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.

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

[☆] This document is a collaborative effort.

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

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 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.

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

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, 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

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

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

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. The vocabulary size for each language

[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.

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 lin.
	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 lin.
	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 lin.
	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).

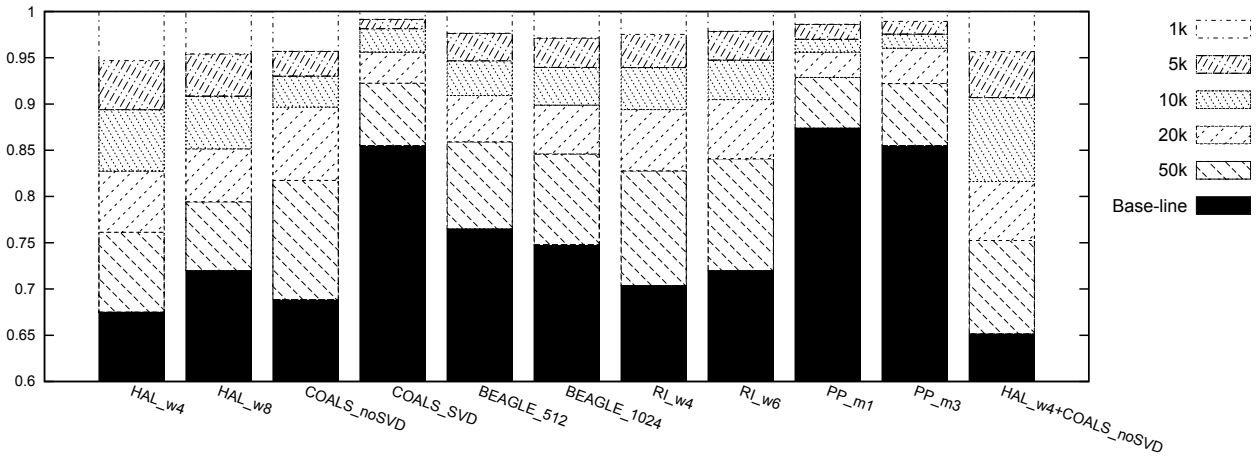
As we indicated before, the HAL-based models are efficient in case of dense clusters (1,000 or 5,000 classes) and the COALS model looks more suitable for sparse clusters (20,000 or 50,000 classes). The combination of both (the last row of the tables) provides the best perplexity results for each corpus. 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.

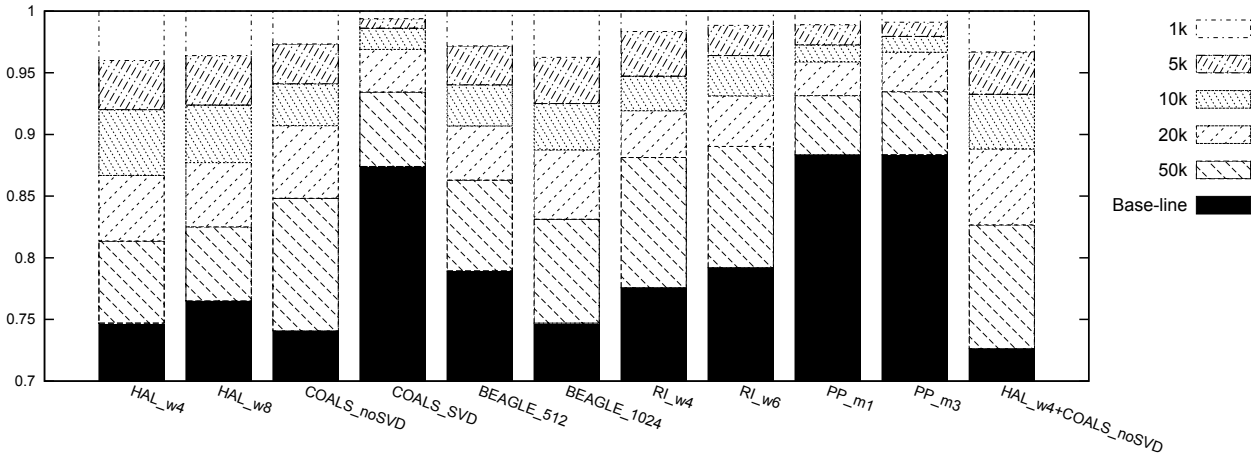
4.3.1. Interpolation weights

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.

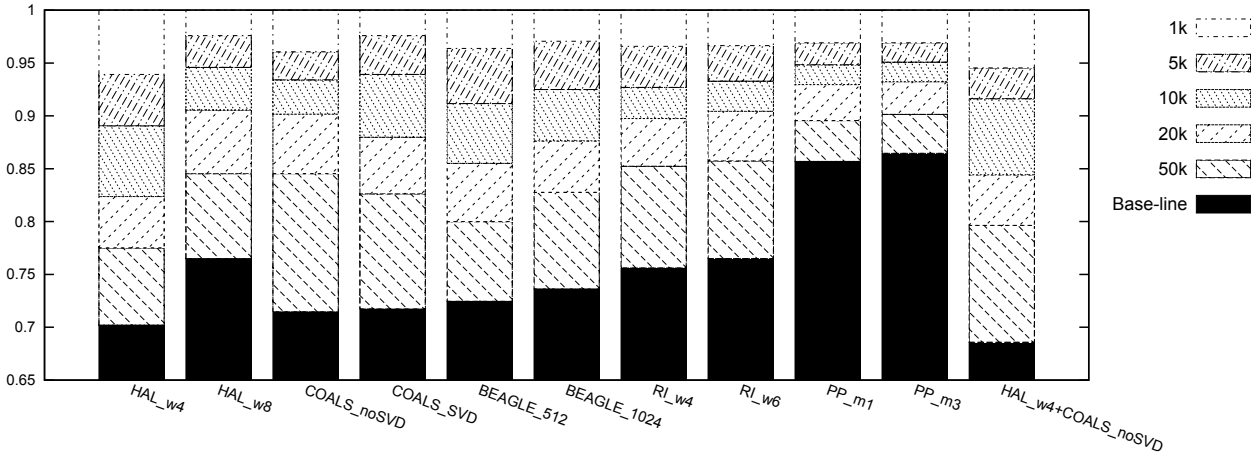
Figures show a trend where the sparse clusters (20k or 50k classes) received bigger weights (immediately after the baseline) and the weights decrease with decreasing number of classes. A direct correlation between weights and perplexity results (section 4.3) can be seen. The bigger the weights allocated 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 lin.
	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 lin.
	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 lin.
	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

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

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.

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 lin.
	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 lin.
	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 lin.
	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 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

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

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.