ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Unsupervised Methods for Language Modeling

PhD Study Report

Tomáš Brychcín

Technical Report No. DCSE/TR-2012-03
June, 2012

Distribution: Public

Abstract

Language models are crucial for many tasks in NLP[1] 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.

Thesis summarizes the state-of-the-art approaches to unsupervised language modeling with emphases on the inflectional languages, which are particularly hard to model. It is focused on methods that can discover hidden patterns that are already in a training corpora. These patterns can be very useful for enhancing the performance of language modeling, moreover they do not require additional information sources.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitní 8
30614 Pilsen
Czech Republic

[1] Natural Language Processing

Contents

Chapter 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. Many studies have been proposed about improving n-gram language models by adding external information (morphology, syntax, etc.) about language and significant improvements have been achieved. This approaches supervised by linguists are likely to be very efficient, because they are based on the prior knowledge of the language. There are, however, several disadvantages. There is not always possible to have an expert in this field. Creation of languages rules or manual annotation of data is an expensive and time consuming process. Through these facts the benefits given by the effectiveness are much outweighed especially when we want to work with additional language.

In last years the unsupervised methods become popular. These methods require more complex techniques, however they ensure no cost to model additional language. This thesis concentrates on the unsupervised methods for improving language modeling. Thesis is also focused on inflectional languages as we believe that these languages offer some room for improvement. However, modeling of these languages is more difficult due the significant data sparsity problem.

Thesis is organized as follows. The state-of-the-art architectures for language modeling are discussed in chapter 2. Chapter 3 describes clustering methods that play the key role in unsupervised techniques. The language models that use morphology of the language are investigated in chapter 4 and the models that try to bring semantic information into language models are discussed in chapter 5. Some preliminary ideas to future development

that imply the aims of doctoral thesis are discussed in chapter 6.

Chapter 2

Language models

The goal of language model is very simple, to estimate probability of any word occurrence possible in the language. Even the task looks very easy, the satisfactory solution for natural language is very complicated.

2.1 Statistical language models

In this work we will look at the language as the information source that produces word sequences from some word vocabulary. Let W denotes the word vocabulary. The $W^{\mathbb{N}}$ is the set of all combination of word sequences possible to create from the vocabulary W. Let

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

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

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

$$S = w_1^k = w_1, \cdots, w_k, \qquad S \in \mathcal{L}. \tag{2.2}$$

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

The only way to calculate this probability correctly should be to process all utterances (that have been ever written, spoken, etc.) of that language $\mathcal{L}$ and calculate the frequency of this sequence over all possibilities. On first looking this is impossible. This is actually even more complicated, because the natural language is evolving process. New expressions regularly enter

the language while others die out. So 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 as large training data as possible. The probability estimation of $P(S)$ will be referred to $\tilde{P}(S)$ in the following text

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

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

$$P(S) = P(w_1^k) = P(w_1)P(w_2|w_1)\cdots P(w_k|w_1^{k-1}) = \prod_{i=1}^{k} P\left(w_i|w_1^{i-1}\right). \tag{2.4}$$

2.2 Evaluation

To be able to compare two different language models it is required to have some measure of quality for these language models. Of course, the best way of language model evaluation is to evaluate the whole system where the language model is used. However, there is not always possible to measure performance of the whole system so 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 on information theory. The information theory measures the amount of information by entropy of source (i.e. of modeling language in our case). Entropy of language is defined as

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

In 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 logarithm is 2, the entropy is measured in bits). In terms of language modeling, the entropy means the average number of bits needed to encode each word by word in the text. Entropy is a measure of uncertainty. Lower entropy means that the following word in the text is more predictable in average. The bigger the probability is assigned to the text by language model, the lower the entropy is and the better predictability the language model has.

There are of course a few problems. We do not know the correct probabilities $P(S)$ of modeling language so we can only estimate entropy. So called *cross entropy* is defined as

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

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

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

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

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

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

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

It means that it is not possible to create a better language model than the original source model producing the correct probabilities $P(S)$. The lower the cross entropy our language model has, the better it approximates the modeling language.

Let

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

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

2.3 N-gram language models

In previous subsection the formula 2.4 shows that the probability of each word w_i is conditioned by 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 k. The number of training parameters needed to be estimated rises exponentially with extending the history.

According to all problems mentioned above, truncating the word history is done to decrease the number of training parameters. It means, that the

probability of word w_i is estimated only by $n-1$ preceding words (not by complete history)

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

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 since a long time. N-grams, where $n = 1$, are called *unigrams*. The most often used are, however, *bigrams* ($n = 2$) and *trigrams* ($n = 3$).

Note that even for the trigram model of natural language, the number of training parameters is still enormous. For example, in the case of 65,536 words 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 for parameters and probability estimate retrieval time will not be satisfactory.

The lack of training data is sometimes referred to as the *data sparsity problem*.

2.4 Smoothing

Now, let we try to estimate the probability of word w_i conditioned by the fact that history of $n-1$ words corresponds to w_{i-n+1}^{i-1}. Let the $c\left(w_{i-n+1}^{i}\right)$ denotes the frequency of n-gram w_{i-n+1}^{i} in training data. Then the so called *maximum likelihood estimation (MLE)* is defined as

$$P\left(w_i | w_1^{i-1}\right) \approx P^{MLE}\left(w_i | w_{i-n+1}^{i-1}\right) = \frac{c\left(w_{i-n+1}^{i}\right)}{\sum_{w_i \in W} c\left(w_{i-n+1}^{i}\right)} = \frac{c\left(w_{i-n+1}^{i}\right)}{c\left(w_{i-n+1}^{i-1}\right)}. \tag{2.12}$$

Simply, number of times that word w_i occurs after the history w_{i-n+1}^{i-1} over the frequency of this history.

On first looking, it seems that this method leads to very good probability estimations, but as it was described earlier, there is a problem with data sparsity. The probability estimate of word w_i that has never be seen after the history w_{i-n+1}^{i-1} in the training data, is 0. Even if in the modeling language the words could follow it this order, the MLE method will still give as zero probability estimates. This problem is sometimes called the *zero problem* and it is solved by *smoothing* probabilities.

The goal of smoothing is simple. To spread out part of probability mass of seen events (seen n-grams) to unseen events and eliminate the zero problem. Detailed overview about smoothing techniques is presented in (Chen and Goodman, 1998). In following subsections we will describe some of them.

2.4.1 Additive smoothing

Additive smoothing, sometimes also called *Laplace smoothing* is probably the simplest and simultaneously the least efficient smoothing technique for language modeling. This method is described only for its simplicity to demonstrate the smoothing principle.

The main idea is to artificially increase the frequency of each n-gram about some $\delta > 0$

$$P\left(w_i|w_1^{i-1}\right) \approx P^{add}\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{c\left(w_{i-n+1}^{i}\right) + \delta}{c\left(w_{i-n+1}^{i-1}\right) + \delta\left|W\right|}. \tag{2.13}$$

The smoothed conditional probabilities must sum up to 1

$$\sum_{w_i \in W} P^{add}\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{\sum\limits_{w_i \in W}\left[c\left(w_{i-n+1}^{i}\right) + \delta\right]}{c\left(w_{i-n+1}^{i-1}\right) + \delta\left|W\right|} = \frac{c\left(w_{i-n+1}^{i-1}\right) + \delta\left|W\right|}{c\left(w_{i-n+1}^{i-1}\right) + \delta\left|W\right|} = 1. \tag{2.14}$$

2.4.2 Good-Touring estimation

Good-Touring estimation (Good, 1953) is another smoothing method that can be applied into language modeling. However, this technique is seldom used alone, many other methods use it for parameter estimation.

Firstly, let N_r denotes the number of different n-grams that occur in training data exactly r-times

$$N_r = \sum_{w_{i-n+1}^{i}:c(w_{i-n+1}^{i})=r} 1, \qquad 1 \leq r < \infty. \tag{2.15}$$

This number is sometimes referred to as the *frequency of frequency*. Especially, for N_0 we must estimate the number of unseen events (n-grams). Simply, the number of all possible n-grams minus the number of seen n-grams.

The basic idea of Good-Touring estimation is that the frequencies of n-grams $c(w_{i-n+1}^{i})$ are modified to $c^{GT}(w_{i-n+1}^{i})$ according the equation

$$c^{GT}\left(w_{i-n+1}^{i}\right) = (r+1)\frac{N_{r+1}}{N_r}, \qquad r = c\left(w_{i-n+1}^{i}\right). \tag{2.16}$$

The probability $P\left(w_i|w_1^{i-1}\right)$ is than estimated as

$$P\left(w_i|w_1^{i-1}\right) \approx P^{GT}\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{c^{GT}\left(w_{i-n+1}^{i}\right)}{c\left(w_{i-n+1}^{i-1}\right)}. \tag{2.17}$$

The proof that the probabilities sum up to 1 can be found in original paper.

2.5 Model combination

In this section we will continue with describing different smoothing methods, but where the smoothing is done by way of combination of several different language models.

2.5.1 Linear interpolation

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

$$P^{LI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right), \tag{2.18}$$

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

$$\sum_{k=1}^{K} \lambda_k = 1. \tag{2.19}$$

The *expectation-maximization (EM)* algorithm (described in section 2.7) is used to calculate optimal weights λ_k by maximization of the probability of the data.

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

2.5.2 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. The main idea is that the weights λ_k should be different for words with histories of varying frequencies. The formula then transforms to

$$P^{BLI}\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{k=1}^{K} \lambda_k\left(w_{i-n+1}^{i-1}\right) \cdot P_k\left(w_i|w_{i-n+1}^{i-1}\right). \tag{2.20}$$

The weights $\lambda_k()$ certainly cannot be different for each possible frequency of history. Too many 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.

2.5.3 Back-off smoothing

Another architecture for combining different language models is based on so called *back-off* principle. It is described by following equation

$$p^{BO}\left(w_i|w_{i-n+1}^{i-1}\right) = \begin{cases} \alpha\left(w_i|w_{i-n+1}^{i-1}\right) & \text{if } c\left(w_{i-n+1}^{i}\right) > 0 \\ \gamma\left(w_{i-n+1}^{i-1}\right) P^{BO}\left(w_i|w_{i-n+2}^{i-1}\right) & \text{otherwise} \end{cases} \tag{2.21}$$

If an n-gram has nonzero frequency the probability is estimated by the distribution $\alpha\left(w_i|w_{i-n+1}^{i-1}\right)$. Otherwise, we use the lower-order distribution $P^{BO}\left(w_i|w_{i-n+2}^{i-1}\right)$. These back-off steps are repeated until it is needed. The discounting function $\alpha\left(w_i|w_{i-n+1}^{i-1}\right)$ is constructed in a way to save some probability mass for lower-order models. The scaling function $\gamma\left(w_{i-n+1}^{i-1}\right)$ is chosen to make the probability distribution sum up to 1. Many authors present different ways to setting appropriate discounting and scaling functions.

In the previous section the linear interpolation of models was described. However, both architecture of combining models (linear interpolation, back-off) used lower-order distributions to determine probability of unseen n-grams, there is one big difference between them. Interpolated models use information from lower-order distributions for calculating probability of n-grams with nonzero frequency, while back-off models do not.

2.5.4 Kneser-Ney smoothing

Authors in (Kneser and Ney, 1995) have introduced improved back-off smoothing. It is derived by

$$P^{KN}\left(w_i|w_{i-n+1}^{i-1}\right) = \begin{cases} \frac{c\left(w_{i-n+1}^{i}\right)-D}{\sum_{w_i} c\left(w_{i-n+1}^{i}\right)} & \text{if } c\left(w_{i-n+1}^{i}\right) > 0 \\ \gamma\left(w_{i-n+1}^{i-1}\right) P^{KN}\left(w_i|w_{i-n+2}^{i-1}\right) & \text{otherwise} \end{cases} \tag{2.22}$$

where $0 < D \leq 1$ is a fixed coefficient subtracted from frequency of seen n-grams, that save part of probability mass for unseen n-grams.

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

$$P\left(w_i\right) = \frac{N_{1+}\left(\bullet w_i\right)}{N_{1+}\left(\bullet\bullet\right)}, \tag{2.23}$$

where 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 $c(w_{i-n+1}^{i}) \geq 1$). 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.

In (Chen and Goodman, 1998) this smoothing was upgraded to so called *modified Kneser-Ney interpolation*, which at present is the state-of-the-art approach for smoothing methods. Firstly, instead of using the back-off principle, the linear interpolation is used. Secondly, the discounting coefficient becomes the function of n-gram frequency.

The formula for smoothing of word probabilities is

$$P^{MKN}(w_i|w_{i-n+1}^{i-1}) = \\ \frac{c(w_{i-n+1}^{i})-D(c(w_{i-n+1}^{i}))}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma\left(w_{i-n+1}^{i-1}\right) P^{MKN}\left(w_i|w_{i-n+2}^{i-1}\right). \tag{2.24}$$

The definitions and derivations of functions $D()$, $\gamma()$ may be founded in the original paper.

2.5.5 Log-linear interpolation

The log-linear interpolation is another method for combining models. It has been show in (Klakow, 1998) that this method is likely to be more effective than the linear interpolation (see subsection 2.5.1). Log-linear interpolation is given by equation

$$P^{LLI}\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{1}{Z\left(w_{i-n+1}^{i-1}\right)} \prod_{k=1}^{K} P_k\left(w_i|w_{i-n+1}^{i-1}\right)^{\lambda_k}, \tag{2.25}$$

where $Z\left(w_{i-n+1}^{i-1}\right)$ is normalization function that ensures that probabilities of words w_i after each history w_{i-n+1}^{i-1} sum up to 1. Note that there is a big problem with this normalization, because we must normalize probabilities over all words, which could follow after history.

The parameters λ_k of log-linear interpolation can be optimized again by some variation of *expectation-maximization* algorithm such as *improved iterative scaling*[1].

2.6 Other architectures

2.6.1 Class-based n-gram models

Class-based modeling is the most popular technique used for reducing the huge vocabulary-related sparseness of statistical language models (Brown

[1]Improved iterative scaling (IIS) is a hill-climbing algorithm for finding optimal parameters in log-likelihood space. Algorithm is described for example in (Darroch and Ratcliff, 1972, Berger et al., 1996)

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.

Let W denotes the set of possible words (word vocabulary) and C denotes 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} (where n is the length of the n-gram) is given by the following formula

$$\tilde{P}\left(w_i|w_{i-n+1}^{i-1}\right) = \tilde{P}\left(w_i|c_i\right) \cdot \tilde{P}\left(c_i|c_{i-n+1}^{i-1}\right). \tag{2.26}$$

The probability estimate of the word occurrence given by its class is calculated as follows

$$\tilde{P}\left(w_i|c_i\right) = \frac{c\left(w_i, c_i\right)}{c\left(c_i\right)}, \tag{2.27}$$

where $c(w_i, c_i)$ is the number of times the word w_i is mapped to the class c_i over the frequency of class c_i.

2.6.2 Maximum entropy

Maximum entropy models have been successfully applied into language modeling in (Berger et al., 1996, Rosenfeld, 1996). Maximum entropy is used in such a way of smoothing that combines a various language constrains in fundamentally different way than the above described techniques. Instead of combining models themselves, their features are combined into one new model.

Firstly, we describe the maximum entropy framework from a general point of view and then the application into language modeling is specified. Goal is to estimate the conditional probability $p(y|x)$ of y as the observation on the output of the process given by the knowledge x about y. The y is an event that is being predicted, a member of a finite set Y. The event y could be predicted by some knowledge x, the member of a finite set X.

The training data is used to set constraints on the conditional distribution. Each constraint expresses a characteristic (knowledges) about the training data that we also want to present in the final probability distribution.

To express these facts (the knowledges) about training data we denote m real valued feature functions $f_i(x, y) \in \langle 0, 1\rangle$.

The final model distribution should be restricted to have the same expected values for all features as it was seen in the training data. We denote this idea as

$$E\left(f_i\left(x, y\right)\right) = \tilde{E}\left(f_i\left(x, y\right)\right), \quad 1 \leq i \leq m, \tag{2.28}$$

where $\tilde{E}(f_i(x,y))$ is an expectation value of feature $f_i(x,y)$ estimated from training data and $E(f_i(x,y))$ is an expectation value of this feature given by the final model. This formula can be expressed in details as

$$\sum_{x,y} \tilde{p}(x)\, p(y|x) f_i(x,y) = \sum_{x,y} \tilde{p}(x,y)\, f_i(x,y), \quad 1 \leq i \leq m, \tag{2.29}$$

where $p(y|x)$ is given by probability distribution of final model and $\tilde{p}(x)$ and $\tilde{p}(x,y)$ are estimations of probabilities from training data.

According to all these constraints on final distribution, the last question need to be answered. What is the best distribution of p among the feature functions $f_i(x,y)$ that are available? The maximum entropy philosophy says the best is the most uniform distribution that satisfied the constraints (the one with the greatest entropy).

According to (Berger et al., 1996), the model that maximizes entropy of p has the exponential form

$$p(y|x) = \frac{1}{Z(x)} \prod_{i=1}^{m} e^{\lambda_i f_i(x,y)}, \tag{2.30}$$

where $Z(x)$ is a normalization function. The parameters λ_i of maximum entropy model could be estimated for example by OWL-GN[2] procedure, or another algorithm for finding global maximum of the function such as IIS.

To apply maximum entropy approach into language modeling the features need to be defined. As it was described above, the features represent the knowledge about data so the features could be defined for example as

$$f\left(w_{i-n+1}^{i-1}, w_i\right) = \begin{cases} 1 & \text{if } w_i \text{ follows after } w_{i-n+1}^{i-1} \\ 0 & otherwise \end{cases} \tag{2.31}$$

where the word w_i represents the output y and word history w_{i-n+1}^{i-1} represents the knowledge x about y.

2.6.3 Factored language models

In *factored language model (FLM)* (Bilmes and Kirchhoff, 2003, Kirchhoff et al., 2006, 2008) the word is viewed as a vector of K factors

$$w_i \equiv \left\{f_i^1, f_i^2, \ldots, f_i^K\right\} = f_i^{1:K} \tag{2.32}$$

[2]OWL-GN (Orthant-Wise Limited-memory Quasi-Newton) described in (Andrew and Gao, 2007) is an algorithm for efficient optimization a huge number of parameters in log-linear models. It is based on L-BFGS (Limited-memory variation of the Broyden-Fletcher-Goldfarb-Shanno) algorithm, however authors present that OWL-GN is much faster than other algorithms.

where each factor can be any knowledge about the word such as word classes, stem, root, part-of-speech etc. even the word itself.

Probability of word w_i conditioned by history w_{i-n+1}^{i-1} can be expressed by factored language model and by application a chain rule as

$$\begin{aligned} P^{FLM}\left(w_i|w_{i-n+1}^{i-1}\right) &\equiv P\left(f_i^{1:K}|f_{i-1}^{1:K},\ldots,f_{i-n+1}^{1:K}\right) \\ &= \prod_{k=1}^{K} P\left(f_i^k|f_i^{1:k-1},f_{i-1}^{1:K},\ldots,f_{i-n+1}^{1:K}\right) \end{aligned} \tag{2.33}$$

Note that certainly not all factors have to be used to effectively estimate the probability. For notation simplicity, let the conditional probabilities in equation above are expressed as $P\left(F|F_1,F_2,\ldots,F_N\right)$, where each F represents some factor f. The goal of FLM is then to estimate these conditional probabilities.

In standard back-off language models (see subsection 2.5.3) probability of unseen events is estimated by lower-order distribution. This process can be visualized as a back-off path on figure 2.1.

For smoothing of factored language models the similar idea to the back-off scheme is used, but there is not clear in witch order we should drop the variables. This is caused by the fact that in FLM the conditional variable are not exclusively words and so there is not clear witch variable should contain more mutual information about the word that is being predicted.

Through these facts, so called *generalized parallel back-off scheme* is used. Figure 2.2 shows all possible back-off paths for 4-gram factored language model.

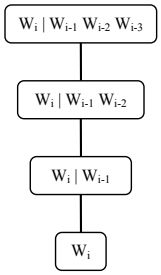

Figure 2.1: Back-off path in standard 4-gram language model.

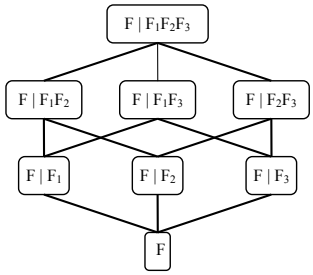

Figure 2.2: Back-off graph for a factored 4-gram language model.

There are many possibilities to selecting appropriate back-off path. Selection can be based on linguistic knowledge, can be made automatically according some statistical criteria, even the multiple paths can be selected to interpolate their probability estimates. The formula for generalized back-off smoothing is

$$P^{GBO}\left(F|F_1,\ldots,F_N\right) = \begin{cases} \alpha\left(F|F_1,\ldots,F_N\right) & \text{if } c\left(F,F_1,\ldots,F_N\right) > \tau \\ \gamma\left(F_1,\ldots,F_N\right) g\left(F,F_1,\ldots,F_N\right) & \text{otherwise} \end{cases} \tag{2.34}$$

where $\alpha\left(F|F_1,\ldots,F_N\right)$ is a discounting function that saves part of probability mass for unseen events. The $\gamma\left(F_1,\ldots,F_N\right)$ is a normalization function. The function $g\left(F,F_1,\ldots,F_N\right)$ determines a back-off strategy (choose a path) and τ is a count threshold. Definition and derivation of these functions can be found in original papers.

2.6.4 Decision trees

Language models can be seen as a classification problem where we want to classify the history of words w_1^{i-1} to some word w_i (we want to estimate the probability that this word w_i follows after this history).

A *decision tree language models* (Bahl et al., 1989, Xu and Jelinek, 2007, Oparin, 2008) attempt to classify all histories into equivalence classes and each history in the same equivalence class shares the same probability distribution over the predicted words w_i, which could follow.

A decision tree $\mathfrak{T}$ consists of a finite set of internal nodes $\mathfrak{N}$ and finite set of leaf nodes $\mathfrak{L}$. In a decision tree we ask questions about histories of the word we try to predict. These questions are associated with each internal node $p \in \mathfrak{N}$. According to the answer we choose the path to another node that follows in the tree $\mathfrak{T}$. For simplicity only *yes/no* questions are considered so it is the binary decision tree we are talking about. Questions can be anything and can use any knowledge about word history. The leaf nodes $l \in \mathfrak{L}$ represent the equivalence classes of histories. Clustering of word histories is a key idea why the decision trees became popular in language modeling, because it particularly solves the problem of data sparsity that the language models fight with.

The selection of questions and decision tree construction play the key role in a decision tree language modeling. There are two possible strategies. The tree construction can be manually created by linguist or can be done automatically. In original papers an efficient algorithm can be found, where the entropy on held-out data is used as a goodness measure for tree building.

Decision tree language models can be extended to so called *random forest language models* (Xu and Jelinek, 2007, Oparin, 2008). Random forest is a collection of decision trees that includes randomization in the process of building the tree. The underlying assumption is that the randomization should generalize the decision trees to perform better on unseen data. More information about the construction of random forests can be found in original papers.

2.7 Expectation-maximization algorithm

The *expectation-maximization (EM)* algorithm (Dempster et al., 1977) is an iterative process for finding maximum likelihood estimations of parameters in statistical models (this is an equivalent to minimizing the entropy). Important is the fact, that the parameters need to be tuned on the so called *held-out data*. The held-out data are different from the training data as well as from the data used for evaluation of performance.

Imagine the case where we want to linearly interpolate the trigram, bigram and unigram language model to achieve smoothed probabilities (as it is describe in subsection 2.5.1). If we estimate the parameters of linear interpolation on the same data that the sub-models were trained on, the trigram model receives the parameter equal to 1 and other models equal to 0. This would be caused by the fact, that the trigram model provides the most appropriate probability estimations on this data.

We demonstrate the EM algorithm on the problem of optimizing parameters of linear interpolation of language models:

1. At first, the initial estimation of parameters λ_k^0 for all $1 \leq k < K$ need to be done. Initial parameters must satisfy the equation 2.19 as well as $\lambda_k^0 > 0$. At second, the stopping threshold ε is defined.

2. *E-step*: The expected parameters $\hat{\lambda}_j^t$ are calculated by

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

   where t denotes the number of iteration.

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

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

4. If the condition

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

   is satisfied, than the iterative process ends. Otherwise, the process continues with point number 2.

Chapter 3

Clustering

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[1] 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.

The optimality criterion can be supplied for example from a semantic space (section 5.2) and an appropriate similarity metric (subsection 5.2.6).

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).

A useful guide can be was found in the article by (Zhao and Karypis, 2002) where a comparison of clustering methods was presented. Many clustering methods are implemented in the CLUTO software package (Karypis, 2003).

The clusters given by some clustering method can be used to build the class-based language models (subsection 2.6.1). Note that the clustering of words in natural language can be very time consuming process, because it deals with really large quantities of data.

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).

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

3.1 MMI clustering

In (Brown et al., 1992) the MMI[2] clustering algorithm was introduced. The idea is simple. What if we cluster together the words in such way of maximizing average mutual information between word classes on training data? These clusters than use for the class-based language models training to decrease well known data sparsity problem. However, there is no way how to find partitioning that maximizes average mutual information over so many possibilities (see previous section). In original paper the *greedy algorithm* is described, which performs very well and in clever way decrease the complexity to order of $|W|^3$. This complexity is however still very problematic. The 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. This method of word clustering is possible only on 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% 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.

[2]Maximum Mutual Information

Chapter 4

Word morphology

During investigation of language modeling it is natural to talk about morphology of the language. As it was said at the beginning of the thesis, we are focused on modeling of inflectional languages. These languages are characterized by rich inflection. The inflection of words in natural language is caused by some linguistic rules of this language. By adding inflectional morphemes to the base form of the word or sometimes by changing entire term, the new inflection may be formed. In many languages this could lead to many different word forms with the same base form.

For example, Czech, which belongs to the group of Slavic languages, is highly inflectional. Czech language has seven cases and three genders. It has a relatively free word order (from the purely syntactic point of view): words in a sentence can usually be ordered in several ways, which carry a slightly different meaning.

These properties of inflectional languages complicate the language modeling task. High number of word forms and more sequences of words that are possible in the language lead to a higher number of n-grams. In chapter 2 the data sparsity problem was emphasized. In inflectional languages this problem is much more evident.

Some earlier works on application of morphology analysis into language modeling are summarizer in section 4.1. However, there are exist even unsupervised methods for morphology learning mainly focused on stemming of words. Stemming can play a key role especially when we work with inflectional languages. These methods are summarized in section 4.2.

4.1 Morphological based language models

An effective solution for language modeling is to use information about the morphology of the language. In (Oparin, 2008) experiments with morphological random forests (subsection 2.6.4) in the Czech and Russian language are shown with the conclusion that they can be used effectively for inflec-

tional 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 (such as class-based models, factor based models, etc.) 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.

4.2 Stemming

The word stemming is one of the basic preprocessing techniques in NLP. Information retrieval (IR) tasks, the machine translations systems and many applications in NLP use stemming. Every improvement in word stemming may also improve the particular job where the stemmer is used. Stemmers attempt to reduce a word to its base form (to stem or root form).

The stemming algorithms can be divided into two categories (the rule based stemmers and statistical ones). Rule based stemmers attempt to transform the word form to its base form using the set of language-specific rules manually created by linguists.

The most known and most often used is the Porter's algorithm (Porter, 1980) that was evolved into whole stemming framework called *Snowball*[1].

In (Dolamic and Savoy, 2009) two rule based stemmers (*light* and *aggressive*) for Czech are introduced[2]. Both stemmers work approximately equally well. Authors present retrieval improvement about 45% by using this stemmers in IR systems.

These approaches supervised by linguists are likely to be more effective than the statistical ones, because they are based on the prior knowledge of the language. However, they have many disadvantages. There is not always possible to have an expert in this field. The benefits given by the effectiveness are much outweighed by the time necessary to morphological analysis especially when we want to work with completely new language. Through these facts the unsupervised methods to word stemming are expanding and

[1] Snowball is a string-handling programming language developed by M.F. Porter. The stemming algorithms can be easily defined in this language. In addition, ANSI C or Java programs can be automatically generated. It is briefly described at http://snowball.tartarus.org together with stemmers for several languages available here.

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

they became crucial in present. These methods try to statistically infer some formation rules from latent patterns in text corpora, without any knowledge about language. These methods require more complex techniques, however they ensure no cost to model additional language. Many studies about the unsupervised learning of morphology of language were presented earlier. The exhausting survey about this problem is discussed in (Hammarström and Borin, 2011).

Interesting method for unsupervised morphological analysis was described in (Goldsmith, 2001). This method is based on MDL (Minimum Description Length) principle. The algorithm tries to find the optimal breakpoint for each word. Each instance of a word in a corpus uses the same breakpoint, which splits this word into stem and suffix. The optimal distribution of breakpoints is modeled in such a way of minimizing number of bits to encode whole collection of words (it is equal to minimizing entropy of this collection). The MDL criterion leads to intuition that breakpoints should segment the word into relative common stems as well as common suffixes. This method is implemented as framework called Linguistica[3] (Goldsmith, 2006).

In (Majumder et al., 2007) the YASS[4] stemmer was introduced. It is very simple approach based on words clustering. The only information that is used is the lexical form of the words (the word lexicon). The set of string distance measures between word pairs is defined. These measures should simulate the morphological similarity between words. The lexicon is than clustered to discover morphological related words (the equivalence classes). Authors present comparable results with rule based stemmers (Porter or Lovins stemmers for English) from retrieval effectiveness point of view. Also for French and Bengali language this approach improves performance against no stemming.

The novel graph based stemmer GRAS[5] was introduced in (Paik et al., 2011). Similarly to the YASS approach, the 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 relationship is represented by a graph, where the words become the 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 root. Authors perform retrieval experiments on seven languages. According the results presented in this article, GRAS outperforms the YASS, Linguistica as well as the rule based stemmer on all seven languages in terms of retrieval effectiveness. For some of these languages, GRAS provides more than 50% performance improvement against no stemming.

[3] Available at http://linguistica.uchicago.edu

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

[5] GRAS (GRAph-based Stemmer)

In (Oikonomidis and Digalakis, 2003) authors present the two different application of stemming into language modeling. As a base-line they choose the language model with modified Kneser-Ney smoothing (subsection 2.5.4). At first, they build class-based language models from stems of words and interpolate it (subsection 2.5.1) with the base-line model. Secondly, they build the maximum entropy model (subsection 2.6.2) from constraint given by the base-line model and information about stems. Both methods bring at least some improvement against the base-line.

Chapter 5

Word semantic

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 that they frequently occur in similar contexts.

There are two main views on semantic modeling that can be used to improve language modeling: to focus on topic (context) (section 5.1) or to focus on word alone (section 5.2).

5.1 Topic models

In NLP the topic model refers to statistical model for discovering the latent topics in collection of text documents. The most known topic models are the LSA (subsection 5.1.1), PLSA (subsection 5.1.2) and LDA (subsection 5.1.3) that are based on so called *bag of words* model. This model assumes that the document is a collection of words where the word order has no significance.

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 WER[1] (down to 16%) in the WSJ[2] 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.

[1] The Word Error Rate (WER) measure is often used in Speech recognition
[2] Wall Street Journal (WSJ)

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.

5.1.1 LSA

Latent Semantic Analysis (LSA) (Deerwester et al., 1990, Landauer and Dumais, 1997, Landauer et al., 1998) is a method that uses a collection of documents to building semantic space.

This algorithm construct $|W| \times |D|$ matrix $\mathbb{M}$, where $|W|$ is number of unique words and $|D|$ is number of documents. Each row correspond to unique word (term) and each column corresponds to a document. Typical example of term weighing is *tf-idf* (term frequency – inverse document frequency). It means, the value at each position of matrix $\mathbb{M}$ (the weight of term) is proportional to frequency the row's term appear in a column's document.

LSA assumes that there is some latent relationship between words in the same document. At final stage, the *Singular Value Decomposition (SVD)* is applied to the matrix $\mathbb{M}$ in order to reduce the dimensionality of the vector space (typically, dimension about 300 is used). The SVD reduction has the effect of bringing out the latent semantic relationships between words so it can discover transitive relations between words.

The LSA method has been successfully applied in many areas of NLP such as language models, text summarization, information retrieval...

5.1.2 PLSA

Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999) is a probabilistic variant of LSA. This method is often prioritized because of its probabilistic output.

Algorithm starts similarly to LSA by building $|W| \times |D|$ matrix $\mathbb{M}$. The PLSA does not use the SVD to discover latent topics, but offers different solution.

The PLSA adds latent topic variable (denoted as Z) into semantic model. The PLSA assumes that the probability of word appearing in a document is related to the probability that this word is produced by each topic $z \in Z$, and the probability of this topic is relevant to the document. It means $P(w|d) = \sum_{z \in Z} P(w|z)P(z|d)$, where $w \in W$ and $d \in D$.

The probabilities are estimated by the *expectation-maximization* algorithm (Dempster et al., 1977) to maximize likelihood of the data.

5.1.3 LDA

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is essentially the Bayesian version of the PLSA model. Each document $d \in D$ is expected to be represented by a mixture of topics. The topics is assumed to have a Dirichlet distribution and each of topics is modeled as a Dirichlet distribution over words $w \in W$.

Similarly to the PLSA, the LDA also uses the *expectation-maximization* algorithm to estimate the optimal parameters of Dirichlet distributions.

5.2 Semantic Spaces

The semantic models investigated in this section 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 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. The fact that a word is characterized by a vector opens up the opportunity to easily compare two words. The ability to compare two words enables us to use a clustering method. Similar words can be clustered into bigger groups of words (clusters) (more information about clustering can be found in chapter 3).

Note that the LSA method (subsection 5.1.1) presented in section about topic models (5.1) also belongs to the semantic spaces because each word is also represented by a vector. 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. However, there are many novel methods (namely HAL

(subsection 5.2.1), COALS (subsection 5.2.2), random indexing (subsection 5.2.3), BEAGLE (subsection 5.2.4), P&P (subsection 5.2.5),...) for word semantic modeling that have never been tested in language modeling before.

5.2.1 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 semantic. 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.

5.2.2 COALS

Correlated Occurrence Analogue to Lexical Semantic (COALS) (Rohde et al., 2004) is a semantic space model based upon HAL and LSA ideals. 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 do 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. The SVD is applied to the matrix $\mathbb{M}$ to achieve similar effects to LSA method. Dimensionality of the vector space is reduced to the dimension typically 800.

This final stage is not mandatory for the COALS method and is sometimes skipped.

5.2.3 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 5.2.4), 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.

5.2.4 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 5.2.3). 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.

5.2.5 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 5.2.1) or the COALS model (subsection 5.2.2)) 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 receive 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.

5.2.6 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

$$S_{mink}\left(\vec{a},\vec{b}\right) = \sqrt[r]{\sum |a_i - b_i|^r}, \tag{5.1}$$

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:

$$S_{\cos}\left(\vec{a},\vec{b}\right)=\cos\left(\theta\right)=\frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\cdot\left\|\vec{b}\right\|}=\frac{\sum a_i b_i}{\sqrt{\sum a_i^2\sum b_i^2}}. \tag{5.2}$$

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

$$S_{corr}\left(\vec{a},\vec{b}\right)=\frac{E\left[\left(\vec{a}-\mu_a\right)\left(\vec{b}-\mu_b\right)\right]}{\sigma_a\sigma_b}=\frac{\sum\left(a_i-\mu_a\right)\left(b_i-\mu_b\right)}{\sqrt{\sum\left(a_i-\mu_a\right)^2\sum\left(b_i-\mu_b\right)^2}}, \tag{5.3}$$

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 matrixes, while the cosine metric is used for sparse matrixes.

Chapter 6

Future work

This chapter describes preliminary ideas for future work that imply the aims of PhD thesis. The three main directions are indicated: the morphological analysis, semantic analysis and OOV word analysis.

In chapter 4 the problems with morphologically rich languages were studied. Many authors successfully deal with these problems by using morphological information about these languages. For example, in Brychcín and Konopík (2011) we investigate the language modeling of inflectional languages enriched about morphological analysis. The performance was significantly enhanced on these languages, but it was done by supervised training on manually annotated data. In section 4.2 several unsupervised stemming algorithms were described. We suppose that some of these algorithms could be also very useful to dealing with rich morphology and its associated data-sparseness problem.

In chapter 5 the unsupervised methods that capture semantic properties of language were described. Some of these methods have already been applied into language modeling with successful results. We believe there is a big potential to research. There are many novel methods for semantic analysis and some of them should be potentially useful also in language modeling.

Working with out-of-vocabulary or low occurred words is a standard problem in many NLP tasks. In language modeling there is also hard to work with them, because they simply did not occur in training data (or occur a too few times). In common smoothing techniques for language modeling the OOV word always receives the same probability estimation (OOV words are uniformly distributed). This is evidently wrong, but how to estimate these probabilities better?

If we think about this problem, the only information we have about OOV word, is the word itself and the context of this word. We suppose that some kind of morphological analysis should particularly solve this problem. Stemming or segmentation into morphemes should discover already seen

patterns and this information can be used for more accurate probability estimations.

Another idea is to use the context of this word. In some cases the occurrence of OOV word can be more probable than in others. Moreover, as it was described in section 5.2 the context of the word is related to the meaning of that word. So we suppose the context information can be used to give a simple estimate of OOV word semantic.

Chapter 7

Summary

This thesis presents overview of the current state-of-the-art in unsupervised approaches for language modeling.

The performance of language models strongly depends on properties of the language and their associated data-sparseness problem. Supervised methods are likely to be very efficient. Data annotation or creating linguistically motivated rules trying to consider specific properties of modeling language is an expensive and time consuming process. Due to these facts, the unsupervised methods become very popular and for poor-resources languages it is unique and easy way to improving performance.

As a future work we choose to reach a hard target. We will try to beat n-gram language models merely with better probability estimations and without any external knowledge (morphology, syntax, partitioning into documents, etc.). My research will be mainly focused on inflectional languages, their modeling efficiency is far worse than efficiency of modeling of low inflectional languages such as English. This thesis is a theoretical preparation for the future development.

7.1 Aims of the PhD thesis

The goal of doctoral thesis is to propose novel unsupervised methods for improving 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 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.

Bibliography

Andrew, G. and Gao, J. (2007). Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 33–40, New York, NY, USA. ACM.

Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1001–1008.

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

Bellegarda, J. R. (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., and 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*, pages 172–175.

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

Bilmes, J. A. and Kirchhoff, K. (2003). Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers - Volume 2*, pages 4–6, Stroudsburg, PA, USA. Association for Computational Linguistics.

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. 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*.

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

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

Darroch, J. N. and Ratcliff, D. (1972). Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and 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., 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.

Dolamic, L. and Savoy, J. (2009). Indexing and stemming approaches for the czech language. *Information Processing and Management*, 45:714–720.

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

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

Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

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

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

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

Hammarström, H. and Borin, L. (2011). Unsupervised learning of morphology. *Computational Linguistics*, 37:309–350.

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

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.

Karypis, G. (2003). Cluto - a clustering toolkit.

Kirchhoff, K., Bilmes, J. A., and Duh, K. (2008). Factored language models tutorial. Technical report, Department of Electrical Engineering, University of Washington, Seattle, Washington, USA.

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

Klakow, D. (1998). Log-linear interpolation of language models. In *Proceedings of the International Conference on Spoken Langauge Processing*, Sydney, Australia.

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, volume 1, pages 181–184.

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

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

Liu, Y. and 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*, pages 4921–4924.

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.

Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P., and Datta, K. (2007). Yass: Yet another suffix stripper. *ACM Transactions on Information Systems*, 25.

Maltese, G., Bravetti, P., Crepy, H., Grainger, B. J., Herzog, M., and 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*, pages 21–24. Eurospeech.

Oikonomidis, D. and Digalakis, V. (2003). Stem-based maximum entropy language models for inflectional languages. In *8th European Conference on Speech Communication and Technology*. ISCA.

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

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

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

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.

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.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228.

Rubenstein, H. and Goodenough, J. B. (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., and Kanerva, P. (2008). Permutations as a means to encode order in word space. *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 1300–1305.

Tam, Y. and Schultz, T. (2005). Dynamic language model adaptation using variational bayes inference. In *Proceedings of Interspeech*, pages 5–8.

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

Vaiciunas, A., Kaminskas, V., and 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., 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 and Language*, 25:440–461.

Whittaker, E. W. D. (2000). *Statistical Language Modelling for Automatic Speech Recognition of Russian and English*. PhD thesis, Cambridge University, Cambridge, MA, USA.

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

Xu, P. and Jelinek, F. (2007). Random forests and the data sparseness problem in language modeling. *Computer Speech and Language*, 21(1):105–152.

Yamamoto, H. and 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., and 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*, pages 71–74.

Zhao, Y. and Karypis, G. (2002). Criterion functions for document clustering: Experiments and analysis. Technical report, Department of Computer Science, University of Minnesota, Minneapolis.