stemming are more significant. The almost complete lack of improvement

for English as a representative of Germanic languages is not surprising, as word normalization (stemming) is meaningless for languages with almost no inflection.

Table 3: Table represents baseline perplexities (denoted as BL) and also perplexities of linear interpolation of baseline with stem-based model (denoted as BL+HPS).

	BL	BL + HPS
CZ	217.0	201.6 (-7.11%)
SL	168.8	157.4 (-6.77%)
SK	200.7	186.3 (-7.17%)
PL	228.0	209.9 (-7.94%)
HU	287.4	252.5 (-12.17%)
EN	65.2	64.3 (-1.42%)

Local semantics language models are investigated in the next group of experiments, where we use semantic spaces (HAL, COALS, and RI) together with their stemmed versions (S-HAL, S-COALS, and S-RI) clustered into four different depths (1,000, 5,000, 10,000, and 20,000 clusters). Class-based language models given by appropriate clusters are always interpolated with a baseline. Results are shown in Table 4. The last column (where a combination of all class-based language models is shown) in the table is especially interesting. We can see that the semantic spaces clustered into different depths enrich each other and are able to improve language models more than interpolations with only a stand-alone class-based language model.

For all languages, the best semantic space for 1,000, 5,000, and 10,000 clusters, respectively, is conclusively HAL without stemming. In the case of 20,000 clusters, S-HAL performed better for all languages except Hungarian (HAL performed best) and English (the RI model performed best, which we suppose is due to chance rather than its properties). The stemmed version of semantic spaces worked well only in the case of COALS, where S-COALS was almost always better, and also for sparse clusters (20,000 clusters) where the stemmed model was almost always better than the unstemmed one. From these results we can state that the HAL model is most suitable for language modelling of all languages. Perplexity improvements by HAL (baseline interpolated with all four class-based language models created from HAL) are 13% on average for inflectional languages. For English, the improvement is not as big (approximately 7%).

Global semantic language models are shown in Table 5. The word unigram probability given by the LDA and S-LDA models with the number of topics ranging between 20 and 1,000 is interpolated with the baseline model. We can see similar improvements compared to baseline for all six languages including English. In the case of LDA the best results are achieved with 300 topics on average. LDA can be trained well on English corpora even for higher a number of topics (because of lower significance of data sparsity). We can see that S-LDA models always performed better than unstemmed versions. Moreover, thanks to stemming, it is possible to

Table 4: Perplexities of local semantics language models. Columns, denoted as 1k, 5k, 10k, 20k, respectively, represent linear interpolation of baseline with the class-based language model given by a particular semantic space clustered into corresponding number of clusters. Last column, denoted as BL+{1k–20k}, represents the linear interpolation of baseline with four class-based language models created from all clusters (1k, 5k, 10k, 20k) of appropriate semantic space. Row denoted as BL represents the perplexity of baseline.

(a) CZ					
BL	217.0				
	Number of classes				BL+{1k–20k}
	1k	5k	10k	20k	
HAL	**199.8 (-7.97%)**	**199.6 (-8.05%)**	**200.4 (-7.67%)**	203.2 (-6.39%)	**191.0 (-12.00%)**
S-HAL	203.9 (-6.05%)	201.9 (-6.99%)	200.7 (-7.53%)	**200.1 (-7.79%)**	195.4 (-9.97%)
COALS	206.9 (-4.66%)	206.6 (-4.83%)	203.5 (-6.26%)	202.6 (-6.65%)	198.7 (-8.47%)
S-COALS	208.3 (-4.05%)	205.0 (-5.54%)	201.4 (-7.21%)	201.2 (-7.30%)	196.6 (-9.41%)
RI	204.9 (-5.61%)	203.5 (-6.25%)	203.4 (-6.28%)	205.5 (-5.33%)	197.1 (-9.21%)
S-RI	206.1 (-5.02%)	204.5 (-5.80%)	203.3 (-6.36%)	202.5 (-6.69%)	198.7 (-8.47%)
(b) SL					
BL	168.8				
	Number of classes				BL+{1k–20k}
	1k	5k	10k	20k	
HAL	**153.7 (-8.99%)**	**154.4 (-8.56%)**	**154.6 (-8.40%)**	157.0 (-6.98%)	**146.7 (-13.09%)**
S-HAL	156.5 (-7.32%)	155.4 (-7.93%)	154.8 (-8.30%)	**154.3 (-8.63%)**	149.7 (-11.33%)
COALS	159.9 (-5.29%)	160.8 (-4.78%)	159.0 (-5.85%)	156.7 (-7.17%)	154.3 (-8.59%)
S-COALS	160.8 (-4.74%)	159.1 (-5.77%)	157.2 (-6.87%)	155.7 (-7.77%)	152.2 (-9.87%)
RI	158.4 (-6.20%)	157.5 (-6.70%)	157.3 (-6.80%)	159.1 (-5.78%)	152.0 (-9.96%)
S-RI	158.8 (-5.97%)	157.4 (-6.74%)	156.9 (-7.09%)	156.5 (-7.29%)	152.7 (-9.55%)
(c) SK					
BL	200.7				
	Number of classes				BL+{1k–20k}
	1k	5k	10k	20k	
HAL	**182.4 (-9.10%)**	**182.7 (-8.96%)**	**183.5 (-8.56%)**	186.8 (-6.90%)	**174.0 (-13.27%)**
S-HAL	186.5 (-7.08%)	184.7 (-7.94%)	183.8 (-8.42%)	**183.3 (-8.66%)**	178.4 (-11.10%)
COALS	189.3 (-5.66%)	190.1 (-5.27%)	188.1 (-6.26%)	186.6 (-7.03%)	182.1 (-9.25%)
S-COALS	191.8 (-4.45%)	188.8 (-5.93%)	185.9 (-7.34%)	185.8 (-7.42%)	180.9 (-9.83%)
RI	188.6 (-6.04%)	187.2 (-6.73%)	187.0 (-6.82%)	189.7 (-5.46%)	181.0 (-9.79%)
S-RI	189.0 (-5.82%)	187.3 (-6.69%)	186.3 (-7.16%)	185.7 (-7.47%)	181.8 (-9.39%)
(d) PL					
BL	228.0				
	Number of classes				BL+{1k–20k}
	1k	5k	10k	20k	
HAL	**206.6 (-9.39%)**	**206.3 (-9.54%)**	**206.9 (-9.25%)**	210.5 (-7.68%)	**195.7 (-14.18%)**
S-HAL	213.3 (-6.46%)	210.3 (-7.78%)	209.0 (-8.33%)	**208.6 (-8.51%)**	203.3 (-10.86%)
COALS	214.5 (-5.92%)	214.2 (-6.07%)	211.7 (-7.14%)	210.8 (-7.54%)	204.0 (-10.53%)
S-COALS	217.4 (-4.65%)	213.3 (-6.48%)	210.3 (-7.77%)	209.6 (-8.06%)	204.3 (-10.40%)
RI	214.0 (-6.14%)	212.2 (-6.95%)	212.1 (-6.99%)	214.5 (-5.95%)	204.5 (-10.30%)
S-RI	215.8 (-5.35%)	213.7 (-6.29%)	212.6 (-6.76%)	211.4 (-7.28%)	207.5 (-9.01%)
(e) HU					
BL	287.4				
	Number of classes				BL+{1k–20k}
	1k	5k	10k	20k	
HAL	**263.8 (-8.24%)**	**261.6 (-8.98%)**	**262.5 (-8.67%)**	**268.3 (-6.65%)**	**251.4 (-12.54%)**
S-HAL	277.3 (-3.53%)	275.4 (-4.20%)	274.8 (-4.38%)	274.6 (-4.45%)	272.6 (-5.18%)
COALS	273.5 (-4.85%)	271.4 (-5.58%)	271.0 (-5.73%)	274.7 (-4.43%)	262.0 (-8.85%)
S-COALS	280.3 (-2.48%)	277.6 (-3.43%)	276.8 (-3.72%)	275.1 (-4.30%)	273.0 (-5.03%)
RI	273.6 (-4.80%)	270.8 (-5.78%)	270.2 (-6.01%)	274.3 (-4.56%)	263.4 (-8.38%)
S-RI	278.6 (-3.08%)	277.0 (-3.61%)	276.3 (-3.86%)	275.9 (-4.02%)	274.1 (-4.64%)
(f) EN					
BL	65.2				
	Number of classes				BL+{1k–20k}
	1k	5k	10k	20k	
HAL	**61.9 (-5.16%)**	**62.4 (-4.35%)**	**62.6 (-3.99%)**	63.42 (-2.80%)	**60.6 (-7.11%)**
S-HAL	62.8 (-3.74%)	63.1 (-3.29%)	63.3 (-3.03%)	63.5 (-2.62%)	62.4 (-4.37%)
COALS	62.8 (-3.68%)	63.3 (-3.04%)	63.5 (-2.73%)	64.0 (-1.93%)	61.7 (-5.45%)
S-COALS	63.4 (-2.87%)	63.7 (-2.38%)	63.8 (-2.27%)	64.0 (-1.92%)	62.8 (-3.72%)
RI	62.7 (-3.83%)	63.0 (-3.52%)	63.1 (-3.28%)	**63.4 (-2.82%)**	61.7 (-5.44%)
S-RI	63.1 (-3.31%)	63.3 (-3.01%)	63.4 (-2.85%)	63.6 (-2.52%)	62.6 (-3.99%)

infer the more distinct latent topics (as was expected mainly for inflectional lan-

guages). S-LDA performs almost 3% better than the LDA model in the modelling of inflectional languages. For English, both models are similar. By comparison with local semantics, the global semantics improves the language modelling slightly more (up to 16%).

Table 5: Table displays the perplexities of the LDA and S-LDA language model (denoted as BL+LDA and BL+S-LDA, respectively) when combined with baseline according to different number of latent topics. Row denoted as BL represents perplexities of baseline.

	CZ	SL	SK	PL	HU	EN
BL	217.0	168.8	200.7	228.0	287.4	65.2
topics	BL+LDA					
20	198.5 (-8.53%)	155.1 (-8.12%)	184.1 (-8.27%)	209.4 (-8.17%)	264.0 (-8.14%)	60.5 (-7.30%)
50	194.3 (-10.47%)	151.8 (-10.07%)	180.4 (-10.13%)	205.2 (-10.00%)	259.3 (-9.79%)	59.5 (-8.76%)
100	190.3 (-12.32%)	149.0 (-11.74%)	176.6 (-11.99%)	201.4 (-11.67%)	255.3 (-11.19%)	58.7 (-9.97%)
200	188.3 (-13.24%)	**146.6 (-13.16%)**	174.6 (-12.99%)	**199.0 (-12.71%)**	255.3 (-11.19%)	57.8 (-11.42%)
300	**187.9 (-13.43%)**	146.8 (-13.02%)	**174.6 (-13.00%)**	199.4 (-12.57%)	255.6 (-11.08%)	57.3 (-12.20%)
400	188.1 (-13.33%)	146.8 (-13.02%)	174.6 (-12.99%)	199.8 (-12.38%)	**254.5 (-11.47%)**	56.9 (-12.81%)
500	188.4 (-13.19%)	147.1 (-12.88%)	174.8 (-12.88%)	199.8 (-12.38%)	254.6 (-11.43%)	56.5 (-13.35%)
600	188.3 (-13.25%)	147.1 (-12.85%)	175.2 (-12.68%)	199.6 (-12.47%)	255.1 (-11.23%)	56.4 (-13.61%)
700	188.8 (-13.00%)	147.2 (-12.81%)	175.3 (-12.66%)	200.0 (-12.28%)	255.2 (-11.20%)	**56.3 (-13.66%)**
800	188.5 (-13.14%)	147.4 (-12.67%)	175.3 (-12.64%)	199.7 (-12.44%)	255.1 (-11.26%)	56.4 (-13.61%)
900	188.8 (-13.02%)	147.5 (-12.66%)	175.6 (-12.52%)	200.5 (-12.08%)	255.2 (-11.23%)	56.4 (-13.52%)
1000	189.2 (-12.82%)	147.8 (-12.47%)	175.8 (-12.40%)	199.9 (-12.31%)	255.2 (-11.23%)	56.4 (-13.50%)
topics	BL+S-LDA					
20	198.7 (-8.43%)	154.7 (-8.35%)	184.3 (-8.15%)	209.3 (-8.22%)	263.3 (-8.41%)	60.5 (-7.26%)
50	194.4 (-10.45%)	151.8 (-10.08%)	180.4 (-10.13%)	205.3 (-9.97%)	258.0 (-10.24%)	59.7 (-8.50%)
100	190.2 (-12.35%)	148.8 (-11.85%)	176.8 (-11.91%)	201.6 (-11.59%)	252.5 (-12.17%)	58.9 (-9.74%)
200	185.8 (-14.39%)	146.1 (-13.48%)	172.6 (-13.98%)	197.4 (-13.45%)	247.2 (-13.99%)	58.0 (-11.03%)
300	183.1 (-15.66%)	144.1 (-14.65%)	170.3 (-15.11%)	194.6 (-14.67%)	**245.5 (-14.58%)**	57.5 (-11.85%)
400	**182.0 (-16.15%)**	142.7 (-15.47%)	**168.7 (-15.94%)**	**193.1 (-15.30%)**	246.1 (-14.39%)	57.2 (-12.41%)
500	182.5 (-15.93%)	**141.9 (-15.95%)**	168.9 (-15.82%)	193.2 (-15.28%)	247.1 (-14.04%)	56.8 (-12.95%)
600	182.8 (-15.79%)	142.4 (-15.68%)	169.3 (-15.62%)	193.7 (-15.04%)	246.8 (-14.12%)	56.6 (-13.23%)
700	183.4 (-15.49%)	142.7 (-15.48%)	169.8 (-15.41%)	194.2 (-14.84%)	247.5 (-13.89%)	56.4 (-13.57%)
800	183.6 (-15.42%)	142.9 (-15.34%)	170.1 (-15.22%)	194.6 (-14.64%)	247.6 (-13.87%)	56.3 (-13.76%)
900	184.0 (-15.21%)	143.3 (-15.14%)	170.2 (-15.20%)	194.7 (-14.59%)	247.0 (-14.05%)	56.2 (-13.91%)
1000	184.1 (-15.19%)	143.4 (-15.04%)	170.4 (-15.10%)	195.3 (-14.33%)	247.9 (-13.76%)	**56.2 (-13.93%)**

The most important experiments are shown in Table 6, where a combination of different sources of information is depicted. As the best-performing model for local semantics, the HAL model was chosen (a combination of HAL-based language models of 1,000, 5,000, 10,000, and 20,000 clusters). Global semantics in language models was best modelled by S-LDA with 400 topics. The stem-based model (HPS) is also added together with the baseline (BL) and the final model is thus a combination of all seven language models (the last two columns in the table, where we finally compare linear interpolation with a bucketed linear interpolation).

From the table we can clearly state that all three sources of information (morphology, local semantics, and global semantics) significantly enrich each other (which is especially evident for inflectional languages). Their bucketed linear combination leads to improvements of up to almost 26% for inflectional languages and up to 15% for English compared to the stand-alone baseline. Our language models perform similarly for all inflectional languages (improvements by each part are quite similar). We can also see that the bucketed linear interpolation (BLI-all) produces slightly better results than simple linear interpolation (LI-all).

Table 6: Perplexity results by combining different sources of information. The operator + denotes the linear interpolation of appropriate models. The S-LDA model uses 400 topics, and HAL denotes all four class-based language models based on HAL (1k, 5k, 10k, and 20k clusters). The last two columns are the most important, where all models (BL+HPS+HAL+S-LDA) are combined by linear interpolation (LI-all) and bucketed linear interpolation (BLI-all).

	BL	BL + HPS	BL+HAL	BL+HAL+HPS	BL+S-LDA	LI-all	BLI-all
CZ	217.0	201.6 (-7.11%)	191.0 (-12.00%)	183.3 (-15.53%)	182.0 (-16.15%)	167.1 (-23.03%)	**165.4 (-23.79%)**
SL	168.8	157.4 (-6.77%)	146.7 (-13.09%)	141.6 (-16.11%)	142.7 (-15.47%)	130.7 (-22.56%)	**128.1 (-24.11%)**
SK	200.7	186.3 (-7.17%)	174.0 (-13.27%)	167.3 (-16.63%)	168.7 (-15.94%)	153.5 (-23.51%)	**151.0 (-24.75%)**
PL	228.0	209.9 (-7.94%)	195.7 (-14.18%)	187.4 (-17.81%)	193.1 (-15.30%)	173.7 (-23.82%)	**171.1 (-24.97%)**
HU	287.4	252.5 (-12.17%)	251.4 (-12.54%)	233.6 (-18.74%)	246.1 (-14.39%)	216.2 (-24.79%)	**213.4 (-25.77%)**
EN	65.2	64.3 (-1.42%)	60.6 (-7.11%)	60.3 (-7.62%)	57.2 (-12.41%)	55.7 (-14.60%)	**55.4 (-15.10%)**

In order to visualize what weights in the linear interpolation (LI-all model) are allocated for each sub-model by the EM algorithm, we render Figure 3, where our final model created from seven sub-models is shown. We can see that the baseline always has the highest weight, and then, in order of decreasing weight, S-LDA, HPS, HAL{20k}, HAL{10k}, HAL{5k} and HAL{1k} follow, where the numbers in brackets mean the numbers of clusters. A direct correlation between weights and perplexity improvements can be seen. The bigger the weights allocated by the EM algorithm, the greater the improvement in perplexity achieved.

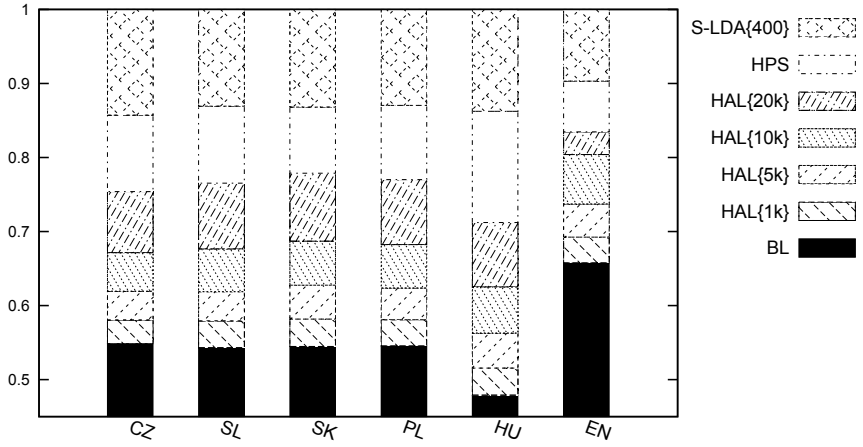


Figure 3: Interpolation weights of each sub-model in the final linear interpolation.

5.3. Machine translation results

This section describes the performance of the proposed language models in terms of a machine translation task. Success in this task should verify the ability of the models to improve the performance of a real-world application. The system used in this test is based upon the statistical machine translation toolkit called Moses[3], briefly described in [Koehn et al., 2007].

[3]Available at http://www.statmt.org/moses/.

Europarl parallel corpora were used for training and evaluation of machine translation (see Subsection 5.1). Table 7 shows the settings for translation model training.

Table 7: Parameters used for training the machine translation system.

Casing normalization	True casing
Minimum-maximum tokes per sentence	1-80
Language model order	4 (binary)
Language model smoothing	Modified Kneser-Ney
Language model toolkit	IRSTLM
Alignment heuristic	grow-diag-final-and
Reordering model	msd-bidirectional-fe
Tuning	MERT

Our translation experiment consists in measuring the difference in translation performance when the standard four-gram language model is used and when our improved language model is employed. Language models are used during training as well as during decoding in Moses. The best approach would be to replace the standard model with our model. However, our model does not support left-to-right decoding because global semantic models require knowledge of word contexts. This prevents the use of our model for training as well as directly for decoding. Instead, we generate 5,000 best hypotheses and re-score them with our model. Such a procedure is not as effective as direct incorporation of the model into Moses; however, it is the only possible approach.

The change of the language model in Moses is not possible without re-computing the model weights. Moses uses weights for translation, reordering, word penalty, and language models. The weights are set during the optimization phase by the MERT (Minimum Error Rate Training) algorithm [Och, 2003]. The algorithm optimizes the weights for best translation scores on the held-out data. However, such weights are valid only for the original model. A different model returns different probability estimates and thus they play different roles during interpolation of the final translation probability.

We estimate the correct weight for our improved language model with the same algorithm, MERT. We use the n-best list generated during the optimization phase. We replace the probability estimates of the original language model with the estimates from our model. Then, we run the optimization procedure, which returns news weights including the weight for our improved language model. These new weights are used for translating test sentences.

The results of our translation experiment are shown in Table 8. We measure the translation scores with the BLEU metric [Papineni et al., 2002]. This metric is based on the ratio of n-gram overlaps with reference translations. We compare original translations from Moses with translations obtained by re-scoring them with our language model. We generate and re-score 5000 hypotheses for each translated sentence. The most probable hypothesis is taken as the translation result. Beside

showing scores for the whole language model composed of all sub-models (local, global semantics, and stemming), we also study the performance of language models composed of various combinations of sub-models. Numbers in brackets denote the absolute improvements in BLEU score.

Table 8: BLEU scores achieved by combination of different language models. Arrows show the direction of translation. BL denotes the baseline language model (the standard output from the Moses). HPS is a stem-based language model. HAL denotes the use of all four HAL-based language models (1k, 5k, 10k, and 20k clusters). S-LDA denotes language model based on stemmed version of LDA using 400 topics. Linear interpolation of all seven sub-models is denotes as LI-all.

	BL	BL + HPS	BL+HAL	BL+HAL+HPS	BL+S-LDA	LI-all
EN → CZ	26.02	26.24 (+0.22)	26.64 (+0.62)	26.75 (+0.73)	26.34 (+0.32)	**26.93 (+0.91)**
EN → SL	28.58	28.77 (+0.19)	29.07 (+0.49)	29.17 (+0.59)	28.75 (+0.17)	**29.32 (+0.74)**
EN → SK	27.04	27.20 (+0.16)	27.60 (+0.56)	27.69 (+0.65)	27.26 (+0.22)	**27.79 (+0.75)**
EN → PL	23.70	23.86 (+0.16)	24.13 (+0.43)	24.27 (+0.57)	23.95 (+0.25)	**24.38 (+0.68)**
EN → HU	18.51	19.00 (+0.49)	19.19 (+0.68)	19.39 (+0.88)	19.02 (+0.51)	**19.47 (+0.96)**
CZ → EN	37.29	37.42 (+0.13)	37.70 (+0.41)	37.73 (+0.44)	37.75 (+0.46)	**37.97 (+0.68)**

We can see that each of the three sources of information (morphology, local semantics, and global semantics) gives at least some improvement. Their combination again enriches each of them even in machine translation tests. The highest improvements are always achieved by HAL (combination of 1k, 5k, 10k, and 20k clusters), which is the best representative of local semantics models. HPS and S-LDA provide similar improvements. Stemming is again useless for English. The fact that S-LDA performs worse than HAL is probably caused by working on the sentence level (not document level), on which Moses is focused. If the S-LDA model had wider context (i.e. the whole document) it would probably perform better as was shown in Subsection 5.2. The combination of all sources of information (the last column in the table) studied in this paper leads to signification improvements in BLEU score, which is a solid proof that the proposed language models are usable and effective in a real-world application.

6. Discussion

In this section we summarize our results and discuss the behaviour of our methods. In previous sections we presented various experiments on our language models. We experimented with a combination of three different sources of information (morphology, local semantics, and global semantics). Firstly, we tested language models from the theory of information point of view, where the perplexity was used as an evaluation measure. The results of these tests were followed by evaluation in a real-world application, where machine translation with the Moses system proved the quality of our methods.

First, let us look at the perplexity results. Stem-based language models (HPS) proved to be efficient for inflectional languages. The average relative improvement in perplexity for Slavic languages (Czech, Slovenian, Slovak, and Polish) was about

7%; for Hungarian it was as much as 12%. As we expected, stemming was found to be useless for English, with almost no improvement in perplexity.

Semantic spaces (HAL, COALS, and RI) and their stemmed versions (S-HAL, S-COALS, and S-RI) were explored as a next step, where we created class-based language models according to different numbers of clusters (i.e. 1,000, 5,000, 10,000, and 20,000). The HAL model performed best in the majority of experiments. It was found to be better not to use stemming as the results of stemmed versions of semantic spaces were almost always worse than those of unstemmed models. Stemming was often better only in cases where sparse clusters (20,000) were used. We suppose this is caused by the nature of local semantic models, which thanks to the short context, also incorporate morpho-syntactic information of words that is very useful especially in language models. In contrast, stemming inhibits this kind of information. The combination of class-based language models of different numbers of clusters was shown to improve language modelling by approximately 13% on average for inflectional languages and by 7% for English.

In our previous research [Brychcín and Konopík, 2014], HAL was also best for 1,000, 5,000, or 10,000 clusters, but the COALS model performed better than HAL in the case of 20,000 or more clusters. Our current results confirm that for inflectional languages the difference between HAL and COALS is lower with a growing number of clusters; moreover in the case of 20,000 clusters, the performance of COALS was slightly better. In the cases of Hungarian and English, HAL performed best all the time.

The third source of information was global semantics modelled by LDA and S-LDA (the stemmed version of LDA). LDA has already been proven by many researchers to be useful in language modelling (see, e.g., [Tam and Schultz, 2005, 2006; Watanabe et al., 2011]) and our results agree with that. LDA improved the perplexity of models of all languages by about 13% on average. Moreover, our stemmed extension of LDA was able to achieve even better results for inflectional languages (improvements of up to 16%). Here, the situation is opposite to that of local semantics. LDA is based on bag-of-word hypotheses, where the word order is inhibited (the morpho-syntactic information is useless for topic inference). Stemming thus helps to deal with the data sparsity problem better.

The most important finding, which is also the aim of this paper, is that the investigated sources of information mutually help each other in terms of improving language modelling. Their combination was able to dramatically reduce the perplexities of language models. By grouping HPS, HAL, and S-LDA with a baseline we achieved approximately 25% improvement on average for all inflectional languages, which is a very satisfactory result. The improvement for English was not as big (approximately 15%). Our explanation lies in the morphological complexity of languages. As expected, the word normalization was negligible for English. Also, the semantic spaces working with short context incorporate morpho-syntactic information of words. We believe that this is the main reason why they are more suitable for inflectional languages. Thus, the language modelling of English profits mainly from LDA model.

The same model combinations were also investigated in a machine translation task, where we measured BLEU scores. Only by changing the language model was significant growth of the BLEU score achieved (the average improvement among all languages was 0.8 BLEU points). If we compare our results with different works on the same corpora (e.g. works of [Virpioja et al., 2007; Sanchis-trilles et al., 2010]), we can claim that our models are very effective.

The results from machine translation correlate with the perplexity results but there are some deviations in improvements of inflectional languages even though the improvements in perplexities were almost identical. The performance of the machine translation system depends on several modules (not only on the language model) as well as on the optimization of parameters on held-out data (using MERT). The difficulty of the language must also be taken into account, as we observe proportionally different perplexities and different BLEU scores among all languages despite using the corpora within the same domain.

The fact that we significantly improved the modelling of several languages with different properties (different families of languages) testifies to the quality of our methods. Moreover, these methods are attractive due to their unsupervised nature and so they can be easily applied to other languages or tasks.

7. Summary

7.1. Future work

As shown in Section 2, there is a huge number of methods for latent semantics discovery that are worth investigating. These methods use various architectures, e.g. matrix factorization methods, graphical models, or neural networks. It is beyond the scope of this paper to compare them all. This is the main direction for the future work as we believe that other combinations may produce even better language models.

Another alternative for future work consists in studying machine translation more deeply. The experiments with the Moses machine translation framework revealed that it is an extraordinary framework with great extension capabilities. We expect that through the direct link between Moses and the methods investigated in this paper (local and global semantics or stemming), even higher improvements could be achieved. Moses supports several architectures for applying various sources of information. For example clusters given by semantic spaces, stems, or topics can be directly used in factored translation.

In addition we also want to test our methods in different NLP tasks (e.g. speech recognition, optical character recognition, spelling correction, etc.).

7.2. Conclusion

In this article we experimented with three kinds of various information sources whose application into language modelling yields significant improvements in model

prediction ability. The beauty of our method is that all the information comes directly from the data. Nothing is added externally. The information we use is sometimes called *latent* or *hidden* because an algorithm must be employed to discover it. We use tree approaches for the discovery of hidden information. Topic models discover long semantic relations between words and the documents where they are used. Semantic spaces (HAL, COALS, RI) work with short semantic relations between words and their direct contexts. The stemming studies the information hidden directly in different forms of words.

All these sources were previously used in a research (semantic spaces in our preceding work). However, nobody had studied whether their combination carries some extra information. We conclusively proved that the combination provides much higher perplexity reduction than individual models do.

Our stemming algorithm (HPS) proved to be very useful in language modelling of inflectional languages. The use of stems as classes for class-based language models or extending LDA with stem information was shown to be very effective, as we achieved significant improvements in language modelling. Taking into account the results and our findings during testing, we can recommend HAL for modelling local semantics and S-LDA for modelling global semantics.

We believe that our article carries a potential beyond language models. All the methods investigated in this paper are based upon unsupervised training. We successfully used some of these methods in different NLP applications such as sentiment analysis [Habernal and Brychcín, 2013; Brychcín et al., 2014], document classification [Brychcín and Král, 2014], and named entity recognition [Konkol et al., 2014].

Acknowledgement

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.

References

Arsoy, E., Dutaac, H., Arslan, L. M., 2006. A unified language model for large vocabulary continuous speech recognition of turkish. Signal Processing 86 (10), 2844 – 2862.

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

Bellegarda, J. R., Aug. 2000. Exploiting latent semantic information in statistical language modeling. Proceedings of the IEEE 88 (8), 1279 –1296.

Bengio, Y., Ducharme, R., Vincent, P., Janvin, C., Mar. 2003. A neural probabilistic language model. J. Mach. Learn. Res. 3, 1137–1155.
URL http://dl.acm.org/citation.cfm?id=944919.944966

Blei, D. M., Ng, A. Y., Jordan, M. I., Lafferty, J., 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3, 2003.

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., Konkol, M., Steinberger, J., 2014. Uwb: Machine learning approach to aspect-based sentiment analysis. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). Association for Computational Linguistics, pp. 817–822.
URL http://aclweb.org/anthology/S14-2145

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.

Brychcín, T., Konopík, M., 2015. Hps: High precision stemmer. Information Processing & Management 51 (1), 68 – 91.
URL http://www.sciencedirect.com/science/article/pii/S0306457314000843

Brychcín, T., Král, P., 2014. Novel unsupervised features for czech multi-label document classification. In: Gelbukh, A., Espinoza, F., Galicia-Haro, S. (Eds.), Human-Inspired Computing and Its Applications. Vol. 8856 of Lecture Notes in Computer Science. Springer International Publishing, pp. 70–79.
URL http://dx.doi.org/10.1007/978-3-319-13647-9_8

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.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407.

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.

Deschacht, K., Belder, J. D., Moens, M.-F., 2012. The latent words language model. Computer Speech & Language 26 (5), 384 – 409.

Dinu, G., Lapata, M., 2010. Measuring distributional similarity in context. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1162–1172.
URL http://dl.acm.org/citation.cfm?id=1870658.1870771

Erk, K., Padó, S., 2010. Exemplar-based models for word meaning in context. In: Proceedings of the ACL 2010 Conference Short Papers. ACLShort '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 92–97.
URL http://dl.acm.org/citation.cfm?id=1858842.1858859

Firth, J. R., 1957. A Synopsis of Linguistic Theory, 1930-1955. Studies in Linguistic Analysis, 1–32.

Gildea, D., Hofmann, T., 1999. Topic-based language models using em. In: Proceedings of Eurospeech. pp. 2167–2170.

Griffiths, T. L., Steyvers, M., Apr. 2004. Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America 101 (Suppl 1), 5228–5235.

Habernal, I., Brychcín, T., 2013. Semantic spaces for sentiment analysis. In: Proceedings of the 16th international conference on Text, Speech and Dialogue (TSD13).

Hofmann, T., 1999. Probabilistic latent semantic analysis. In: Proceedings of 15th Conference on Uncertainty in Artificial Intelligence. pp. 289–296.

Huang, E. H., Socher, R., Manning, C. D., Ng, A. Y., 2012. Improving word representations via global context and multiple word prototypes. In: Proceedings of the 50th Annual Meeting of the

Association for Computational Linguistics: Long Papers - Volume 1. ACL '12. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 873–882.
URL http://dl.acm.org/citation.cfm?id=2390524.2390645

Jones, M. N., Mewhort, D. J. K., 2007. Representing word meaning and order information in a composite holographic lexicon. Psychological Review 114, 1–37.

Jurgens, D., Stevens, K., 2010. The s-space package: An open source package for word space models. In System Papers of the Association of Computational Linguistics.

Karypis, G., 2003. Cluto - a clustering toolkit.
URL www.cs.umn.edu/~karypis/cluto

Kirchhoff, K., Vergyri, D., Bilmes, J., Duh, K., Stolcke, A., Oct. 2006. Morphology-based language modeling for conversational Arabic speech recognition. Computer Speech & Language 20 (4), 589–608.

Koehn, P., 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In: Machine Translation Summit X. Phuket, Thailand, pp. 79–86.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E., 2007. Moses: open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. ACL '07. Association for Computational Linguistics, pp. 177–180.

Konkol, M., Brychcín, T., Konopík, M., 2014. Latent semantics in named entity recognition. Expert Systems with Applications, in press.
URL http://www.sciencedirect.com/science/article/pii/S0957417414007933

Lund, K., Burgess, C., 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods Instruments and Computers 28 (2), 203–208.

Luong, T., Socher, R., Manning, C., August 2013. Better word representations with recursive neural networks for morphology. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning. Association for Computational Linguistics, Sofia, Bulgaria, pp. 104–113.
URL http://www.aclweb.org/anthology/W13-3512

Martin, S., Liermann, J., Ney, H., 1998. Algorithms for bigram and trigram word clustering. Speech Communication 24 (1), 19 – 37.

McCallum, A. K., 2002. Mallet: A machine learning for language toolkit.
URL http://mallet.cs.umass.edu

McNamara, D. S., 2011. Computational methods to extract meaning from text and advance theories of human cognition. Topics in Cognitive Science 3 (1), 3–17.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. CoRR abs/1301.3781.
URL http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781

Mikolov, T., Karafit, M., Burget, L., ernock, J., Khudanpur, S., 2010. Recurrent neural network based language model. In: Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010). Vol. 2010. International Speech Communication Association, pp. 1045–1048.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=9362

Mikolov, T., Zweig, G., 2012. Context dependent recurrent neural network language model. In: 2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012. pp. 234–239.
URL http://dx.doi.org/10.1109/SLT.2012.6424228

Mitchell, J., Lapata, M., 2009. Language models based on semantic composition. In: In Proceedings of EMNLP. pp. 430–439.

Och, F. J., 2003. Minimum error rate training in statistical machine translation. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1. ACL '03. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 160–167.

Oikonomidis, D., Digalakis, V., 2003. Stem-based maximum entropy language models for inflectional languages. In: INTERSPEECH. ISCA.

Oparin, I., 2008. Language models for automatic speech recognition of inflectional languages. Ph.D. thesis, University of West Bohemia, Pilsen.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. ACL '02. Association for Computational Linguistics, pp. 311–318.

Purandare, A., Pedersen, T., 2004. Word sense discrimination by clustering contexts in vector and similarity spaces. Proceedings of 8th Conference on Computational Natural Language Learning, 41–48.

Reisinger, J., Mooney, R., 2010a. A mixture model with sharing for lexical semantics. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. EMNLP '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1173–1182.
URL http://dl.acm.org/citation.cfm?id=1870658.1870772

Reisinger, J., Mooney, R. J., 2010b. Multi-prototype vector-space models of word meaning. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. HLT '10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 109–117.
URL http://dl.acm.org/citation.cfm?id=1857999.1858012

Riordan, B., Jones, M. N., 2011. Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation. Topics in Cognitive Science 3 (2), 303–345.

Rohde, D. L. T., Gonnerman, L. M., Plaut, D. C., 2004. An improved method for deriving word meaning from lexical co-occurrence. Cognitive Psychology 7, 573–605.

Sahlgren, M., 2005. An Introduction to Random Indexing. Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005.

Sahlgren, M., Holst, A., Kanerva, P., 2008. Permutations as a means to encode order in word space. Proceedings of the 30th Annual Conference of the Cognitive Science Society, 1300–1305.

Sanchis-trilles, G., Cettolo, M., Kessler, F. B., 2010. Online language model adaptation via n-gram mixtures for statistical machine translation. In: Proceedings of the 14th Annual Conference of the European Association for Machine Translation.

Schwenk, H., Jul. 2007. Continuous space language models. Comput. Speech Lang. 21 (3), 492–518.
URL http://dx.doi.org/10.1016/j.csl.2006.09.003

Tam, Y., Schultz, T., 2005. Dynamic language model adaptation using variational bayes inference. In: Proceedings of Interspeech. pp. 5–8.

Tam, Y., Schultz, T., 2006. Unsupervised language model adaptation using latent semantic marginals. In: Proceedings of Interspeech.

Turney, P. D., Pantel, P., 2010. From frequency to meaning : Vector space models of semantics. Journal of Artificial Intelligence Research, 141–188.

Virpioja, S., Vyrynen, J. J., Creutz, M., Sadeniemi, M., 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In: PROC. OF MT SUMMIT XI. pp. 491–498.

Wang, S., Schuurmans, D., Peng, F., Zhao, Y., 2003. Semantic n-gram language modeling with the latent maximum entropy principle. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP03).

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.

Whittaker, E., Woodland, P., 2003. Language modelling for russian and english using words and classes. Computer Speech & Language 17 (1), 87 – 104.

Zhao, Y., Karypis, G., 2002. Criterion functions for document clustering: Experiments and analysis. Tech. rep., Department of Computer Science, University of Minnesota, Minneapolis.

Appendix B

Author's publications

"Milk is for babies. When you grow up you have to drink beer."

— Arnold Schwarzenegger

Journal publications

- Tomáš Brychcín and Miloslav Konopík.
  Latent semantics in language models.
  In *Computer Speech & Language*, Volume 33, Issue 1, 2015, Pages 88–108, ISSN 0885-2308.

- Michal Konkol and Tomáš Brychcín and Miloslav Konopík.
  Latent semantics in named entity recognition.
  In *Expert Systems with Applications*, Volume 42, Issue 7, 2015, Pages 3470–3479, ISSN 0957–4174.

- Tomáš Brychcín and Miloslav Konopík.
  HPS: High precision stemmer.
  In *Information Processing & Management*, Volume 51, Issue 1, 2015, Pages 68-91, ISSN 0306-4573.

- Tomáš Brychcín and Miloslav Konopík.
  Semantic spaces for improving language modeling.
  In *Computer Speech & Language*, Volume 28, Issue 1, 2014, Pages 192–209, ISSN 0885-2308.

Conference publications

- Tomáš Brychcín and Pavel Král.
  Novel Unsupervised Features for Czech Multi-label Document Classification.
  In *Proceedings of the 13th Mexican International Conference on Artificial Intelligence*, 2014, Pages 70–79.

- Tomáš Brychcín and Michal Konkol and Josef Steinberger.
  UWB: Machine Learning Approach to Aspect-Based Sentiment Analysis.
  In *Proceedings of the 8th International Workshop on Semantic Evaluation*, 2014, Pages 817–822.

- Josef Steinberger and Tomáš Brychcín and Michal Konkol.
  Aspect-Level Sentiment Analysis in Czech.
  In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2014, Pages 24–30.

- Tomáš Brychcín and Ivan Habernal.
  Unsupervised Improving of Sentiment Analysis Using Global Target Context.
  In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, 2013, Pages 122–128.

- Ivan Habernal and Tomáš Brychcín.
  Semantic Spaces for Sentiment Analysis.
  In *Proceedings of the 16th international conference on Text, Speech and Dialogue*, 2013, Pages 482–489.

- Tomáš Brychcín and Miloslav Konopík.
  Morphological Based Language Models for Inflectional Languages.
  In *Proceedings of IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, 2011, Pages 560–563.

- Miloslav Konopík and Ivan Habernal and Tomáš Brychcín.
  N-gram Language Models in JLASER Neural Network Speech Recognizer.
  In *Proceedings of IEEE International Conference on Applied Electronics*, 2010, Pages 1–4.

- Tomáš Pavelka and Tomáš Brychcín.
  N-Best Decoder for the JLASER Automatic Speech Recognizer.
  In *Proceedings of the 9th international PhD Workshop on Systems and Control*, 2008, Pages 1–4.

Bibliography

Arisoy, E., Dutağacı, H., and Arslan, L. M. (2006). A unified language model for large vocabulary continuous speech recognition of turkish. *Signal Processing*, 86(10):2844–2862.

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

Blei, D. M. and Lafferty, J. D. (2006). Correlated Topic Models.

Blei, D. M., Ng, A. Y., Jordan, M. I., and Lafferty, J. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

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

Brychcín, T., Konkol, M., and Steinberger, J. (2014). Uwb: Machine learning approach to aspect-based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 817–822. Association for Computational Linguistics.

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

Brychcín, T. and Konopík, M. (2015a). Hps: High precision stemmer. *Information Processing & Management*, 51(1):68–91.

Brychcín, T. and Konopík, M. (2015b). Latent semantics in language models. *Computer Speech & Language*, 33(1):88–108.

Brychcín, T. and Král, P. (2014). Novel unsupervised features for czech multi-label document classification. In Gelbukh, A., Espinoza, F., and Galicia-Haro, S., editors, *Human-Inspired Computing and Its Applications*, volume 8856 of *Lecture Notes in Computer Science*, pages 70–79. Springer International Publishing.

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

Chen, S. F. and Goodman, J. T. (1998). An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University.

Cohen, T., Schvaneveldt, R., and Widdows, D. (2010). Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. *Journal of Biomedical Informatics*, 43(2):240–256.

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience, New York, NY, USA.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, pages 391–407.

Dempster, A. P., Laird, N. M., and 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.

Firth, J. R. (1957). A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis*, pages 1–32.

Gildea, D. and Hofmann, T. (1999). Topic-based language models using em. In *Proceedings of Eurospeech*, pages 2167–2170.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3/4):237–264.

Habernal, I. and Brychcín, T. (2013). Semantic spaces for sentiment analysis. In *Text, Speech and Dialogue*, volume 8082 of *Lecture Notes in Computer Science*, pages 482–489, Berlin Heidelberg. Springer.

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

Habernal, I., Ptáček, T., and Steinberger, J. (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*, pages 65–74, Atlanta, Georgia. Association for Computational Linguistics.

Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*, pages 289–296.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jones, M. N. and Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

Kirchhoff, K., Vergyri, D., Bilmes, J., Duh, K., and Stolcke, A. (2006). Morphology-based language modeling for conversational Arabic speech recognition. *Computer Speech & Language*, 20(4):589–608.

Konkol, M. (2014). Brainy: A machine learning library. In Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L. A., and Zurada, J. M., editors, *Artificial Intelligence and Soft Computing*, volume 8468 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Konkol, M., Brychcín, T., and Konopík, M. (2015). Latent semantics in named entity recognition. *Expert Systems with Applications*, 42(7):3470–3479.

Konkol, M. and Konopík, M. (2014). Named entity recognition for highly inflectional languages: Effects of various lemmatization and stemming approaches. In Sojka, P., Horák, A., Kopeček, I., and Pala, K., editors, *Text, Speech and Dialogue*, volume 8655 of *Lecture Notes in Computer Science*, pages 267–274. Springer International Publishing.

Li, W. and McCallum, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 577–584, New York, NY, USA. ACM.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods Instruments and Computers*, 28(2):203–208.

McNamara, D. S. (2011). Computational methods to extract meaning from text and advance theories of human cognition. *Topics in Cognitive Science*, 3(1):3–17.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Miller, G. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Oikonomidis, D. and Digalakis, V. (2003). Stem-based maximum entropy language models for inflectional languages. In *INTERSPEECH*. ISCA.

Oparin, I. (2008). *Language Models for Automatic Speech Recognition of Inflectional Languages*. PhD thesis, University of West Bohemia, Pilsen.

Ptáček, T., Habernal, I., and Hong, J. (2014). Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Purandare, A. and Pedersen, T. (2004). Word sense discrimination by clustering contexts in vector and similarity spaces. *Proceedings of 8th Conference on Computational Natural Language Learning*, pages 41–48.

Rapp, R. (2003). Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Ninth Machine Translation Summit*, pages 315–322.

Riordan, B. and Jones, M. N. (2011). Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation. *Topics in Cognitive Science*, 3(2):303–345.

Rohde, D. L. T., Gonnerman, L. M., and Plaut, D. C. (2004). An improved method for deriving word meaning from lexical co-occurrence. *Cognitive Psychology*, 7:573–605.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

Sahlgren, M. (2005). An Introduction to Random Indexing. *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

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

Tam, Y. and Schultz, T. (2006). Unsupervised language model adaptation using latent semantic marginals. In *Proceedings of Interspeech*.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning : Vector space models of semantics. *Journal of Artificial Intelligence Research*, pages 141–188.

Wang, S., Schuurmans, D., Peng, F., and Zhao, Y. (2003). Semantic n-gram language modeling with the latent maximum entropy principle. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP03)*.

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

Whittaker, E. W. D. and Woodland, P. C. (2003). Language modelling for Russian and English using words and classes. *Computer Speech & Language*, 17:87–104.