SPRINGER BRIEFS IN ELECTRICAL AND COMPUTER ENGINEERING · SPEECH TECHNOLOGY

Gregor Donaj
Zdravko Kačič

Language Modeling for Automatic Speech Recognition of Inflective Languages

An Applications-Oriented Approach Using Lexical Data

Springer

SpringerBriefs in Electrical and Computer Engineering

Gregor Donaj • Zdravko Kačič

Language Modeling for Automatic Speech Recognition of Inflective Languages

An Applications-Oriented Approach Using Lexical Data

Springer

Gregor Donaj
Faculty of Electrical Engineering
and Computer Science
University of Maribor
Maribor, Slovenia

Zdravko Kačič
Faculty of Electrical Engineering
and Computer Science
University of Maribor
Maribor, Slovenia

ISSN 2191-8112 ISSN 2191-8120 (electronic)
SpringerBriefs in Electrical and Computer Engineering
ISBN 978-3-319-41605-2 ISBN 978-3-319-41607-6 (eBook)
DOI 10.1007/978-3-319-41607-6

Library of Congress Control Number: 2016946977

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Preface

In the few last years, we saw the rise of practical speech recognition applications, which work well in English and a few other languages. There is no doubt that this trend will continue and a more natural interaction between humans and technology will become part of our lives.

Language is one of the most important components of one's culture and identity. With English being the most present language in information technology, other languages face the danger of being diminished in this area. The development of speech recognition systems represents a potential risk for further diminishing languages, in which a sufficient speech recognition performance is harder to achieve. This does not consider languages spoken by very few people but also other languages spoken by many people, which are simply hard in speech recognition, like inflective languages, i.e., languages which due to the inflection of main parts of speech have a rich morphology and therefore the need for a much larger vocabulary in speech recognition.

This book gives an overview of inflective languages and their place in speech recognition. We describe the most important language features that are important for the development of a speech recognition system. This is then presented through the analysis of errors in the system and the development of language models and their inclusion in speech recognition systems, which specifically address the errors that are relevant for targeted applications.

Maribor, Slovenia
April 2016

Gregor Donaj
Zdravko Kačič

Contents

Chapter 1
Introduction

Abstract Language modeling is used in a variety of applications for the estimation of a word sequences probability of appearance in the modeled language. Language modeling has an important role in automatic speech recognition, where those estimates are used to distinguish between likely and unlikely hypotheses. We can make language models for different languages using the same techniques. However, we cannot expect that the models will be equally effective in different languages. Even in the same language we may need different language models based on the application for which they are intended. In this introduction chapter we briefly describe the reasons for the need of different language modeling techniques in different applications.

1.1 Speech Recognition

Speech is the most natural and simple way of communication between people. In the modern world we are more and more surrounded with technology that is not always understandable and easy to use. The typical way to interact with computer systems is to use keyboards, pointing devices, monitors and recently touchscreen displays. The development of speech technologies enables a more natural way of interaction with computers systems. Several technologies are needed for such an interaction: speech recognition, dialog systems and speech synthesis.

Another possible use for speech recognition systems is in a translation system, where spoken text is recognized, then fed into a machine translation system, which translates the text into some other language. Finally, the text is then converted back into speech by a speech synthesis system. A fast translation system without errors would help to overcome language barriers between people who have different mother languages and do not speak a common foreign language.

Speech recognition can also have other purposes like transcribing spoken text for the deaf or searching for key phrases in spoken text. In all applications the main purpose of a speech recognition system is to correctly recognize the spoken words and return them in text form. This is not easily achieved. Recognition errors, i.e. deleted, inserted, or substituted words, are common in speech recognition. However, in some applications they are more common. While the speech recognition with a

G. Donaj, Z. Kačič, *Language Modeling for Automatic Speech Recognition of Inflective Languages*, SpringerBriefs in Electrical and Computer Engineering,
DOI 10.1007/978-3-319-41607-6_1

small vocabulary of words, e.g. digits or a limited number of names, typically has a high accuracy rate, the recognition of continuous speech with a large vocabulary of 60,000 or more words is more prone to errors.

Errors can have different causes. Often acoustic factors may be the cause for a recognition error. Language models help to reduce error rates by assigning low scores to unlikely sentences. In large vocabulary applications we use statistical language models. They estimate a probability for a given sequence of words, based on how often this sequence or at least parts of it are found in a training corpus. For better estimates larger corpora are needed. Building large corpora (several 100 million words) is a time consuming and expensive work. The largest corpora usually represents a collection of many different texts on a variety different topics. The texts can also be gathered from different sources. Such general corpora are intended to give a reference representation of the language.

General corpora are used for building language models for speech recognition in a general domain, where we expect any possible sentence to appear in the spoken text. However, sometimes we may not wish for a general language model. In some applications we may expect only certain type of sentences to appear or at least we may expect a somewhat limited vocabulary. In such cases we want to have a specialized corpus, which is similar to what we expect in out application.

Building a specialized corpus is somewhat more complicated, because the texts have to be selected more carefully. Also, the amount of available texts is smaller. Such corpora can be build together with a speech database, which will be used for the target application. Such texts may represent the targeted domain better, since they are limited to this domain and do not contain a variety of dissimilar texts. Therefore, a language model build on such a corpus can be used to better estimate scores for given word sequences and might achieve better results if the corpus is either large enough or the domain is limited enough. The downside is that we need another corpus for each new domain or new specialized application.

Even before we build a language model, we must have a vocabulary of all the words we want to include in our model. A simple way of generating a large vocabulary (60,000 words or more) is to take a large corpus and calculate the frequencies of all the different words in the corpus. The most frequent words are then added to the vocabulary. This approach will give small amounts of out-of-vocabulary (OOV) words and therefore help to reduce word error rate. The larger the vocabulary is the smaller the OOV rate will be. However, even if we take all the different words of the corpus into a vocabulary we can not guarantee that in practice we will not encounter new words. In a specific domain, where only a small corpus is available, this approach could result in a vocabulary that is missing several words, which could later occur more often.

Inflective languages in which major parts of speech (nouns, adjectives, verbs) are inflected given their grammatical role. A typical example is the change in word endings when changing the case or number of nouns. Those languages have a more rich morphology compared to non-inflective languages. They present a greater challenge for speech recognition as they demand a larger vocabularies and larger learning corpora for language modeling to achieve a comparable performance.

1.2 Performance in Specific Applications

A speech recognizer is typically optimized to produce the lowest possible word error rate (WER). That is simply the ratio between the sum of all deletions, insertions and substitutions of words between two sentences (the true spoken text and the speech recognizers output) and the total number of words in the true spoken text. WER is automatically calculated and represent an objective measure for the performance of a speech recognition system. It is also the most widely used measure for performance.

Given a specific application for speech recognition we may find out that using WER is not the best way of estimating performance. This thought can be illustrated with a few hypothetical applications:

- Recognition of financial news. We expect many numbers and maybe names of companies. The names of the companies and other more technical terms may indicate the need for a special vocabulary. Also, a general corpus does not guarantee that all possible numerals will be in the corpus. Moreover, in such an application the correct recognition of the numbers will be probably more crucial than the recognition of other words. A score that considers this would be more suitable for performance evaluation.
- Recognition of proper names. This example is similar to the previous one. There are many proper names and a corpus most probably will not contain all of them. If we have a spoken text which contains many proper names and correct recognition of the names is more crucial than the recognition of other words, it is again important to differentiate between errors.
- Keyword spotting. In this application we want to search through an audio database looking for certain keywords or key phrases. In this case we could speak of binary classification, where we say if the speech segment contains the keyword or not. Errors are then false positive and false negative errors. If the purpose is to identify speech segments that will be later human checked, we could say that a false negative error has a greater effect, since false positive errors can be later manually dismissed. We could modify the language model in respect to these keywords to increase their probabilities inside the model.
- Speech translation. The final output of a speech recognizer in combination with a machine translation system and a speech synthesis system is the spoken sentence in a new language. The score can then be measured by metrics usually used in machine translation. In this case we would have to compare the final output to a reference translation of the true spoken sentence. We can not definitely say that a lower WER will produce a corresponding lower error rate in the final translation. It would be also interesting to know if different recognition errors have a different effect on the final score.

In the later chapters we will present techniques for an application specific performance estimation of speech recognition and how these estimations can be used to adapt language models with the goal of a better speech recognition system for a given application.

1.3 Organization of the Book

In Chap. 2 basic concepts of speech recognition are presented. We will also discuss inflective languages, why they present a challenge for speech recognition and some established methods for speech recognition in inflective languages. Morphosyntactic description (MSD) tagging for inflective languages is presented as well as it is used in further chapters.

In Chap. 3 we first describe established metrics for measuring the accuracy of a speech recognition system and then introduce a new alignment method based on an extension of the well-known and used Levenshtein distance. This new metric can then be used for a better analysis of error rates An example of such an analysis in Slovene is also presented in this chapter.

In Chap. 4 aspects of speech recognition with regard to inflective languages, measures of performance, and error sources for target applications are discussed. This is further illustrated with short description on how these factors apply to sample applications. Also, a procedure for language modeling for specific target application is described. This procedure is then used with a full example in Chap. 5.

Chapter 2
Speech Recognition in Inflective Languages

Abstract In this chapter basic concepts of speech recognition are presented. Acoustic processing, acoustic modeling and search algorithms are briefly described. A more detailed explanation is given on language modeling. Afterwards some features of inflective languages are described and how these features are important in the process of designing speech recognition systems. First inflective languages are discussed in general, then Slovene as an example is discussed in more detail. Next some typical methods to overcome the difficulties of speech recognition in inflective languages and improve speech recognition accuracy are described. These are the enlargement of the vocabulary, the use of sub-word language models and other more sophisticated language models. The last part of the chapter discusses morphosyntactic description tagging in inflective languages that will be used in further chapters. This chapter does not give a comprehensive overview of speech recognition, solely basic descriptions and some more information that is necessary to understand the content of further chapters are given.

2.1 Basic Structure of a Speech Recognition System

The basic structure of an automatic speech recognition (ASR) system is shown in Fig. 2.1. After a sound signal which contains a spoken sentence is recorder, it is first acoustically processed. The result of acoustic processing is a set of features—a so-called feature vector for any given short time period of the speech signal. Feature vectors are then fed into the search algorithm, which is the central part of a speech recognizer. The search algorithm uses a pronunciation dictionary, acoustic models and a language model to determine the best hypothesis of the spoken sentence. These basic parts of a speech recognizer are described in the following sections.

2.2 Acoustic Processing and Feature Extraction

The recorded audio signal is first divided into several segments. This is done based on segmentation or voice activity detection algorithms (VAD) [42, 48]. Only some of

G. Donaj, Z. Kačič, *Language Modeling for Automatic Speech Recognition of Inflective Languages*, SpringerBriefs in Electrical and Computer Engineering,
DOI 10.1007/978-3-319-41607-6_2

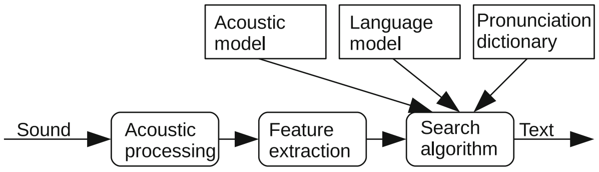


Fig. 2.1 Basic architecture of a typical speech recognition system

those segments will actually contain spoken text and the task of VAD algorithms is to identify such segments. We call them speech segments as opposed to non-speech segments.

In the acoustical processing [11] the speech signal of a segment is framed. This means that only a small part of the overall signal is processed at once. Frames are typically very short, around 32 ms and two consecutive frames overlap each other by approximately one third to one half. The signal of each frame is then processed by a windowing function, e.g. the Hamming window [12], resulting in smaller amplitude at the beginning and the end of each window. The windowed signal is the basis for the calculation of the feature vector [10] in the course of which a Fast Fourier Transformation algorithm is used to transform the data into the frequency domain, where all further processing is done.

The two most widely used types of features are Mel-Frequency-Cepstrall Coefficients (MFCC) [5] and Perceptual Linear Prediction Coefficients (PLP) [16]. Both types are based on characteristic of the human hearing and have been found to be effective in speech recognition. Perceptual linear prediction was introduced as an improvement to linear prediction.

The PLP features take into account three features of human hearing:

- critical-band spectral resolution,
- equal-loudness preemphasis and
- intensity-loudness power law.

The calculation of MFCC features also considers these characteristics of human hearing although they are differently implemented. With both methods a certain number of coefficient is considered for speech recognition. For more information on the calculation of PLP and MFCC coefficients the reader can look at [5, 16] or the manuals of a specific speech recognition systems, e.g. the HTK Book [53].

The next step is to construct feature vectors. This is done by taking the obtained coefficients and adding delta and delta-delta coefficient. Those are calculated based on the difference between the same coefficients in consecutive frames, sometimes also considering more distant frames. They therefore present the first and second derivatives of the basic features [11]. The speech signal is now presented as a sequence of feature vectors, which contain the information needed for speech recognition.

An example: We have an input signal sampled at 16,000 samples per second in 16-bit quantization. This quality is found to be enough for speech recognition. The information rate is therefore 32,000 bates per second. We have selected MFCC as features and decided to use 13 of them and calculate them with 16-bit precision. The first feature represents the energy of the signal while the others represent higher frequencies. We added delta and delta-delta coefficients to the feature vector. The total number of features in the vector is now 39. We then calculate such a vector for every frame of the audio signal. With a typical frame spacing of 10 ms, this results in 100 frames per second or 3900 features per second or 7800 bytes of data per second. We see that not only feature vectors represent well the characteristic of speech, they are also a more compact representation of speech then raw audio data.

2.3 Acoustic Modeling

An acoustic model is a mathematical model used to estimate the probability that a given audio signal corresponds to a certain phoneme or an entire word. The prevalent types of acoustic models are hidden Markov models (HMM) [11, 41] and weighted finite state transducers (WFST) [35].

HMMs are models with a hidden Markov chain processes and a visible output sequence of symbols. Each state in the Markov chain has an associated probability distribution of output symbols and every time the hidden process goes into a new state, an output symbol is generated based on the distribution for this state. HMMs also include transition probabilities from one state to each other state in the model. In ASR however, models are used which have a linear distribution of states and non-zero probabilities only for transition from one state into itself or some near further state. In speech recognition the output symbols are feature vectors and the hidden states of the chain can represent words or sentences. However, in large vocabulary applications several states in sequence are used to represent triphones [11, 34]—phonemes that have been further classified based on the preceding and the following phoneme as this has been found to improve the speech recognition accuracy. The goal is now to find the sequence of hidden states, which would most likely produce the observed sequence of feature vectors. Since any sequence of states represents a sequence of speech parts, a hypothesis of what was spoken can then be formed by the search algorithm.

WFSTs [35] are finite automatons, which map a sequence of input symbols onto a sequence of output symbols. The state transitions of this automaton are labeled with input and output transitions. Each mapping has also an appropriate weight. The goal is now to find the transitions with the lowest total weight. HMMs, dictionaries and also language models can be represented as WFST. Therefore, a composition of several transducers into one can be used to include all knowledge sources for a speech recognition system into one model.

Both kinds of acoustic models have to be trained—their parameters have to be determined. In the case of a HMM the parameters are transition probabilities and the parameters of output distributions for each state. For the parameter estimation the Baum-Welch algorithm is often used [11], which maximizes the maximum likelihood estimator (MLE), although other estimator can be used in discriminative acoustic modeling [24]. In case of a WFSTs the parameters are weights of all transitions. For this purpose a set of speech recordings with carefully prepared phonetic transcriptions is used.

To train acoustic models with sub-word units (triphones) also a dictionary with phonetic transcriptions of each word is necessary. This dictionary is later also used by the search algorithm.

The above mentioned set is called the training set. It should be as large as possible as we need many examples of triphones to efficiently estimate the models parameters.

2.4 Language Modeling

In general we know two basic kinds of language models [20]. The first is based on fixed rules that are often manually defined. We call them formal grammars [26]. They can describe a sequence of word as either right or wrong. They can be used in applications with a small vocabulary and where we expect a limited number of different possible word sequences. In large vocabulary continuous speech recognition (LVCSR) this sort of language models can not be used since it is not feasible to implement all possible rules into a formal grammar.

The second type of language models are called stochastic models or statistically based models and are used to estimate the probability that the given word sequence will appear in the modeled language. We say that the model in trained on this language. Most widely word based *n*-gram models are used [23, 52] or other models based on them [56].

2.4.1 Word Based n-Gram Models

Let $W = (w_1, w_2, \ldots, w_N)$ be a sequence of words. For simplicity we just call it a sentence. The task of the language model is to calculate a probability for this sentence:

$$P(W) = P(w_1, w_2, \ldots, w_N). \tag{2.1}$$

Like with all statistical models we need a sample of the language to train the model. Large digital collections of text from a language, called corpora, are used for training. A simple method for determining the probabilities would be to count all the different sentences in the corpus. Because of the large number of words in a language and consequently very large number of possible sentences, such a method is not feasible since we neither have a sufficient amount of data nor current hardware would be able to process and use such amounts of data.

To make language modeling feasible at all we first have to make two assumptions. The first one is the Markov assumption, that says that the probability of the occurrence of a word in a sentence depends only on a certain number of the preceding words. Using the chain rule for probabilities we can write the probability from the previous equation as

$$P(w_1, w_2, \ldots, w_N) = \prod_{i=1}^{N} P(w_i|w_1, \ldots, w_{i-1}). \tag{2.2}$$

Using the Markov assumption we can say that the probability depends on the previous $n-1$ words:

$$P(w_i|w_1, \ldots, w_{i-1}) = P(w_i|w_{i-n+1}, \ldots, w_{i-1}). \tag{2.3}$$

Combining these two equations we get

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-n+1}, \ldots, w_{i-1}). \tag{2.4}$$

This equation is later the basis for determining the probability of a sentence. A sequence of n words is called an n-gram. Hence the name for this type of model. Basically an n-gram model is an $n-1$ order Markov chain. In the context of language models we define the model order as the number of word in the n-gram. We therefore know unigram models (order 1), bigram models (order 2), trigram models (order 3), and higher order models. Names of higher order models are often written with a digit, e.g. 4-gram model, 5-gram model etc.

The second assumption says that the probabilities in the right hand side of Eq. (2.4) are independent of i. Simply put this means that the probability of the occurrence of a n-gram inside a sentence is independent of this n-grams position inside the sentence (whether it occurs at the beginning of a sentence or after any number of preceding words).

Before we can use a model, we have to train it. A simple method is to use the maximum likelihood MLE estimator. We calculate the probabilities for each possible n-gram with the equation:

$$P_{MLE}(w_i|w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1})}. \tag{2.5}$$

Here $C(\cdot)$ is a simple count function that tells us how many times its argument was found in the training corpus. The total number of all possible n-grams is much smaller than the number of all possible sentences. Therefore, the before mentioned assumptions have made it possible to construct a usable language model. Nonetheless, while using the model some n-grams may occur, that have not been seen in the training corpus, but are still plausible to occur. To prevent the language model to assign a probability 0 to such an n-gram, and therefore to any hypothesis containing it, we use smoothing techniques [7] and back-off algorithms [25].

An important point when building the language models is the selection of the vocabulary. Larger vocabularies result in larger language models, which usually perform slower in speech recognition since the search for an n-gram inside the models takes more computational time. However, smaller vocabularies result in a larger number of words being outside the vocabulary. This later results in recognition errors. A larger vocabulary on the other hand has a negative effect on recognition speed. We will discuss the vocabulary size further in later sections.

2.4.2 Smoothing and Back-Off

The most widely used n-gram models are bigram and trigram models. Let us assume a reasonably large vocabulary consisting of 100,000 words. In a language model a parameter is needed for every possible n-gram. A bigram model would need 10^{10} parameters and a trigram model even 10^{15} parameters to account probabilities for all possible n-grams. In a typical training corpus most of those do not appear, and many do appear once or only a few times, which is not suitable for a good estimation of probabilities.

Smoothing is used to change the probabilities of n-grams in a statistical model. The simplest way of smoothing is to artificially increase the number of all n-grams by 1, whether they occurred in the training corpus or not. By this we achieve the main purpose of smoothing, that is to have all probabilities greater than 0. We then modify the above mentioned equation for estimating probabilities into the following form:

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) = \frac{C(w_{i-n+1},\ldots,w_i)+1}{C(w_{i-n+1},\ldots,w_{i-1})+1}. \tag{2.6}$$

This simple smoothing technique turned out to be inappropriate for language models, as it did not produce better results. More complex smoothing techniques were developed for speech recognition. Examples are Good-Turing, Witten-Bell, Knesser-Ney and modified Knesser-Ney [7]. Often language modeling toolkits like SRILM and IRSTLM have several of those smoothing techniques are implemented into them.

When we encounter an n-gram, which was not or rarely seen in the training corpus we would not have a good probability estimation. The basic idea of the back-off algorithm [25] is to go to a lover order model. This means we will add probabilities of lower order n-grams into the model and while estimating sentences use these probabilities if the corresponding highest order n-grams are not in the model.

We need a number of occurrences that will serve as a limit when to use back-off and when not at each model order: N_n. We can express this the back-off algorithm with this equation:

$$P(w_i|w_{i-n+1},\dots,w_{i-1}) =$$
$$= \begin{cases} d_{N(w_i,\dots,w_{i-1})}P(w_i|w_{i-n+1},\dots,w_{i-1}) & ; C(w_{i-n+1},\dots,w_i) > N_n \\ \alpha(w_{i-n+1},\dots,w_{i-1})P_{BO}(w_i|w_{i-n+2},\dots,w_{i-1}) & ; \text{otherwise} \end{cases} . \tag{2.7}$$

We also added the function α and the discount number d_N, which are necessary to obtain a model with the total probability of all sentences equal to 1. Back-off can be used several times, e.g. using a 4-gram model, we can back-off to trigrams, then bigrams and finally unigrams. On every step we discard the most distant word.

2.4.3 Perplexity

In order to evaluate if a language model adequately represents a language or to compare two language models we need a measure for language models. In an application we could obviously use the final performance, e.g. the accuracy rate in a speech recognition system. This is however time consuming and we need a complete and working application.

The most widely used measure to evaluate language models on their own is perplexity [20]. Its definition comes from information theory. We can consider the language to be an information source which generates a sequence of output symbols, in this case words. After the model is trained on a training corpus, we need an additional text to calculate the perplexity. The probability of a sentence W in this test set can be calculated by Eq. (2.4). The necessary probabilities of individual words is obtained from the language model. The probability for the whole test set $\mathscr{W}$ is simply the product of the probabilities for all sentences:

$$P(\mathscr{W}) = \prod_{i=1}^{T} P(W_i). \tag{2.8}$$

For a given model P and test set $\mathscr{W}$ we calculate the cross-entropy as

$$H_P(\mathscr{W}) = -\frac{1}{N_T}\log_2 P(\mathscr{W}), \tag{2.9}$$

where N is the number of all words in the test set. The perplexity PPL is then defined as the inverse of the geometric mean of all probabilities of the words in the test set. It can be now calculated using the cross-entropy:

$$PPL_P(\mathscr{W}) = 2^{H_P(\mathscr{W})}. \tag{2.10}$$

Lower perplexities mean that the probabilities for all words were greater, thus showing us that the language model fits the test set better than a model with a high perplexity. It has been shown that language model with a lower perplexity perform better in speech recognition than models with a higher perplexity [7, 29].

It should be noted that two language models can be compared by their perplexities if both models are build upon the same vocabulary and both perplexities are calculated on the same test set. The model order plays a significant role in the perplexity of a model. Higher order model have a lower perplexity the lower order models. However, increasing the models order for getting a better model results in a larger model and at some point seizes to improve the model.

2.5 Search Algorithms

The search algorithm [3] is the central part of a speech recognition system. Its task is to determiner the word sequence ($\hat{W} = w_1, w_2, \ldots, w_n$) of unknown length n over all possible word sequences $\mathscr{W}$, which maximizes the probability of the language and acoustic model:

$$\hat{W} = \arg\max_{W \in \mathscr{W}} \{P(O|W)P(W)\}, \tag{2.11}$$

where $O = (o_1, o_2, \ldots, o_T)$ is a given sequence of feature vectors and W can be any sequence consisting of words from the vocabulary. The factor $P(O|W)$ is the acoustic probability that the word sequence fits the observed feature vector sequence. This probability is determined using the acoustic model and the pronunciation dictionary. In the recognition process the acoustic models of triphones are added together to account as models for words. These models are then added together to accounted as models for the whole sentence. This happens on-line while executing the search algorithm. The second factor $P(W)$ is the probability of the word sequence W calculated using the language model.

In implementations of a speech recognizer the above equation will be for practical reasons implemented using logarithms of probabilities (log-probabilities) as using regular probabilities would lead to underflow [53]. Consequently, the product of probabilities in converted to a sum of log-probabilities.

Often a heuristic constant is added to the search algorithms which differently weights the probabilities of acoustic and language models. The weight for the acoustic model can be fixed to 1, so that only one weight need to be added. It is called the language model weight (α). We also add a heuristic factor called the word insertion penalty (β) that lowers the probability of longer hypotheses given the number of words (N). This has been found to be necessary as search algorithms seem to prefer several short words over a similarly sounding long word [38] . The equation to be solved by the search algorithm so becomes:

$$\hat{W} = \underset{W \in \mathscr{W}}{\arg\max}\{\log P(O|W) + \alpha \log P(W) + \beta N\}. \tag{2.12}$$

The search algorithm considers several possible sentences and gives them a score based on the acoustic model and language model probabilities. These sentences are here called hypotheses. The search algorithms always keeps track of a limited number of partial hypotheses during recognition. The final output of the search algorithm can simply be the hypothesis with the best score at the end of the recognition of the current speech segment. In other cases the output can be a list, a word lattice, or some other representation of several hypotheses. Such a list is then suitable for further processing.

Search algorithms can be classified into different types, based on several different criteria. Most classification criteria are not important for the content of this book. A detailed classification is presented in [3]. However, we shall still mention that search algorithms can be classified into one-pass algorithms and more-pass (usually two-pass) algorithms [1, 56]. In a one-pass algorithm all knowledge sources are used in one process and the best hypothesis is determined.

In a two or more-pass algorithm first a search algorithm is executed that produces a list of hypotheses. This list is later refined with additional algorithms that consider only the hypotheses in this list but use more knowledge sources, e.g. additional language models. We will use this approach in later chapters.

2.6 Features of Inflective Languages

The above mentioned speech recognition techniques are found to be working well for recognition of English. While using them on some other languages it has been found that performance levels are often lower [40, 43, 54]. One of the reasons for this difference if the complex nature of those languages.

When comparing inflective and non-inflective languages in the context of language modeling and speech recognition the most important feature of inflective languages is their morphological richness [22, 39, 40, 47, 54]. This results in a much higher number of different word forms than in non-inflective languages. For example the form of a verb depends on grammatical categories such as gender, person, and number.

Slavic languages fall in the category of inflective languages. They have a similar morphology and syntax. Examples are Russian, Polish, Czech, Croatian, Serbian, and Slovene. Another category of morphologically rich languages are agglutinative language in which the words are composed of stems and possibly several affixes. Other demanding languages for speech recognition are German [36, 37], Turkish [44] Finnish [17, 18], French [21] and Arabic [27].

In inflective languages words can be categorized either in inflective parts of speech on non-inflective parts of speech. Inflective parts of speech in most inflective languages are nouns, adjectives, verbs and in some cases adverbs. The word-

Table 2.1 Word forms of the Slovene verb *delati* (*to work*)

	Singular	Dual	Plural
First person	delam	deleva	delamo
Second person	delaš	delata	delate
Third person	dela	delata	delajo

formation of inflective words can be a rather complicated process. Inflective words change their endings with declension and conjugation. In some cases not only the endings changes with inflection but also a letter in the stem.

Different word forms in inflective languages can be recognized by different word endings. An example for the Slovene word "delati" (to work) is in Table 2.1. The word is written in present form in three persons and three numbers (singular, dual, and plural). We see eight different forms. There are further differences for infinitive, conditional, and imperative forms. For example, the Slovene verb "biti" (to be) has 39 different word forms.

The large number of word form causes two problems in speech recognition. We need larger vocabularies to cover a language. Given a certain vocabulary we can use large corpora to estimate how many percent of a language is covered with this vocabulary. We do not just consider different words, but also how often their appear. If a large proportion of the corpus (say 99 %) is covered with a given vocabulary we can consider this vocabulary to be suitable for LVCSR in a general domain.

The use of larger vocabularies implies a larger number of acoustically similar words and therefore causes greater chances for them to be misrecognized—substituted with each other.

Also, a greater vocabulary demands a greater training corpus for training the language model. A typical size for an English dictionary for large vocabulary speech recognition is about 60,000 words. It has been estimated that for the same coverage of an inflective language the vocabulary should be seven to ten times bigger than the vocabulary of English [43, 54]. Still, there will be out of vocabulary words. For example the Russian Zaliznyak's grammatical dictionary contains 160,000 basic words in 3.7 million word forms [54].

Another feature of inflective languages is a relative free word order in sentences. In English the order Subject-Verb-Object is very rigid and to change the order of words in a sentence might change the meaning of the sentence, due to the fact that the position of a word in a sentence is important in determining its role. Since in inflective language the role of a word is determined by the word form, its place inside the sentence has a smaller role. Thus, a free word order is possible. Taking a sentence and change the word order will not change the meaning of the sentence.

Since the word order is free, there is a larger number of sentence forms to express the same meaning. This is a difficulty for using n-gram language models since they specifically model sequences of words. Consequently, we again need a larger corpus to adequately train a language model.

Numbers are again a special issue. Language models are build on written text where numbers are typically written with digits and not with words. In a non-inflective language this may not be so much of an issue, since we could simply convert the digits into words. In inflective languages we must know the correct word form. Also, depending on the language, there could be several word forms for each of the numbers we may encounter in speech.

2.6.1 Features of Slovene

In this and following chapters we will present some example results from speech recognition in Slovene. Therefore, we will also write some extra details about it so that the reader will understand the results. While it shares most characteristics with other Slavic languages, there are some characteristics rarely found in other languages. Here are some basic grammatical characteristics important in the context of inflection [50]:

- Inflective parts of speech: noun, adjective, and verb. Nouns and adjectives are inflected with regard to case (declension) and number; adjectives also with gender and some with degree. Verbs are inflected with regard to person (conjugation), number, tense and mood.
- Non-inflective parts of speech: predicative (with inflective exceptions), adverb, preposition, particle, conjunction, and interjection.
- Inflective parts of speech are divided into stems and endings. The stem gives the word a semantic meaning while the ending indicates grammatical features.
- Stems remain mostly unchanged while a word is inflected.
- Three genders: masculine, feminine and neuter. The gender of a noun is determined by the word ending, resulting in many objects having a masculine or feminine gender as opposed to some other languages where lifeless objects are mostly neuter.
- Six cases: nominative, genitive, dative, accusative, locative, and instrumental.
- Three numbers: singular, dual, and plural. While most languages have only singular and plural with residual traces of dual (e.g. in English the word *both*), Slovene has dual forms for verbs, nouns, adjectives and pronouns. This feature is shared with only a handful of living languages.
- Three persons: first, second, and third.
- Four tenses: present, future, perfect and pluperfect.

Slovene has a clear morphological structure, in which words have the form stem-ending. The stem represents the basic concept or lexical meaning and the ending represents the words grammatical properties like gender, case, and number.

In declension and conjugation mostly endings change. There are some exception when also a letter in stem is changed. This usually happens in diminutive forms. This further increases the number of possible word forms.

Another feature of Slovene and other inflective language is the grammatical matching between words in a sentence. An adjective describing a noun matches it in gender, case and number. Verbs match with the subject of the sentence in gender, number and person. When we use verbs in different tenses results we also need auxiliary verbs (mostly forms of *to be*), whose form depends on the used case.

In language and speech technology Slovene is considered a rather poorly researched language with missing language resources compared to other European languages [30]. Still some large corpora are available like the 600 million words FidaPLUS corpus [2] and the 1.2 billion words GigaFIDA corpus [31], as well as there are some speech resources including a broadcast news system suitable for research in LVCSR [55].

2.7 Language Modeling for Inflective Languages

In the beginnings of speech recognition the English language has been mostly studied where methods and algorithms have performed well. Later these methods were also applied to different languages with different results. In general the recognition of inflective languages turned out to be harder than that of non-inflective languages. In this section we will present some techniques that have been applied to improve the speech recognition on inflective languages.

2.7.1 Vocabulary Size, Language Model Order and Corpus Size

A simple method to improve speech recognition in inflective languages is to increase the size of the vocabulary. The vocabulary is built on a training corpus. Usually the most frequent words from the corpus are included in the vocabulary in an attempt to maximally increase the coverage of the language with a certain vocabulary size. Less frequent words are not in the vocabulary and if they appear in a sentence in the test set of a speech recognizer or in later in a practical application they can not be correctly recognized.

If we count all the words in the test set that are not in the vocabulary and divide the result with the total number of words in the set, we calculate the out-of-vocabulary (OOV) rate. A larger vocabulary will result in a smaller OOV rate.

Figure 2.2 shows an example of the dependence of OOV rate given the vocabulary size. The corpus used to build the vocabulary in this example is the Slovene FidaPLUS corpus. Vocabulary sizes reach from 60,000 to 300,000 of the most frequent words. The OOV rates are calculated on the test set of the Slovene Broadcast News (BNSI) speech database. We see a decrease of the OOV rate from 6.84 % down to 1.02 %.

We used the same vocabularies to build bigram language models on the FidaPLUS corpus. These language models were used in a speech recognition system on the BNSI database. Figure 2.3 shows the dependence of word error rate (WER) given the vocabulary size. A comparison of Figs. 2.2 and 2.3 (both figures are drawn

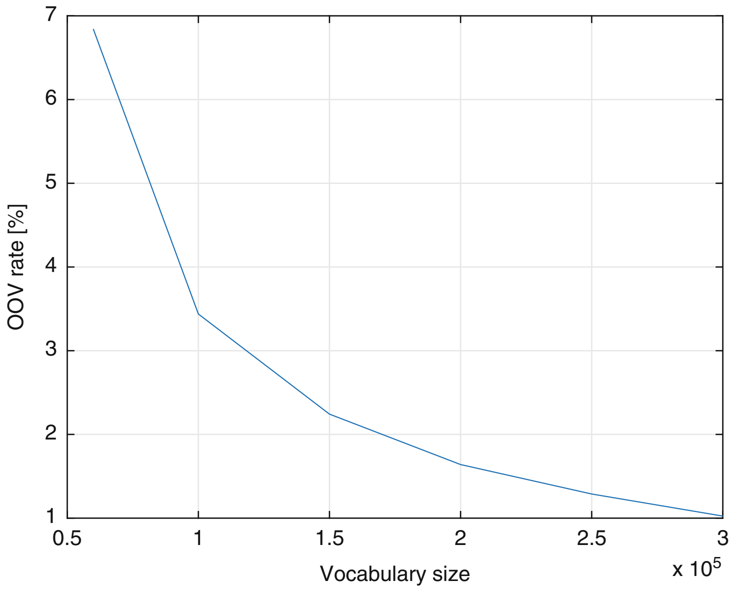


Fig. 2.2 OOV rates given vocabulary size

in the same scale) shows that the WER decreases similar to OOV rate. We conclude that increasing the size of the vocabulary removes errors that are caused by OOV words. Similar results can be found for other Slavic languages [39, 40, 54]. However, there are differences in the graphs at larger vocabularies, where the OOV rate falls faster than WER. We can assume that while the increase in vocabulary size indeed removes errors that are due to OOV words, the higher chance for word substitution introduces new errors.

Given the fact that the OOV rate at a vocabulary of 300,000 words in only about 1 % and that WER decreases more slowly, is does not seem reasonable to further increase vocabulary size, especially given the fact that a larger number of errors seem not to be due to OOV words.

The order of the language models also plays an important role. Higher order models have a lower perplexity and are expected to give better results since they take into account a longer word history (sequence of preceding words). It is also expected that larger training corpora are better, since from more data we can better estimate probabilities.

Table 2.2 shows WER results while using bigram and trigram models with the whole corpus and one tenth of it. In all cases we again see a WER decrease of about 4–5 % when the vocabulary size is increased from 60,000 (60k) to 300,000 (300k) words. Comparing results obtained with bigram and trigrams models shows a WER reduction of about 3–4 %, but only when the whole corpus is used. Using only 1/10 of the corpus, the resulting decrease in WER is only about 2 %.

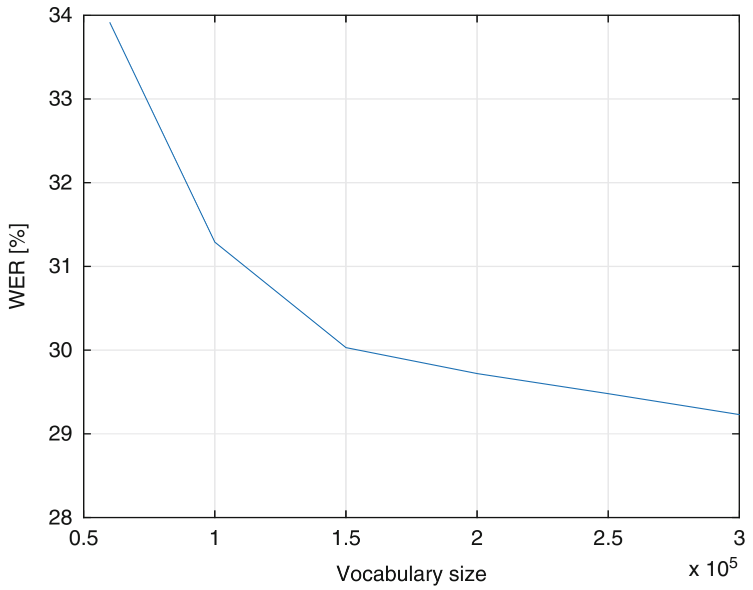


Fig. 2.3 WER rates given vocabulary size

Table 2.2 WER obtained with different sized vocabularies, different orders of language models and different training corpora sizes

Vocabulary size	WER [%]			
	Whole corpus		1/10 Corpus	
	Bigram LM	Trigram LM	Bigram LM	Trigram LM
60k	33.91	30.77	35.95	34.20
300k	29.23	25.67	31.89	30.10

From the results in Table 2.2 we can conclude that all three discussed factors—vocabulary size, language model order and training corpus size—have a significant impact on WER in speech recognition. Furthermore, we see that the benefit of using a high order model increase with corpus size. While the use of a 4-gram or even 5-gram language model would not decrease WER using one corpus, they might if a larger corpus would be available.

2.7.2 *Subword Based Models*

When we look at the word formation rules of a morphologically rich language we see that based on these rules we can also take the words back apart given their grammatical features. Taking words apart means we have to deal with sub-word units like morphemes. We can expect the number of morphemes to be smaller that the number of whole words to cover a language.

An often used approach for language modeling of morphologically complex languages is the use of subword units [19, 43]. In this approach words are divided into smaller parts. In Arabic and other languages an morpheme-based language model [27] can be applied in speech recognition. In Slavic languages a natural way is to break the words into stems and endings [43].

Based on the language we have to decide how to split words. This can be done in different ways:

- stem + ending
- stem + morpheme + morpheme + …
- morpheme + morpheme + …
- syllable + syllable + …
- no splitting (short words).

A morphological analyzer is needed to perform the separation of words into smaller units. The implemented algorithms can be rule-based on data-driven.

The selection of subword units can have several effects on the speech recognition system. We must also consider which splitting is appropriate for which language. In spoken language, shorter unit contain less acoustical information and thus can be more easily substituted [43]. The selection of the units also has an impact on the search space and therefore, affects the selection of the search algorithm. The use of subword units also open the probability that the search algorithm will construct words from smaller units that are supposed to be used together and in doing so generating false words.

Let us express a word as a stem and ending: $w_i = \{s_i, e_i\}$. The term in the language model probability in Eq. (2.4) can now be written using pairs of stems and endings:

$$P(w_i|w_{i-n+1}, \ldots, w_{i-1}) = P(\{s_i, e_i\}|\{s_{i-n+1}, e_{i-n+1}\}, \ldots, \{s_{i-1}, e_{i-1}\}). \tag{2.13}$$

This can be expressed as the product of two probabilities:

$$P(s_i|\{s_{i-n+1}, e_{i-n+1}\}, \ldots, \{s_{i-1}, e_{i-1}\}) \tag{2.14}$$

and

$$P(e_i|\{s_{i-n+1}, e_{i-n+1}\}, \ldots, \{s_{i-1}, e_{i-1}\}). \tag{2.15}$$

The goal now is to build those two language models for these two probabilities. A simple method is to build simply n-gram models for stems and endings separately. However, in one o the two equation we must also consider the other subword unit of the current word. Other, more sophisticated methods for modeling subword units and using them in speech recognition have been presented over the years.

Many researchers have compared the recognition accuracy of speech recognizers while using word-based and sub-word based setups. They show that using subword unit has improved the recognition accuracy. This is somewhat expected given the fact that using subword units decreases the OOV rate. Research has also shown that the benefit of sub-word unit decreases when using larger vocabularies. It is therefore questionable if there is any benefit in simply dividing the words into smaller units.

A problem that has often been pointed out while working with subword units is the necessary order of the language model. Since the units are now shorter than a word, an n-gram language model, does not contain sequences of n words, but sequences of n morphemes. To cover the same amount of a words context, the order of the language model has to be higher. In the case of stem-ending based units, the order has to be twice as high. A good overview of sub-word based research with further reference can be found in [19].

2.7.3 Factored Language Models

Factored language models have been introduces with the purpose of improving speech recognition accuracy for Arabic [4, 6, 27, 28], but there are still usable for other languages.

The term factor refers to a words feature. This can be the word itself, the prefix, suffix, ending, any grammatical feature or the words lemma. A word is then presented as a collection of K factors:

$$w_i = \{f_i^1, f_i^2, \dots, f_i^K\} = \{f_i^{1:K}\}. \tag{2.16}$$

In a traditional word based language model a words probability is estimated given the preceding words. In a factored language model the probabilities are estimated given the factors of the preceding word. This can be expressed as

$$P(f_i^{1:K}) = P(f_i^{1:K} | f_{i-n+1}^{1:K}, \dots, f_{i-1}^{1:K}). \tag{2.17}$$

As we want to calculate the probability for several factors, we can use the chain rule to transform the above equation into

$$P(f_i^{1:K}) = \prod_{k=1}^{K} P(f_i^k | f_i^{1:k-1}, f_{i-n+1}^{1:K}, \dots, f_{i-1}^{1:K}), \tag{2.18}$$

where we first calculate the probability of the first factor given all factors of the preceding $n-1$ words. The probabilities of the other factors are calculated given the factors of the preceding and the preceding factors of the current word.

There a two main problems in building factored language models. The first one is to define the factors. Based on grammatical knowledge, a morphological analyzer or other knowledge sources we can define a wide variety of features for a word. The question remains which of those to select as the factors in the language model. Suitable factors are those which have a statistical significance for the following words and would therefore be helpful in language modeling. Two approaches are possible. The first is a data-driven approach where an algorithm determines which factors give the best results. The second approach is the manual selection of the factors based on grammatical knowledge. For example since there is a match in the gender, case and number of adjectives and nouns in inflective languages, it is reasonable to select those features as factors.

In Eq. (2.17) we see that the probabilities of words are estimated on a possibly large set of factors. Like in traditional word based language models, we can expect combination that will not appear in the training corpus. Therefore smoothing techniques and a back-off algorithm should also be applied.

The second problem of factored language models is the selection of the back-off path. The back-off path is a sequence of sets of factors that are used in the probability estimation. In word-based models the selection if rather clear. At first we consider all words, i.e. the highest order n-grams. At every next step we discard the most distant word. In factored language models this path is not so clear. We have several factors that have the same distance from the current word and we need to know which to discard first. Also, it may not be the best to discard a factor from the most distant word, but rather a less useful factor from a closer word.

Factored language models are not as well-studied as other models, but there is some research that show promising results. In [28] the authors introduced factored language models and experimented with them by calculating perplexities on Arabic. Similar results can be found for English [6]. Factored language models have also been used in Turkish, Amharic, German, and other languages including Slovene [8].

Factored language models can be viewed as a broad generalization of word-based language models. In the end we can express many other language models as special cases of factored language models.

When we introduce stems and ending as factors, the probabilities in Eqs. (2.14) and (2.15) can easily be expressed in terms of factored language models.

2.7.4 Other Methods

When using a statistical language model, often used phrases can be added in the form of a regular grammar. The resulting language model profits from the included phrases as from the statistical n-gram probabilities [26].

In some cases new language models can be build by interpolating other language models. In this case the resulting probability of an n-gram in the final language model in a weighted combination of the probabilities of the same n-gram in the initial language models. An interpolation also requires a data set to find optimal values for the weights of individual models. Interpolation can also be used in hierarchical structures of language models [32].

We can build a language model from a general corpus, on language model from in-domain written data, one model from general spoken data, and one model from in-domain spoken data. General corpora are larger and therefore can be better used to estimate probabilities. In-domain data is smaller but can contain special phrases not so much found in a general corpus but useful for the application. Spoken language differs from written language. A language model build with transcriptions of spoken language can be better at modeling the specifics of spoken language like word omissions, repetitions, fillers and other.

Class based language models are a special case in which word are classified into several different classes. The classification can be rule based or data driven. The probability of a n-gram is the calculated using the corresponding classes. There a several different method to accomplish this. Usually a probability of the sequence of the classes and then the probability of the word given the current class are used. Whittaker and Woodland [52] presented a class based system for English and Russian and compared results to tradition word based models. They have achieved performance improvement for both languages.

Geutner et al. [13] has presented a two-pass algorithm which adaptively changes the vocabulary, where in the first pass the vocabulary is adapted based on the current speech segment. The adapted vocabulary is then used in the second pass.

For language modeling also decision trees and random forests can be used. They can be successfully implemented into language models for inflective languages. It was shown that they can outperform trigram models in small scale applications [49].

2.8 Morphosyntactic Description Tagging

Some of the methods mentioned in the previous sections use morphological information in language modeling. To use this information we must have this morphological information not only in the training data but we must also have a system that will add this information to the recognition hypotheses. For this purpose morphosyntactic description (MSD) tagging is employed [15, 21, 46].

In MSD tagging each word is supplemented with tags containing information about part of speech and other grammatical information. Like speech recognition, MSD tagging is also a more challenging task in morphologically rich languages and is also more prone to errors.

MSD tagging is used in various fields of natural language processing. Examples are machine translation [46], lemmatization [15], and also speech recognition.

Let $w_1, \ldots, w_n$ be a sequence of words—a sentence. The task of the MSD tagger is to determine a corresponding sequence of MSD tags $t_1, \ldots, t_n$. In this process a dictionary is available with one or more possible MSD tags for each word. When only one MSD tag is possible for a word the tagging of this word becomes trivial.

For all other words the best tag has to be determined using a tagging model. This can be expressed with the equation:

$$\widehat{\{t_1,\ldots,t_n\}} = \underset{\{t_1,\ldots,t_n\}}{\arg\max}\left(P(\{t_1,\ldots,t_n\}|\{w_1,\ldots,w_n\})\right), \tag{2.19}$$

where P is a probability obtained by the tagging model. Using Bayes' formula we can express the above equation as:

$$\widehat{\{t_1,\ldots,t_n\}} = \underset{\{t_1,\ldots,t_n\}}{\arg\max}\left(P(\{t_1,\ldots,t_n\})P(\{w_1,\ldots,w_n\}|\{t_1,\ldots,t_n\})\right). \tag{2.20}$$

Now the problem is similar to that of the search algorithm in Eq. (2.11).

One problem in MSD tagging is data sparsity. Tagging models have to be trained and are later used by the tagging algorithm to tag an arbitrary text. Those models may contain a large variety of parameters that have to be learned from a training corpus. In morphologically rich languages this number is even higher, since the words are marked with a large number of features. To train an appropriate model an appropriate training corpus is needed. Such a corpus has to be already manually tagged. Thus, the building of it will be very expensive and time consuming. Existing corpora, which are manually tagger are very smaller than untagged or automatically tagged corpora. An example for Slovene: the largest general corpora of Slovene (GigaFida) has 1.2 billion words, while the larger tagged corpora has about 500,000 words [15].

Another problem is the presence of unknown words—word that appeared in the text we want to tag but do not appear in the training corpora. In an inflective language such words can be tagged based on their endings. Another special case are proper nouns (names). Since training corpora are small, we can expect that most proper nouns will not appear in them.

2.8.1 Tagging Models

Markov models are often used in MSD tagging. For estimating $P(\{t_1,\ldots,t_n\})$ in Eq. (2.20) we can use a n-gram model of tags similar as in language modeling. Regarding the term $P(\{w_1,\ldots,w_n\}|\{t_1,\ldots,t_n\})$ we can assume that the current word depends solely on the current tag:

$$P(\{w_1,\ldots,w_n\}|\{t_1,\ldots,t_n\}) = \prod_{i=1}^{n} P(w_i|t_i). \tag{2.21}$$

Using maximum likelihood estimation we get

$$P(w_i|t) = \frac{C(w_i, t_i)}{C(t_i)}, \tag{2.22}$$

where $C(\cdot)$ is a counting function. An analysis of all possible sequence of MSD tags would not be feasible. The Viterbi algorithm [51] can be applied to find the most probable sequence.

Some MSD tagging algorithms are based o a maximum entropy approach [15]. Let t be the current unknown tag and h a set of data used to determine t. This can be other MSD tags, other words and other information. In such a model the probability of the sequence of MSD tags is expressed as

$$P(t|h) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(t|h)}. \tag{2.23}$$

Here π is a normalization constant, μ and α are model parameters. The functions f_j are called features and can have values 0 and 1. Each parameter α_j corresponds to the feature f_j. The parameters are selected in a way that maximizes the probability given by (2.23) on the training set.

Other algorithms for MSD tagging employ neural networks [45], support vector machines [14] and decision trees [33]. Sometimes also expert rules are manually programmed into a tagger [15].

2.8.2 Example

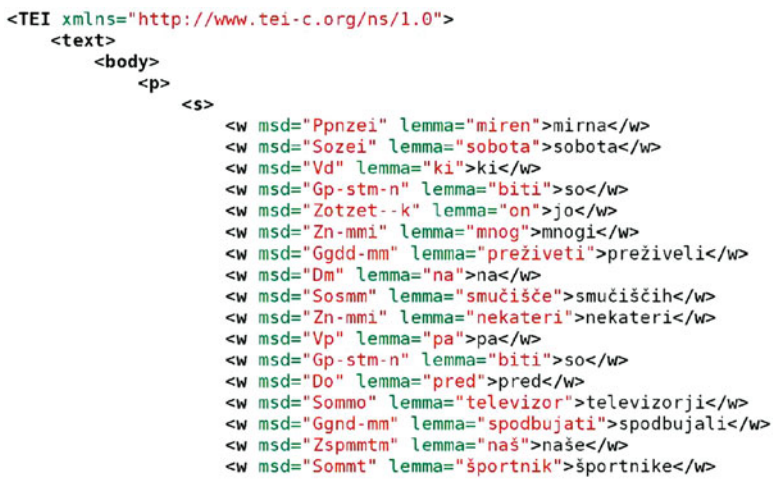
As an example we can look at the JOS Morphosyntactic Specifications for Slovene [9], a derivative of MULTEXT-EAST system. MSD tags carry information about part of speech and other grammatical categories. The JOS system specifies 12 different parts of speech with which words can be tagged (not counting unknown).

The parts of speech do not completely correspond with the parts of speech as in formal Sloven Grammar. In JOS pronouns are a part of speech while in the formal grammar different types of pronouns belong are types of other parts of speech. Also in JOS numerals are a part of speech while in the grammar they are a type of adjectives. The JOS systems also list abbreviations as parts of speech. The last part of speech in the JOS system is residual, which consists of typos, foreign words and programs (numbers, special symbols, URLs etc.).

Further the system specifies 37 grammatical categories for all parts of speech. They are stated in Table 2.3. All together there are 1902 different possible MSD tags. For comparison a MSD tag system for English could have approximately 100 different tags.

Table 2.3 Part of speech in the Slovene JOS specifications for MSD tags and corresponding grammatical categories

Part of speech	Categories
Noun	Type, gender, number, case, animate
Verb	Type, aspect, form, person, number, gender, negative
Adjective	Type, degree, gender, number, case, definiteness
Adverb	Type, degree,
Pronoun	Type, person, number, gender, case, owner number, owner gender, clitic
Numeral	Type, gender, number, case, definiteness
Preposition	Case
Conjunction	Type
Particle	/
Interjection	/
Abbreviation	/
Residual	Type



```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
    <text>
        <body>
            <p>
                <s>
                    <w msd="Ppnzei" lemma="miren">mirna</w>
                    <w msd="Sozei" lemma="sobota">sobota</w>
                    <w msd="Vd" lemma="ki">ki</w>
                    <w msd="Gp-stm-n" lemma="biti">so</w>
                    <w msd="Zotzet--k" lemma="on">jo</w>
                    <w msd="Zn-mmi" lemma="mnog">mnogi</w>
                    <w msd="Ggdd-mm" lemma="preživeti">preživeli</w>
                    <w msd="Dm" lemma="na">na</w>
                    <w msd="Sosmm" lemma="smučišče">smučiščih</w>
                    <w msd="Zn-mmi" lemma="nekateri">nekateri</w>
                    <w msd="Vp" lemma="pa">pa</w>
                    <w msd="Gp-stm-n" lemma="biti">so</w>
                    <w msd="Do" lemma="pred">pred</w>
                    <w msd="Sommo" lemma="televizor">televizorji</w>
                    <w msd="Ggnd-mm" lemma="spodbujati">spodbujali</w>
                    <w msd="Zspmmtm" lemma="naš">naše</w>
                    <w msd="Sommt" lemma="športnik">športnike</w>
```

Fig. 2.4 Example of a MSD tagged text fragment from the Slovene broadcast news database

As an example of a MSD tagger we can look Obeliks [15], a freely available MSD tagger for Slovene which includes also a lemmatizer. The tagger consists of an endings tree, feature vector generation algorithm, training algorithm, and tagging algorithm. It is based on a maximum entropy approach. Figure 2.4 shows an example of a Slovene text fragment tagged with Obeliks. The text segment is taken from the test set of the Slovene Broadcast News system.

2.9 Summary

We will present a quick summary of this chapter. The basic steps in the processes of speech recognition are:

- In speech recognition the audio signal is first processed with digital signal processing methods and the converted into a stream of so-called feature vectors. Those vectors represent the characteristics of the speech signal which are useful for speech recognition, i.e. they can be used to distinguish different phonemes.
- Feature vectors are used for training acoustic models. In large vocabulary speech recognition acoustic models are used for modeling triphones. A speech set is necessary for the training of acoustic models. A transcription of all the sentences in this set must be available for training.
- A phonetic pronunciation dictionary is necessary for all words in the training set.
- A large corpus of text is used to train language models. The most widely used type are *n*-gram language models, which use smoothing techniques and a back-off algorithm.
- A dictionary is necessary for language model training. This dictionary is later used by the search algorithm and therefore must also contain phonetic transcriptions.
- A search algorithm uses acoustic models, the language model and the dictionary to find the best hypothesis for any give acoustic signal. We can also generate are list of hypotheses.

Because our work is concatenated about language modeling, we also repeat the important properties of language models:

- The dictionary on which it is built. A larger dictionary results in a larger model, which is slower to use. However, a larger dictionary may also result in less error since in a test set there will be less word, which are not in the vocabulary.
- The models order n. This is the highest number of all words in any *n*-gram contained in the language model. A higher order is generality better, but again larger and slower to use.
- The used smoothing technique and back-off algorithm.
- The perplexity of the model can be used for a quick evaluation. When perplexity is used to compare two language models, its dictionaries must, and the perplexity must be calculated on the same test set.

The main properties of inflection languages are:

- Word inflection resulting in a large set of possible word forms.
- Relatively free word order in sentences.
- A need for larger corpora for building language models.
- A large set of different MSD tags.

References

1. Alleva F, Huang X, Hwang MY (1993) An improved search algorithm using incremental knowledge for continuous speech recognition. In: 1993 IEEE international conference on acoustics, speech, and signal processing, Minneapolis, April 1993, pp 307–310
2. Arhar Š, Gorjanc V, Krek S (2007) FidaPLUS corpus of Slovenian: the new generation of the Slovenian reference corpus: its design and tools. In: Davies M (ed) Proceedings of the corpus linguistics conference, Birmingham, 2007, pp 27–30
3. Aubert XL (2002) An overview of decoding techniques for large vocabulary continuous speech recognition. Comput Speech Lang 16:89–114. doi:10.1006/csla.2001.0185
4. Axelrod AE (2006) Factored language models for statistical machine translation. Dissertation, University of Edinburgh
5. Biem A, McDermott E, Katagiri S (1996) A discriminative filter bank model for speech recognition. In: Proceedings of the IEEE, ICASSP-96, Atlanta, May 1996, pp 545–548
6. Bilmes JA, 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, Edmonton, 2003, pp 4–6
7. Chen SF, Goodman J (1999) An empirical study of smoothing techniques for language modeling. Comput Speech Lang 13:359–394. doi:10.1006/csla.1999.0128
8. Donaj G, Kačič Z (2011) Perplexity testing of factored language models on morphological tags in the Slovene language. In: International conference; 1st, information technology and computer networks; latest trends in information technology, Vienna, 2011, pp 237–242
9. Erjavec T, Fišer D, Krek S, Ledinek S (2010) The JOS linguistically tagged corpus of Slovene. In: 7th International conference on language resources and evaluations (LREC-10), Valletta, 19–21 May 2010, pp 1806–1809
10. Flynn R, Jones E (2012) Feature selection for reduced-bandwidth distributed speech recognition. Speech Commun 54:836–843. doi:10.1016/j.specom.2012.01.003
11. Gales M, Young S (2007) The application of hidden Markov models in speech recognition. Found Trends Signal Process 1:195–304. doi:10.1561/2000000004
12. Gemmeke JF, Cranen B, Remes U (2011) Sparse imputation for large vocabulary noise robust ASR. Comput Speech Lang 25:462–479. doi:10.1016/j.csl.2010.06.004
13. Geutner P, Finke M, Scheytt P (1998) Adaptive vocabularies for transcribing multilingual broadcast news. In: Proceedings of the 1998 IEEE international conference on acoustics, speech and signal processing, Seattle, 1998, pp 925–928
14. Giménez J, Màrquez L (2004) SVMTool: a general POS tagger generator based on support vector machines. In: Proceedings of the 4th international conference on language resources and evaluation (LREC-04), Lisbon, 26–28 May 2004, pp 43–46
15. Grčar M, Krek S, Dobrovoljc K (2012) Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik. In: Jezikovne tehnologije 2012, Ljubljana, September 2012, pp 89–94
16. Hermansky H (1990) Perceptual linear predictive (PLP) analysis of speech. J Acoust Soc Am 87:1738–1752. doi:10.1121/1.399423
17. Hirsimäki T, Kurimo M (2004) Decoder issues in unlimited Finnish speech recognition. In: Proceedings of the 6th nordic signal processing symposium, Espoo, 9–11 June 2004, pp 320–323
18. Hirsimäki T, Creutz M, Siivola V, Kurimo M, Virpioja S, Pylkkönen J (2005) Unlimited vocabulary speech recognition with morph language models applied to Finnish. Comput Speech Lang 20:515–541. doi:10.1016/j.csl.2005.07.002
19. Hirsimäki T, Pylkkonen J, Kurimo M (2009) Importance of high-order N-gram models in morph-based speech recognition. IEEE Trans Audio Speech 17:724–732. doi:10.1109/TASL.2008.2012323
20. Huang X, Acero A, Hon HW (2001) Spoken language processing: a guide to theory, algorithm and system development. Prentice Hall PTR, Upper Saddle River

21. Huet S, Gravier G, Sebillot P (2010) Morpho-syntactic post-processing of N-best lists for improved French automatic speech recognition. Comput Speech Lang 24:663–684. doi:10.1016/j.csl.2009.10.001
22. Ircing P, Psutka JV, Psutka J (2009) Using morphological information for robust language modeling in Czech ASR system. IEEE Trans Audio Speech 17:840–847. doi:10.1109/TASL.2009.2014217
23. Jelinek F (1976) Continuous speech recognition by statistical methods. Proc IEEE 64:532–556. doi:10.1109/PROC.1976.10159
24. Jiang H (2010) Discriminative training of HMMs for automatic speech recognition: a survey. Comput Speech Lang 24:589–608. doi:10.1016/j.csl.2009.08.002
25. Katz S (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans Acoust Speech 35:400–401. doi:10.1109/TASSP.1987.1165125
26. Kaufmann T, Pfister B (2012) Syntactic language modeling with formal grammars. Speech Commun 54:715–731. doi:10.1016/j.specom.2012.01.001
27. Kirchhoff K, Vergyri D, Bilmes J, Duh K, Stolcke A (2005) Morphology-based language modeling for conversational Arabic speech recognition. Comput Speech Lang 20:589–608. doi:10.1016/j.csl.2005.10.001
28. Kirchhoff K, Bilmes J, Duh K (2008) Factored language models tutorial. http://ssli.ee.washington.edu/people/duh/papers/flm-manual.pdf. Accessed 1 June 2015
29. Klakow D, Peters J (2002) Testing the correlation of word error rate and perplexity. Speech Commun 38:19–28. doi:10.1016/S0167-6393(01)00041-3
30. Krek S (2012) Slovenski jezik v digitalni dobi: the Slovene language in the digital age. Springer, Heidelberg
31. Logar Beginc N, Kosem I (2011) Gigafida – the new corpus of modern Slovene: what is really in there? In: The second conference on Slavic Corpora, Dubrovnik, 12–14 September 2011
32. Lv Z, Liu W, Yang Z (2009) A novel interpolated N-gram language model based on class hierarchy. In: International conference on natural language processing and knowledge engineering, Dalian, 24–27 September 2009, pp 1–5
33. Màrquez L, Rodríguez H (1998) Part-of-speech tagging using decision trees. In: Nedellec C, Rouveirol C (eds) Machine learning: ECML-98. Springer, Heidelberg, pp 25–36
34. Ming J, Smith FJ (1999) A Bayesian triphone model. Comput Speech Lang 13:195–206. doi:10.1006/csla.1999.0120
35. Mohri M, Pereira F, Riley M (2001) Weighted finite-state transducers in speech recognition. Comput Speech Lang 16:69–88. doi:10.1006/csla.2001.0184
36. Mousa AED, Shaik MAB, Schlüter R, Ney H (2010) Sub-lexical language models for German LVCSR. In: 2010 IEEE spoken language technology workshop (SLT), Berkeley, 2010, pp 171–176
37. Mousa AED, Shaik MAB, Schlüter R, Ney H (2011) Morpheme based factored language models for German LVCSR. In: Proceedings of interspeech 2011, Florence, August 2011, pp I-1053–I-1056
38. Najedlova D (2002) Comparative study on bigram language models for spoken Czech recognition. In: Sojka P, Kopeček I, Pala K (eds) Text, speech and dialogue: 5th international conference, TSD 2002, Brno, September 2002. Lecture notes in computer science, vol 2448. Springer, Heidelberg, pp 197–204
39. Nouza J, Nejedlova D, Zdansky J, Kolorenc J (2004) Very large vocabulary speech recognition system for automatic transcription of Czech broadcast programs. In: Proceedings of interspeech 2004, Jeju, pp 409–412
40. Nouza J, Zdansky J, Cerva P et al (2010) Challenges in speech processing of slavic languages (case studies in speech recognition of Czech and Slovak). In: Esposito A, Campbell N, Vogel C et al (eds) Development of multimodal interfaces: active listening and synchrony: second COST 2102 international training school. Springer, Heidelberg, pp 225–241
41. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77:257–286. doi:10.1109/5.18626

42. Ramirez J, Gorriz JM, Segura JC (2007) Voice activity detection. Fundamentals and speech recognition system robustness. In: Grimm M, Kroschel K (eds) Robust speech recognition and understanding. InTech, Vienna, pp 1–22
43. Rotovnik T, Sepesy Maučec M, Kačič Z (2007) Large vocabulary continuous speech recognition of an inflected language using stems and endings. Speech Commun 49:437–452. doi:10.1016/j.specom.2007.02.010
44. Sak H, Saraçlar M, Güngör T (2010) Morphology-based and sub-word language modeling for Turkish speech recognition. In: 2010 IEEE international conference on acoustics speech and signal processing (ICASSP), Dallas, 14–19 March 2010, pp 5402–5405
45. Schmid H (1994) Part-of-speech tagging with neural networks. In: Proceedings of the 15th international conference on computational linguistics, Kathmandu, 6–12 April 1994, pp 172–176
46. Sepesy Maučec M, Donaj G, Kačič Z (2013) Improving statistical machine translation with additional language models. In: 6th Language & technology conference, Poznan, 7–9 December 2013, pp 137–141
47. Shaik MAB, Mousa AED, Schlüter R, Ney H (2011) Using morpheme and syllable based subwords for polish LVCSR. In: 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP), Prague, 22–27 May 2011, pp 4680–4683
48. Shin JW, Chang JH, Kim NS (2010) Voice activity detection based on statistical models and machine learning approaches. Comput Speech Lang 24:515–530. doi:10.1016/j.csl.2009.02.003
49. Su Y, Jelinek F, Khudanpur S (2007) Large-scale random forest language models for speech recognition. In: Proceedings of interspeech, Antwerp, 2007, pp 598–601
50. Topirišic J (1984) Slovenska slovnica. Obzorja, Ljubljana
51. Viterbi AJ (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans Inform Theory 13:260–269. doi:10.1109/TIT.1967.1054010
52. Whittaker EWD, Woodland PC (2003) Language modelling for Russian and English using words and classes. Comput Speech Lang 17:87–104. doi:10.1016/S0885-2308(02)00047-5
53. Young SJ, Evermann G, Gales MJF et al (2006) The HTK book, version 3.4. Cambridge University Press, Cambridge
54. Zablotskiy S, Zablotskaya K, Minker W (2010) Some approaches for Russian speech recognition. In: 2010 Sixth international conference on intelligent environments, Kuala Lumpur, 19–21 July 2010, pp 96–99
55. Žgank A, Verdonik D, Zögling Markuš A, Kačič Z (2005) BNSI Slovenian broadcast news database – speech and text corpus. In: Proceedings of interspeech 2005 – Eurospeech, Lisbon, 4–8 September 2005, pp 2525–2528
56. Zitouni I (2007) Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. Comput Speech Lang 21:88–104. doi:10.1016/j.csl.2006.01.001

Chapter 3
Performance Evaluation Using Lexical Data

Abstract In this chapter we will present established methods to measure the accuracy of a speech recognition system: word error rate, character error rate, and also lemma error rate and well as their usefulness. We will describe the Levenshtein distance used for sentence alignment and calculating word error rate. Some issues that arise from its use that affect the performance evaluation and error analysis are pointed out. This issues will be presented in the form of a typical example. We also introduce a generalization of the Levenshtein distance in the case of sentence alignment that compares words inside sentences and also characters inside those words. This expended Levenshtein distance is then used to perform a better alignment of the recognizers hypothesis and the reference transcription. This new alignment helps to automatically recognize different types recognition errors. Using this improved alignment an error analysis with regard to lexical features will be presented. Errors are analyzed based on word lengths, part-of-speech classification, grammatical categories and word lemma. The final analysis categorizes errors based on lexical properties in several disjoint groups. Based on this categorization we state some further guidelines on how to reduce error rates in inflective languages. This categorization also presents how the inflective nature of the used language increases the number of errors in speech recognition.

3.1 Established Measures of Accuracy

Every speech recognition system is susceptible to errors. This means that the recognizers output hypothesis is not equal to the actual spoken sentence. Errors can have different causes. Often acoustical features can have an important impact on the recognition accuracy. These effects are well researched, e.g. [4]. Other research show for example the impact of gender [2], speech rate [10] or spontaneous speech [3]. The impact of lexical features is not as well researched. In this chapter we will give an example of error analysis based on different lexical features that can be derived from the basic word form, the lemma, or the words MSD tag.

In speech recognition errors are typically firstly classified into 3 types:

- Substitution errors where a spoken word was misrecognized for another word which is then present in the recognizers output.

G. Donaj, Z. Kačič, *Language Modeling for Automatic Speech Recognition of Inflective Languages*, SpringerBriefs in Electrical and Computer Engineering,
DOI 10.1007/978-3-319-41607-6_3

```
LAB: se je zaradi zagotavljanja nege     oskrbovancev
REC: se    zaradi zagotavljanje neka ste bova
```

Fig. 3.1 Examples of the three error types. The words *zagotavljanja*, *nege*, and *oskrbovancev* were substituted; the word *je* was deleted; the word *ste* was inserted. The label `LAB:` is used to indicate the reference transcription and the label `REC:` to indicate the recognizers output hypothesis

- Deletion errors where a spoken word was not recognized at all and is therefore not in the recognizers output.
- Insertion errors where a word is present in the recognizers output that has not been spoken.

A rather poorly recognized example from the Slovene Broadcast News system is shown in Fig. 3.1. In this example all tree types of errors are present.

The errors and their types in Fig. 3.1 are rather obvious. The question now is how to count errors and how give a final score of the accuracy of the speech recognizer.

Word error rate (WER) is a well established metric for evaluating the performance of a speech recognition system. WER weights all errors equally important. It is calculated by the equation

$$WER = \frac{I + D + S}{N} \cdot 100\,\%, \tag{3.1}$$

where I, D, S and N are the numbers of inserted, deleted, substituted and the total number of words respectively. To find these numbers we must perform an alignment between the true spoken sentence and the recognizers output. These alignments are found through a dynamic programming algorithm that minimizes the number of editing operations, called the Levenshtein distance and also represents the number of errors used to calculate WER. The particular implementation of the algorithm determines which alignment will be the result by preferring one editing operation when several are possible.

However, it has been before mentioned that all errors may not have the same impact on human perception of the quality of the speech recognition output [1] and an error rate that weights different kinds of errors with different coefficients has been suggested. A speech recognition system can also be part of a speech translation system and it is also not necessary that the overall performance of a speech translation system is best when the speech recognition module is tuned for minimal WER [6]. Also in other applications the overall performance can not be best expresses by WER. We will show more detail on this aspect in the next chapter.

Here, we will also show that standard WER is not appropriate for an error analysis. For this purpose we will introduce a modified WER and a word alignment based on extending weighted Levenshtein distance [1].

The next performance measure is the sentence error rate, which is calculated as the rate of all sentences in the test set which contain at least one error. Sentence error rate are usually much higher than WER and are rarely used in research. Still, they could be useful in a limited domain and its calculation it also included in some speech recognition tools like HTK [11].

Another measure is character error rate. Substituted words are sometime very similar because the speech recognizer mistook the word for a similar one. Using editing operation on a character level will also convert a recognized sentence into the correct one. The character error rate is then calculated as the rate between editing operations of characters and the total number of characters. Character error rate is usually used in Chinese speech recognition, because of the nature of its writing, e.g. [8].

Huet et al. [7] have introduced the lemma error rate and the lemma error rate on lexical words. The former will ignore errors in inflection and the latter will also ignore all errors on grammatical words (interjections, conjunctions, etc.). The nature of these metrics makes them an interesting choice for research in inflective languages. To calculate these metrics we must use first have a MSD tagger to determine which words are lexical and which grammatical and also a lemmatizer. Since inflection errors are ignored, lemma error rate results will be lover than WER results.

Morris et al. [9] have also introduced the match error rate and word information lost as they felt a more meaningful performance measure would be given by the proportion of information.

3.2 Alignment Issues of the Levenshtein Distance

We will show that the alignment achieved by using the Levenshtein distance is not always appropriate when we want to analyze error types in speech recognition on a lexical level. We will present a new alignment method, that counts more errors that Levenshtein distance, but is more suited for automatic error analysis. We also show that this new alignment can detect improvements in speech recognition, which WER can not detect.

The Levenshtein distance gives us the minimum number of edit operations that are necessary for transforming the recognition output into the reference transcription or vice versa. Edit operations are insertion, deletion and substitution. Thus, in the process of calculating this distance also a possible alignment between the hypothesis and the reference is determined. However, there is often more that one alignment with the minimum number of editing operation possible. An example of such an alignment of a speech recognition hypothesis with the corresponding reference transcription is show in Fig. 3.2. The Levenshtein distance for the example in the figure is 2. This means we have two errors:

- the word “o” was substituted with the word “kloniranja”
- the word “kloniranju” was deleted

```
LAB: področje o          kloniranju človeških celic
REC: področje kloniranja            človeških celic
```

Fig. 3.2 One possible Levenshtein distance alignment with a substitution first and a deletion second

```
LAB: podroCje o kloniranju človeških celic
REC: podroCje   kloniranja človeških celic
```

Fig. 3.3 An alternative alignment of the segment shown in Fig. 3.2 with a deletion first and a substitution second

However, if we look at the transcription and the hypothesis, we may say that the two errors, which seem to be better suited to describe the recognition output are:

- the word “o” was deleted
- the word “kloniranju” was substituted with the word “kloniranja”

This results will be obtained with an alternative alignment shown in Fig. 3.3. Both alignment show the same number of editing operations. These alignments are found through a dynamic programming algorithm that minimizes the number of editing operations. Which alignment will be the result by is determined by the particular implementation of the algorithm, which tries one editing operation first when more than one is possible for obtaining the same number of operations.

3.3 An Expanded Levenshtein Distance

A well-known generalization of the Levenshtein distance is the weighted Levenshtein distance (WLD) [1], defined by the equation

$$WLD = W_i \cdot I + W_s \cdot S + W_d \cdot D, \tag{3.2}$$

where I, S, and D are the numbers of insertions, substitutions, and deletions respectively. W_i, W_s , and W_d are the corresponding weights. If these weights are all equal to 1, the result will be the standard or unweighted Levenshtein distance.

We now introduce a further generalization of WLD that works with words and characters and has a total of six weights:

- word insertion weight: W_i,
- word substitution weight: W_s,
- word deletion weight: W_d,
- character insertion weight: C_i,

- character substitution weight: C_s,
- character deletion weight: C_d.

The weights can be classified into two groups. We call the first three word level weights and the last three character level weights. The alignment algorithm is designed to find the alignment that minimize the sum:

$$D = \sum_{i \in \mathscr{I}} [W_I + C_i \cdot L(i)] + \sum_{d \in \mathscr{D}} [W_D + C_D \cdot L(d)] + \sum_{(s_1,s_2) \in \mathscr{S}} [W_S + C_S \cdot CLD(s_1, s_2)] \,. \tag{3.3}$$

$\mathscr{I}$, $\mathscr{S}$, and $\mathscr{D}$ are the sets of all insertions, substitutions and deletions in the alignment respectively. $L(i)$ and $L(d)$ are the lengths of the inserted and deleted word respectively. $CLD(s_1, s_2)$ is the Levenshtein distance on the character level between the two words in the substitution word pair s_1 and s_2.

Thus, this method calculates a total scores based on the weighted number of inserted words, substituted words, deleted words, inserted characters, substituted characters and deleted characters.

For example: a word is to be deleted. This deletion adds a term to the total score in the above equation. The terms value equals the word deletion weight summed with the character deletion weight times the number of characters in this word.

We will later present an error analysis on the Slovene Broadcast News (BNSI) [12] system. It should be noted that the current LVCSR system on the BNSI database uses grapheme based transcriptions for acoustical models instead of phoneme based ones. This is due to the fact that a grapheme-phoneme conversion for Slovene is a non-trivial process and there is no tool available for an automatic conversion, that has given us satisfactory results. Should one be available we propose the algorithm should be based on word lengths and distances in terms of phonemes rather than characters.

If the word based weights are large enough compared to the character based weights, the resulting alignment will have the same number of editing operations as the Levenshtein distance, but the algorithm will prefer one particular alignment, e.g. the alignment in Fig. 3.3 over that in Fig. 3.2.

The same number of errors will always be the case if for example the word based weights are equal, the character based weights are equal, and the word based weights are greater by a factor that exceeds the number of characters in the hypothesis. However, if the proportion between this two values is small enough, e.g. 1:5, the resulting alignments may results in scoring more recognition errors than the Levenshtein distance. Why this is useful can be seen by comparing the alignments in Figs. 3.4 and 3.5.

These two alignments are obtained with our new algorithm using different weights. The alignment on Fig. 3.5 shows the insertion of one short word, the deletion of one short word and two substitutions of two long words with similar words. This alignment is more reasonable than the alignment in Fig. 3.4, which shows three substitutions of very dissimilar words. Thus, the alignment in Fig. 3.5 is more useful for an automatic error analysis.

```
LAB: tonček s     svojimi   vragolijami občinstvo navdušuje
REC: tonček svoje vragolije mi          občinstvo navdušuje
```

Fig. 3.4 An alignment with three errors, all substitutions

```
LAB: tonček s svojimi vragolijami    občinstvo navdušuje
REC: tonček   svoje   vragolije   mi občinstvo navdušuje
```

Fig. 3.5 Alternative alignment with four errors—one error more compared to the Levenshtein alignment for the same sentences in Fig. 3.4

```
LAB: tonček s svojimi vragolijami    občinstvo navdušuje
REC: tonček   svojimi vragolije   mi občinstvo navdušuje
```

Fig. 3.6 An improved hypothesis to the one in Figs. 3.4 and 3.5

We can analyze this example ever further. Let us assume that we have tried to improve our speech recognition system and that we obtained the hypothesis in Fig. 3.6. We see that one more word was correctly recognized. However, the Levenshtein distance between this hypothesis and the reference transcription is still 3. Thus, no improvement in WER would be obtained. With the proposed alignment system and our generalized Levenshtein distance the number of error would also be three, but that would show us an improvement from four errors to three errors.

Before we can use the proposed alignment method to determine speech recognition errors, some reasonable values for the weights must be determined. We used speech recognition result and aligned them with the reference transcriptions using different combinations of weights. We restricted the tests to the cases where the word based weights are equal and the character based weighs are as well. We fixed the word based weights to the value 100 and generated alignment for character based weight values of: 0, 1, 2, 5, 10, 20 and 50. With 0, the result will always be the same as using the standard Levenshtein distance. Increasing the weights resulted in an increase in the number of errors. We compared the differences in alignments obtained with two consecutive values. All the changes were manually verified and classified into three categories. A change was classified as positive (+) if the new alignment was clearly considered better than the previous, e.g. when comparing Figs. 3.4 and 3.5. A change could also have been considered negative (−) if it was clearly considered worse than the previous one. If there was a doubt if the new alignment was better or worse, the change was classified neutral (0). We accepted an increase in character based weights if at least half the changes were classified positive.

Table 3.1 Number of errors with different character level weights

Weight	Insertions	Deletions	Substitutions	Total
0	674	1568	5466	7708
1	724	1618	5366	7708
2	724	1618	5366	7708
5	724	1618	5366	7708
10	737	1631	5355	7723
20	772	1666	5319	7757
50	807	1701	5283	7791
100	832	1726	5260	7818

Table 3.2 Comparison of alignments with different character level weights

Weights	Changed segments	+	0	−
005–010	15	14	1	0
010–020	33	25	8	0
020–050	33	15	18	0

Table 3.1 presents the exact number of different types of errors and the total number of error when using different character level weights. Considering the comparison of the two alignments in Figs. 3.4 and 3.5, the changes in the number of different types of errors are as expected. The number of insertions and deletions increases, while the number of substitutions decreases.

With higher values of the weights we also see an increase in the total number of errors, but most of the alignment changes are still considered positive as shown in Table 3.2. The table shows the two compared weights, the number of segments in which different alignments occurred and the classification of those differences.

The classification was not done comparing results with weights 0 and 1, since there was no difference in number of errors. Comparing results with weights between 1 and 5 showed no changes in the resulting alignments. We stopped when comparing results with weight values 20 and 50, where the number of positive changes was first lower than 50 %. Thus, we decided to use the value 20 for generating alignments for error analysis.

We used this weight in defining the modified word error rate (mod-WER) as the ratio between needed editing operation in the final alignment as shown in Fig. 3.6 and the total number of words in the reference transcription. Since any sequence of editing operation will have at least as many steps as the value of the Levenshtein distance, modified WER will always be greater that standard WER.

3.4 Error Analysis Example

Now we will show an example of the use of the above presented alignment on a LVSCR example. We used the BNSI database and performed speech recognition using the same acoustical models and different language models.

Table 3.3 Word recognition results with standard WER

Model	Correct	Insertions	Deletions	Substitutions	Total errors	WER (%)
2g + 60k	15,759	724	1618	5366	7708	33.89
2g + 300k	16,441	343	1776	4526	6645	29.22
3g + 60k	16,510	756	1331	4902	6989	30.73
3g + 300k	17,282	371	1417	4044	5832	25.64

Table 3.4 Word recognition results with modified WER

Model	Correct	Insertions	Deletions	Substitutions	Total errors	Mod-WER (%)
2g + 60k	15,758	772	1666	5319	7757	34.11
2g + 300k	16,439	370	1803	4501	6674	29.35
3g + 60k	16,511	799	1374	4858	7031	30.92
3g + 300k	17,282	400	1446	4015	5861	25.77

3.5 General Results

The error analysis was done in several steps. The first step was concentrated on the numbers of insertion, deletion, substitutions, the total number of errors and as well as the number of correct words in the speech recognizer hypothesis. The results are analyzed for using word error rate (WER) and the modified WER presented in the previous section (mod-WER). The results were analyzed for the use of four different language models: bigram and trigram models with vocabularies of 60,000 and 300,000 words. The same analysis was later repeated with words replaced by they lemmas. Results are shown in Tables 3.3 and 3.4.

Comparing results in Table 3.3, we see the impact of vocabulary size and language model order on errors. Using the larger vocabulary reduces WER by 4–5 %. Most of this reduction is due to the reduction in the number of substitutions. We also see that the number of insertion reduces to less then a half. However, the number of deletions increases slightly. The comparison between results obtained using bigram and trigram models shows a reduction in WER of 3–4 %, with a slight increase in the number of insertions.

Comparing Tables 3.3 and 3.4 we see the impact of the modified alignment algorithm on recognition results. Modified WER results are between 0.11 and 0.22 % greater than WER results. The main impact of using modified WER can be seen by comparing results for individual error type. While the number of substitutions decreases, the numbers of insertions and deletions increases.

One of the assumptions for having many errors in LVSCR systems of an inflective language is that words are misrecognized for other words of the same lemma and different grammatical features. This effect can be seen by comparing Table 3.3 with Table 3.5 or comparing Table 3.4 with Table 3.6. Lemma recognition results are between 2.8 and 3.5 % better than word recognition results. Are reasonable assumption is that this is the approximate amount of error in form of false word forms or word endings.

Table 3.5 Lemma recognition results with standard WER

Model	Correct	Insertions	Deletions	Substitutions	Total errors	WER (%)
2g + 60k	16,590	763	1657	4496	6916	30.41
2g + 300k	17,152	371	1804	3787	5962	26.21
3g + 60k	17,314	785	1360	4069	6214	27.32
3g + 300k	17,393	393	1439	3365	5197	22.85

Table 3.6 Lemma recognition results with modified WER

Model	Correct	Insertions	Deletions	Substitutions	Total errors	Mod-WER (%)
2g + 60k	16,588	790	1684	4471	6945	30.54
2g + 300k	17,151	380	1813	3779	5972	26.26
3g + 60k	17,313	805	1380	4050	6235	27.42
3g + 300k	17,938	406	1452	3353	5211	22.91

Table 3.7 Most frequent words in errors

Deleted	(1446)	Inserted	(400)	Substitutions	(4015)	Falsely recogn.	(4015)
je	158	je	54	eee	193	je	100
v	147	v	41	v	68	da	66
eee	80	in	32	je	56	se	52
in	73	na	14	pa	43	v	43
da	51	ne	13	se	38	in	41
na	47	so	12	in	35	pa	37
pa	40	se	11	ki	34	so	35
to	34	to	10	za	27	na	35
se	28	da	10	da	26	za	34
ki	28	Se	9	seveda	25	to	28
ne	25	s	9	ne	23	e	28
z	22	pa	8	so	22	po	27
bi	22	za	6	to	21	ne	25
iz	21	po	6	z	20	bo	20
so	19	od	6	vse	18	tako	19

3.6 Most Frequent Words and Lemmas

After the first step we restricted our self to results obtained with the new alignment method, the trigram language model, and the 300,000 words vocabulary.

The next step was to list the bare word forms of the most frequently inserted and deleted words and the most frequent substitution pairs. The procedure was repeated with replacing the words with the corresponding lemmas. Results are in Tables 3.7 and 3.8.

The first row in Tables 3.7 and 3.8 shows the total number of errors of a specific type. From Table 3.7 we see that the three most frequent words in error are *je*, *v*

Table 3.8 Most frequent lemmas in errors

Deleted	(1446)	Inserted	(400)	Substitutions	(4015)	Falsely recogn.	(4015)
biti	255	biti	84	biti	187	biti	273
v	147	v	41	eee	144	ta	66
in	73	in	32	v	69	da	66
ta	60	na	14	ta	68	se	57
eee	54	ta	13	se	50	v	43
da	51	ne	13	pa	44	in	41
na	47	on	12	eea	38	pa	37
on	44	z	11	in	35	na	35
pa	40	se	11	ki	34	za	34
z	37	da	10	on	32	on	29
se	31	Se	9	jaz	28	e	28
ki	28	pa	8	za	27	po	27
ne	25	za	6	da	26	ne	25
jaz	23	po	6	seveda	25	z	23
iz	21	od	6	z	23	jaz	22

and *in*. Together they account for 12.7 % of all errors. The filler *eee* is always either deleted or misrecognized for another word. This is due to the vocabulary and the language model.

3.7 Word Lengths

The next distinction of error was based on word lengths. Since the speech recognition system is based on grapheme transcriptions, the length based on graphemes. Results are in Table 3.9. The relative numbers are the ration between the number of the given type of error and the total number of error for the given word length.

The results reveal that more than 85 % of all deletion and insertion errors are observed with words of no more than three characters. The rate of substitution error is more equally distributed with regard to the word length. Still almost 50 % of all errors are associated with words with no more than three characters.

3.8 Parts-of-Speech

Part of speech classification is based on the morphosyntactic specifications of JOS, which distinguished parts of speech differently than the formal Slovene grammar. In the JOS specifications we distinguish 12 different POS tags and the *unknown* tag, which is also a possible output of the MSD tagger Obeliks [5], if the tagging algorithm does not recognize the word.

Table 3.9 Word lengths of errors

Length	Total	Deleted		Inserted		Substitutions		Falsely recogn.	
All	22,743	1446	6.36 %	400	1.76 %	4015	17.65 %	4015	17.65 %
1	1068	228	21.35 %	69	6.46 %	122	11.42 %	122	11.42 %
2	5211	761	14.60 %	240	4.61 %	543	10.42 %	714	13.70 %
3	1899	256	13.48 %	40	2.11 %	546	28.75 %	321	16.90 %
4	2051	85	4.14 %	19	0.92 %	386	18.82 %	416	20.28 %
5	2419	42	1.74 %	18	0.74 %	746	30.84 %	508	21.00 %
6	2447	30	1.26 %	5	0.20 %	542	22.15 %	543	22.19 %
7	1960	17	0.87 %	2	0.10 %	352	17.96 %	400	20.41 %
8	1917	14	0.73 %	4	0.21 %	327	17.06 %	353	18.41 %
9	1465	5	0.34 %	1	0.07 %	252	17.20 %	262	17.88 %
>9	2306	8	0.35 %	2	0.03 %	469	20.34 %	376	16.31 %

Table 3.10 Parts of speech of errors

POS	Total words	Total errors	Relative errors (%)
All	22,743	5861	25.77
Noun	6721	1558	23.18
Verb	3901	1101	28.22
Adjective	2646	547	20.67
Adverb	1677	414	24.69
Pronoun	1536	527	34.31
Numeral	747	205	27.44
Preposition	2499	687	27.49
Conjunction	1964	552	28.11
Particle	1038	258	24.86
Interjection	8	8	100.00
Abbreviation	0	0	
Residual	0	0	
Unknown	6	4	66.67

Table 3.10 shows the number of total errors for each POS. For substitution errors, where the words in the substitution pair belong to different parts-of-speech, the error was counts for the POS of the transcription word. Relative error are expressed as the ratio between the number of errors of the given POS and the number of all words with this POS in the transcription.

Error rates of different parts-of-speech have similar values. From this basic results we can not conclude that one part of speech is more likely to produce an error than another part of speech. The exceptions are interjections which are all substituted. We can assume that this is due to the fact that interjections are not very common in written language and the language model was trained only on a written corpus. Also interjections are very rare in the test set, so that the error rate of interjections is not reliable.

Table 3.11 Parts of speech of errors

POS	Total	Deleted		Inserted		Substitutions		Falsely recogn.	
All	22,743	1446	6.36 %	400	1.76 %	4015	17.65 %	4015	17.65 %
Noun	6721	143	2.13 %	40	0.60 %	1375	20.46 %	1383	20.58 %
Verb	3901	293	7.51 %	87	2.23 %	721	18.48 %	801	20.53 %
Adjective	2646	35	1.32 %	10	0.38 %	502	18.97 %	438	16.55 %
Adverb	1677	77	4.59 %	6	0.36 %	331	19.74 %	300	17.89 %
Pronoun	1536	184	11.98 %	40	2.60 %	303	19.73 %	301	19.60 %
Numeral	747	30	4.02 %	18	2.41 %	157	21.02 %	136	18.21 %
Preposition	2499	345	13.81 %	102	4.08 %	240	9.60 %	253	10.12 %
Conjunction	1964	241	12.27 %	64	3.26 %	247	12.58 %	269	13.70 %
Particle	1038	98	9.44 %	32	3.08 %	128	12.33 %	126	12.14 %
Interjection	8	0	0.00 %	0	0.00 %	8	100.00 %	2	25.00 %
Abbreviation	0	0		0		0		0	
Residual	0	0		0		0		0	
Unknown	6	0	0.00 %	1	16.67 %	3	50.00 %	6	100.00 %

Words that were not recognized by the tagger and belong to *unknown POS* are some foreign words (mostly proper names) and a typing error in the reference transcription.

More detailed results are in Table 3.11, where error types are considered. Insertion and deletion errors of major parts of speech (nouns, adverbs, adjectives, verbs and numerals) are small compared to deletion and insertions rates of other parts of speech. An exception are verbs, where most of the deletion and insertion errors are different forms of the verb *biti* (eng. *to be*).

The rate of inserted numerals is however quite high for a major part of speech. Most of these inserted numerals are short words. In some cases a numeral in a segment was misrecognized for two words, both numerals. A merge of these two words then equals the reference word. Writing rules for Slovene numerals are somewhat difficult and in some cases the transcriber made an error writing a number a one word. In these cases the recognizer actually returned the grammatically correct transcription.

We also see that substitution error rates of prepositions, conjunctions and particles are lower compared to other parts of speech. This is not surprising as these parts of speech are often short words and are sooner classified as deleted or inserted by our alignment algorithm.

Table 3.12 show errors of nouns according to whether the noun is a common noun or a proper name. We see that the error rate is much higher for proper names. The error rate on proper names is about 15 % higher, while out of vocabulary rate of proper names is only about 2.7 % higher (3.88 % for proper names and 1.19 % for common nouns). We assume the difference in the error rates are due to the frequency of those words in the corpus.

Table 3.12 Error of different noun types

Noun type	Total	Deleted	Inserted	Substitutions	Falsely recogn.	Error rate (%)
Common noun	5612	106	27	1027	1148	20.67
Proper name	1109	37	13	348	235	35.89

3.9 Word-Forms

It has often been stated that large vocabulary speech recognition of Slavic (and other) inflectional languages is more difficult that the recognition of English. As a possible explanation the inflectional nature of these languages was presented. It is clear that vocabulary enlargement will reduce the out-of-vocabulary rate. In our two vocabularies of 60,000 and 300,000 words the out of vocabulary rates on the BNSI test set are 6.94 and 1.02 % respectively.

In inflectional languages word are inflected based on their grammatical role into several different word forms. In Slovene this inflection causes word forms to have different endings. We assume that the acoustical similarity and the similar meaning of such words is likely to cause a substitution error, where a word is substituted with a different word form with the same word root and the same lemma, but with a different ending.

Table 3.13 show the total number of substitution, where both words in the substitution pair are of the same part of speech, and the number of word form errors. Results are given for each part of speech. An error is considered a word form error, if the two words in the substitution pair have the same lemma. We see that a considerable rate of substitution error of inflectional parts of speech is due to word form error. Other parts of speech are not inflectional. Exceptions are the two substitutions of preposition. In both cases the prepositions *s* and *z* were substituted. The total number of 674 error represents 11.5 % of all error in the speech recognizer.

In Table 3.13 we considered only substitutions with the same part of speech. However, a change in the ending of a word can also substitute adjectives with adverbs. The number of such substitutions is in Table 3.14. The pairs were manually evaluated and the number of substitutions with the same word root was counted. An example of such a pair are the adjective *uspešne* and the adverb *uspešno*. We see that also here a considerable rate of errors is due to word end changing. However, the absolute values are much smaller that those presented in Table 3.13.

3.10 F-classes and Gender

Until now, we analyzed speech recognition on the whole test set. In this section we divide the results based on F-class and the speakers gender. The segments in the BNSI database are divided into the following F-classes [12]:

Table 3.13 Substitutions with the same part-of-speech

POS	Substitutions	Word form errors	
Noun	852	293	34.39 %
Verb	373	183	49.06 %
Adjective	232	134	57.76 %
Adverb	65	0	0.00 %
Pronoun	89	46	51.69 %
Numeral	77	16	20.78 %
Preposition	88	2	2.27 %
Conjunction	59	0	0.00 %
Particle	13	0	0.00 %
Interjection	0	0	
Abbreviation	0	0	
Residual	0	0	
Unknown	0	0	
Total	1848	674	36.47 %

Table 3.14 Substitutions between adjectives and adverbs

POS pair	Substitutions	Same word root
Adjective → adverb	42	11
Adverb → adjective	25	4

- F0: read/prepared speech without acoustical background,
- F1: spontaneous speech without acoustical background,
- F2: low acoustical quality (e.g. over telephone lines),
- F3: speech with background music,
- F4: speech with background noise,
- F5: foreign (non-native) speaker,
- FX: unknown.

Table 3.15 present WER on different F-classes inside the BNSI test set and different genders of speakers. We see a clear difference between F0 and F1, which confirms that spontaneous speech is harder to recognize. Also, higher error rates in F3 and F4 show the impact on acoustical background on speech recognition. These results are as expected. The last two rows also show that higher error rates occur with male speakers which is consistent with previous studies.

From the results we can say that in good acoustical conditions and with prepared speech the error rate of our speech recognition system is about 13 %. This rate increases for spontaneous speech by approximately 15 %. The error rate for F0 and F1 together is 20.47 %. With music or noise in the background the error rate increases by approximately 10 %.

We repeated the basic analyzing procedures from the previous section on the test subsets of different F-classes to see if there is any large difference. We disregarded F2, due to its size. Most results were comparable. A notable exception was the filler *eee*, as it does not appear in read/prepared speech. However, the ratio of word form errors, described in the previous section, was in F0 with 20.6 % almost twice as high as the average in the whole test set.

Table 3.15 F-class and gender errors

Class	Size	WER (%)
F0	7882	13.44
F1	3288	37.35
F2	188	61.86
F3	1409	31.72
F4	8456	29.92
F5	0	0.00
FX	1590	32.96
Male	12,156	29.90
Female	10,536	20.76

Table 3.16 Word error rates of disjoint sources

Type of error	Abs. (all) (%)	Rel. (all) (%)	Abs. (F0) (%)	Rel. (F0) (%)
Out of vocabulary	1.02	3.96	0.70	5.2
Proper names	1.75	6.79	1.04	7.7
Filler *eee*	1.20	4.66	0.18	1.3
To be	2.31	8.97	1.29	9.6
Word form errors*	2.62	10.2	2.50	18.6
Minor POS	6.58	25.5	2.89	21.5

3.11 Summary

Here we will repeat the main findings from the results presented in this section so far:

- Words with lengths smaller or equal 3 characters are more likely to be deleted or inserted that longer words.
- Pronouns, prepositions and conjunctions are more likely to be deleted than other parts of speech. Adjectives are very rarely deleted.
- Nouns, adjectives and adverbs are less likely to be inserted. Mostly prepositions, conjunctions and particles are inserted.
- Prepositions are less likely to be substituted for other words.
- Proper names have high error rates, although their out of vocabulary rate is not much higher compared to other words.
- About 11 % of all error are word form errors (false ending). In F0 this rate is 20
- We can confirm that male speaker have higher error rates.

From the error analysis so far we present a possible error categorization with disjoint error sources in Table 3.16. To see the impact of spontaneous speech and acoustical background in these types of error we present the numbers for the whole test set and for the F0 class.

The Minor part of speech row covers prepositions, conjunctions, and particles. Word form errors are counted those, which are not proper names or the verb to be.

The error classification in Table 3.16 accounts for approximately 60 % of all errors in the BNSI test set and 64 % if the error in the F0 class. Except for word form error and the filler "eee", the results are comparable.

The results in Table 3.16 can give us hints on how to address the problem reducing WER in speech recognition systems and the possible reduction in WER that we can expect when addressing one type of errors. From the results we can define some ideas for reducing WER:

- An enlargement of the vocabulary could reduce WER for approximately 3 % relative. However, the vocabulary should be very large. That would also slow down the speech recognition.
- A specialized language model for proper names, including an additional vocabulary could be used either for rescoring or correcting hypothesis to lower the WER on proper names. We could expect a maximal reduction of approximately 6–7 % relatively.
- Models for spontaneous speech could reduce several errors include the error for the filler *eee*. It is difficult to estimate how much WER reduction we could expect.
- A specialized language model that concentrates on word forms or word endings could be used to detect false word form and correct those errors. The maximal reduction in WER we can expect is approximately 10 %.
- A specialized language model that concentrates on minor part of speech words could be used to correct errors on those words. As those words often appear in group it may be possible to build models that handles those groups as one unit. The maximal reduction in WER we can expect is approximately 25 %.
- The same strategies may be used for correcting error of the word to be. The maximal reduction in WER we can expect is approximately 9 %.

Further work could include an extension to more features or an even more detailed analysis of errors. Possibilities are to define word lengths in different terms, to distinguish not only part of speech but also more detailed subtypes of word classes, and to introduce new type of error in the sense of a word being misrecognized for two or more consecutive words or vice versa.

The results are given only for lexical and morphological error sources. From the results we estimate that about 50 % of errors are due to spontaneous speech and acoustical conditions.

References

1. Ackroyd MH (1980) Isolated word recognition using the weighted Levenshtein distance. IEEE Trans Audio Speech 2:243–244. doi:10.1109/TASSP.1980.1163382
2. Adda-Decker M, Lamel L (2005) Do speech recognizers prefer female speakers? In: Proceedings of interspeech 2005 – Eurospeech, Lisbon, 4–8 September 2005, pp 2205–2208
3. Furui S, Nakamura M, Ichiba T, Iwano K (2005) Why is the recognition of spontaneous speech so hard? In: Matoušek V, Mautner P, Pavelka T (eds) 8th international conference, TSD 2005, Karlovy Vary, 12–15 September 2005, pp 9–22

4. Goldwater S, Jurafsky D, Manning CD (2010) Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates. Speech Commun 52:181–200. doi:10.1016/j.specom.2009.10.001
5. Grčar M, Krek S, Dobrovoljc K (2012) Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik. In: Jezikovne tehnologije 2012, Ljubljana, September 2012, pp 89–94
6. He X, Deng L, Acero A (2011) Why word error rate is not a good metric for speech recognizer training for the speech translation task? In: International conference on acoustics, speech and signal processing, 2011, pp 5632–5635
7. Huet S, Gravier G, Sebillot P (2010) Morpho-syntactic post-processing of N-best lists for improved French automatic speech recognition. Comput Speech Lang 24:663–684. doi:10.1016/j.csl.2009.10.001
8. Li X, Yang Y, Pang Z, Wu X (2015) A comparative study on selecting acoustic modeling units in deep neural networks based large vocabulary Chinese speech recognition. Neurocomputing 170:251–256. doi:10.1016/j.neucom.2014.07.087.
9. Morris AC, Maier V, Green P (2004) From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In: Proceedings of interspeech 2004, Jeju, pp 4–8
10. Siegler M, Stern R (1995) On the effects of speech rate in large vocabulary speech recognition systems. In: 1995 International conference on acoustics, speech, and signal processing, 1995. ICASSP-95, Detroit, 9–12 May 1995, vol 1, pp 612–615
11. Young SJ, Evermann G, Gales MJF et al (2006) The HTK book, version 3.4. Cambridge University Press, Cambridge
12. Žgank A, Verdonik D, Zögling Markuš A, Kačič Z (2005) BNSI Slovenian broadcast news database – speech and text corpus. In: Proceedings of interspeech 2005 – Eurospeech, Lisbon, 4–8 September 2005, pp 2525–2528

Chapter 4
Application Oriented Language Modeling

Abstract In the previous chapters we described the use of language models in speech recognition, morphosyntactic description (MSD) tagging, error rates and error analysis based on information from MSD tags. In this we will present aspects of speech recognition applications that can be related to those factors. We will first describe general consideration on speech recognition performance and whether they might or might not be important in a specific application. In the following sections we will then describe examples of more specific application and the factors to consider when measuring their performance. We will also present a procedure for language modeling for a specific application. In the last sections we will briefly describe some possible final applications in domains where errors with different lexical properties have a different impact on the recognition performance.

4.1 General Considerations

Suppose we already have a speech recognition system with a certain level of accuracy. Normally this is expressed as a word error rate (WER). While this kind of evaluation is well defined and objective, it may not be suited for the evaluation of speech recognition in a specific application.

We can start by stating a few questions about the results of a speech recognition system, specifically about different types of errors, which are classified based on lexical features:

- Are major parts of speech (nouns, adjectives, verbs, adverbs) correctly recognized or are there errors?
- Are major parts of speech recognized with the correct lemma but with an incorrect word form, i.e. in inflective languages a false ending, or are there also errors in the words lemma as well?
- Are minor parts of speech (pronouns, prepositions, conjunctions, particles) correctly recognized or are there errors?
- Are there errors where minor and major parts of speech are substituted with each other? Are there errors in which one part of speech is substituted with any other part of speech?

G. Donaj, Z. Kačič, *Language Modeling for Automatic Speech Recognition of Inflective Languages*, SpringerBriefs in Electrical and Computer Engineering,
DOI 10.1007/978-3-319-41607-6_4

- Are proper nouns (names) correctly recognized or are there errors? Are proper nouns substituted with other proper nouns or other words?
- Are numerals correctly recognized or are there errors? If there are errors are numerals substituted with other numerals or also with other pats of speech? Are numerals substituted with other types of numerals, e.g. cardinal with ordinal?
- Are filler word and other characteristics of spontaneous speech correctly recognized or are there errors? If there are errors, can these characteristics of spontaneous speech still be recognized as such or could they be mistaken for normal speech?

The next question is how many errors do occur and whether these errors matter much or little in the targeted application. Also, we have to consider more exactly what type of errors do occur. In other words which errors impact the performance of our system in the targeted application more than others.

In the above questions there was an emphasis on whether some words were substituted with the same part of speech or a different part of speech. A large rate on substitution with different parts of speech may have later an impact on how to design language models to improve performance.

An important aspect is also the further usage of the recognition output. It can be just stored without further processing, could be manually reviewed with or without making corrections, could serve as input to a speech translation system or could be used by an intelligent agent or some other computer system. In all cases it will be again necessary to know which errors are important in the further use of the recognition output.

4.2 A Weighted Word Error Rate

If we have some speech recognition results we can state either a typical error rate, like WER, or define our performance measure. If we also perform error analysis as presented in the previous chapter, we can define a more detailed formula for the error rate. WER is simply calculated over the whole test set. More details could be available by presenting insertion, deletion, and substitution rates on different parts of speech, e.g. like in Table 3.11.

WER error rate is expressed using the sum of insertions, deletions and substitutions and with the total number of words. We can express this rate as the sum of three separated error rates: insertion error rate (IER), deletion error rate (DER) and substitution error rate (SER):

$$WER = IER + DER + SER = \left(\frac{I}{N} + \frac{D}{N} + \frac{S}{N}\right) \cdot 100\,\%. \qquad (4.1)$$

Each of these individual error rates can be further expanded by taking lexical features into account, e.g. for insertions, we can express IER as the sum over all parts of speech:

$$IER = \frac{I}{N} = \frac{I_n + I_v + I_a + \dots}{N}, \tag{4.2}$$

where I_n is the number of inserted nouns, I_v is the number of inserted verbs etc.

Using parts of speech as a lexical feature to distinguish between types of errors was chosen due to the availability of part of speech or MSD taggers. Other lexical features are also possible if appropriate tools are available.

In the next step we can assign weights to each number of errors, e.g. α_{In} for I_n, α_{Iv} for I_v etc. We can now define a weighted WER as:

$$WWER = \frac{\alpha_{In}I_n + \alpha_{Iv}I_v + \dots + \alpha_{Dn}D_n + \dots + \alpha_{Sn}S_n + \dots}{N} \cdot 100\,\%. \tag{4.3}$$

By fixing all weights α to 1, we would get the standard WER or in combination with the modified Levenshtein distance from the previous chapter the modified WER. By assigning different weights we can adjust the WWER to the needs in the targeted application.

Similarly, we can define a weighted lemma error rate for applications where we are not as much interested in the recognized word form. Further more we can also define other types of a weighted WER, where types of errors are divided by different criteria, e.g. application determined classes of words.

4.3 Sample Applications

Now we will describe some hypothetical sample applications and how an appropriate performance measure shall be defined for them. We will also briefly describe ideas for modifying language models and the parameters of a speech recognizer for an improved performance.

4.3.1 Simple Speech Transcription

We start with simply a LVCSR system whose output shall be the final result. The purpose of such a system may be speech transcription of TV or radio shows [7] for the deaf or other people who will for some reasons read the transcribed speech.

We want a measure that estimates the performance as perceived subjectively by the reader, i.e. how understandable are the automatic transcriptions. For this, we could use the weighted WER as described above. First we have to determine how to define types of errors. For Eq. (4.3) we considered insertions, deletions and substitution of parts of speech. We have to decide if this classification is also appropriate in our application.

Based on our knowledge and experience with speech recognition systems and their typical errors as well as our own subjective perception of errors, we can state a series of hypotheses about the impact of different errors, which will be later the basis of error classification:

- Deleted or substituted words that carry a semantic meaning to the sentence might have the highest impact on the perceived performance.
- However, if the right lemma is recognized and only the word form is not right, the impact will be smaller.
- Inserted word with a semantic meaning will be distracting, but might be well ignored by the reader.
- Deleted short words that have only a grammatical role may have a smaller impact, as the reader will often know which word is missing.
- Inserted short words that have only a grammatical role may also have smaller impact as they are easier to ignore.
- Deleted fillers and false starts will not be an issue as the reader does not need the to understand the text.
- Inserted filler should also to be an issue as they are easy to recognize by the reader.
- Inserted false start could be distracting as they could be mistaken for some other, shorter word.
- Etc.

Based on the above stated hypotheses we could define a classification of errors based on the basic type of error (insertion, deletion or substitution), whether the word carries semantic information, whether the word is a characteristic of spontaneous speech (fillers, false starts), etc. We will then modify Eq. (4.3) based on this classification.

Next we would need estimates for each weight in our adapted equation for WWER. Obtaining them would involve more extensive experiments including human evaluation of speech recognition results and error impacts. This will be a rather time consuming and possibly expensive process.

In the course of weight estimation we might find some of the stated hypotheses inappropriate. As a result we should discard them, possible state new hypotheses and repeat the process. We could also find that some types of errors could by further classified into smaller classes. The whole process might even require several iterations for a perfectly adequate classification. In this process we will have to decide if a better classification if necessary and if the possible benefits would justify the higher development time and costs.

4.3.2 Dictation System with Manual Correction

Let us consider a system designed for accurately transcribing spoken speech in an unlimited (open) domain [6] or a limited domain [4]. The most natural way would be to design it for a minimal number of total errors. Since we can not expect a system to be without any errors, the recognition output should be manually corrected if necessarily. This is time consuming and expensive. Proper tools able of simplifying the correction process will be of course needed, but the correction of different errors will still take different time effort.

In practice, an appropriate measure of performance shall be: how much time or effort is needed to review speech recognition output, manually compare it with the audio recording and correct all errors.

If the speech on the audio recording is rather spontaneous we have to take into account that we do not want repetitions and fillers, which are a characteristics of spontaneous speech, to be present in the final transcription. In this case we must consider a cleaned transcription of the spoken sentences as the end result and measure the effort to correct errors until this transcription is achieved, rather that a transcription more faithful to the actual spoken sentence. In this case we will also need well defined rules how the final text shall differ from the spoken text. Removing repetitions and filler words like *ehm* can be taken for granted while more complicated correction might not be.

The time effort can of course be measured with experiments on test material. If we want to compare different speech recognition systems with different language models to determine which one is better, the experiment has to be repeated for every system. This poses difficulties in the final evaluation. Using the same people for evaluating the performance several times on the same speech segments is problematic as with time the person evaluating will know the sentences from memory, thus distorting the results. Using different groups of people on different systems we would also take differences in the groups into account.

An alternative is to evaluate how much effort is needed for certain types of errors, e.g. deleting falsely inserted words, inserting falsely deleted words, correcting word endings, correcting whole words, merging falsely separated words, etc.

With an estimate of the impact of error types on the correction effort a total score of performance can be defined. Let N be the number of different error types, E_i the estimated effort needed to correct errors of type i, N the number of different types, n_i the number of errors of type i; the total effort can be expressed as

$$E = \sum_{i=1}^{N} E_i \cdot n_i. \tag{4.4}$$

Also considering that the effort to correct words can depend on the word length we can expand the measure to

$$E = \sum_{i=1}^{N} L(E_i)_i^{\alpha} E_i \cdot n_i, \quad \alpha = 0, 1 \tag{4.5}$$

where $L(E_i)$ is the length of the word in error E_i. The length has only effect on those types of errors where we specify it, i.e. α_i is 1. For example the effort of deleting a word is independent of its length, whereas inserting it is dependent.

Similar as with the example if simple speech transcription we again might find that our classification of errors can be improved, so that this evaluation can also be iteratively repeated until a satisfactory classification is achieved.

The goal for the developing of the used speech recognition system is to minimize E. Examining the individual terms in the above sum we can see which errors mostly contribute to the total effort. If it is the need to insert short words that we deleted in the speech recognition system it may be prudent to improve the recognition of those words with an appropriate language model. Short words are mostly minor parts of speech. Later in this chapter we will describe a model for improving the recognition of those words. An important factor for the optimization in this application may be the language model weight and the word insertion penalty.

Lowering the effort to correct a certain type of errors might increase the effort for another type. Finally, an optimized balance has to be chosen.

4.3.3 Recognition of Numerals

Let us consider a system designed to perform LVSCR on news broadcast where the emphasis is on financial news or some other topic involving larger amounts of numbers. The output of the recognition system is the further used by a computer system that searches for certain names and numbers appearing in the same speech segment. The performance of such a system would not be affected by the false recognition of grammatical words, but mostly from the false recognition of numerals.

Recognizing numerals is a challenge because of the large number of different numerals and the fact that they are mostly written with digits instead of words [1]. We need them to be written as words in the dictionary to make a phonetic transcription. Also numerals are inflective parts of speech, therefore the conversion from digits to words is nontrivial. To make a correct conversion we need to know what type is the numeral: cardinal, ordinal or some other type. After that we need to know the gender, case and number to determine the correct word form.

One more difficulty of numerals is the large number of different numbers that are each expressed with different words. Larger numbers are the still expressed with several words. We can estimate that we need about 450 different words to write all Slovene cardinal numerals used in everyday speech. Slovene ordinal numerals are always written in one words. For having each word form for numbers from first to one millionth, we would need about 11 million words (on average 11 word forms for each cardinal numeral).

The third problem is data sparsity. Having 450 different word forms for writing cardinal numerals, we can not expect all of them to be in the training corpora in a sufficient amount for a good probability estimate in the language model. Consequently, numerals could have very different probabilities although this would negatively impact the performance of the application.

A type of language model that deals with the modeling of numerals in an inflective language is presented in [1].

Again we can use and the WWER for scoring such a system and possibly adapt if further considering the role of individual numerals from the target domain. Another score that even more emphasizes the recognition of numerals can be defined as simply the ratio between the number of correctly recognized numerals and the number of all numerals—a numeral error rate.

In both cases a MSD tagger could be used to identify numerals. However, numerals are quite unique words and a rule based algorithm can be easily implemented to write a list of the numerals of a language. A simple look up if a word is on the list can be sufficient. Also, we must decide if it is important that the word is in the right form (gender and case) or not before calculating the performance.

4.3.4 Recognition of Proper Nouns

Recognition of proper nouns (proper names) is somewhat similar to recognition of numerals. Again we deal with a large number of possible words that may not occur in the training corpora. In this case we could build classes of names: first names, last names, geographical names (cities, countries, etc), company names etc. By taking the modeling of numerals in [1] as an example we could build class based language model and the model the probabilities of classes. Inside classes the probabilities of individual words could be equal or be modeled according to the needs of our application. This approach might be interesting in a limited domain like an automatic information system for bus lines or railways, where names of cities are expected.

4.3.5 Speech Translation

In a speech translation system [3] a speech recognizer, a machine translator, and a speech synthesizer are combined. The output from the speech recognizer is processed by the translator. The main purpose is a good translation, thus performance should be evaluated with a score intended for evaluating translators between the spoken text and the translated text. Typical scores for translation are BLUE and Translation Error Rate. The score depends on the performances of both the speech recognition system and the translation system. But they may not be a direct correlation between a lower WER and a better translation score. WER is influences by inserted words and deleted words. We would have to find out how inserted and deleted words afterward effects on the translation performance. If the effects are different, than a weighted WER will be more appropriate. Also, we might consider analyzing the impacts of recognition errors on different parts of speech.

4.3.6 Keyword Spotting in LVCSR Results

Let us consider a system designed to process a set of audio recordings, perform large vocabulary continuous speech recognition (LVCSR) and identify those speech segments that contain one or more specified words. Often other methods are used in keyword spotting, like acoustic and phonetic keyword spotting, in which the keywords are searched for specifically in the audio recordings [5]. We are presenting a hypothetical application that is based on performing LVCSR first and the search for keywords inside the transcriptions.

Keyword spotting can be viewed as a binary classification problem. Either a speech segment does or does not contain the specified keyword. The overall accuracy of such a system can be simply described as the ratio between the number of correctly classified speech segments and the number of all speech segments.

Often the overall accuracy is not as important as the number of false positive errors (detecting a keyword while it was not present), false negative errors (not detecting a keyword while it was present) or the ratio between them. An often used measure for keyword spotters is the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate as a function of the false positive rate. An example of an ROC curve is in Fig. 4.1. Different points on the curve present the results obtained while using different system parameters. The goal is to find the results that best suite our needs. Ideally, the results have to be as far as possible in the upper left corner.

The ideal results in an application can be found by defining weights $\alpha < 0$ and $\beta > 0$ for the true positive rate (TPR) and the false positive rate (FPR) respectively and a final score as:

$$S = \alpha \cdot TPR + \beta \cdot FPR. \tag{4.6}$$

In an alternative definition of a score, we have to decide how important are false positive errors, i.e. can we afford to waste time to check if a speech segment really contains the keyword, and how important are false negative errors, i.e. can we afford to miss a speech segment that contains the keyword. A simple solution would be to define two weights, $\alpha > 0$ and $\beta > 0$, for the rates of false positive and false negative rates (FNR) respectively and then to define a total score as:

$$S = \alpha \cdot FPR + \beta \cdot FNR. \tag{4.7}$$

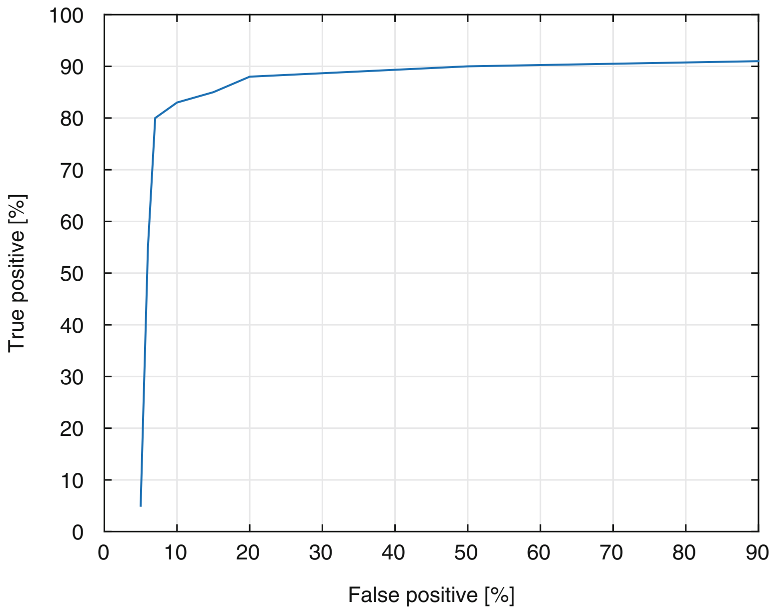


Fig. 4.1 A hypothetical ROC curve for a keyword spotting system

The task of determining the two weights in either of the two mentioned scores would be strongly application dependent.

Language modeling in such a keyword spotting application firstly involves the use of the right vocabulary. If the keywords are already in the used vocabulary and we already performed recognition, its results can be used. If the keyword is not in the vocabulary, it must be added to the vocabulary. In this case we can again build a language model from the corpus and than perform speech recognition. A requirement for this is that the keywords are present in the training corpus. As vocabularies are typically assembled with the most frequent words, missing keywords will rarely appear in the corpus, thus having later small probabilities in the language model.

Building a language model takes some time and if have to do this very often it might be wisely to consider adapting existing language models. A possible example will be presented in the next chapter.

In inflective language an important point to consider is whether we want to find the right term in any word form or we want to find the exact word form we are looking for. On this point a language model designed to improve the recognition of the right lemma becomes interesting, although it might not improve the recognition of the right form it may improve the recognition of lemmas. Interesting keywords are also nouns and maybe verbs. So the performance of correctly recognizing minor parts of speech is of less importance.

Another point in keyword spotting is the use of N-best list instead of only one hypothesis as the recognition output. If it is crucial to have as few as possible false negative errors, a speech segment can be marked as positive if the keyword appears in at least once or a certain number times in the hypotheses in the N-best list. On the other hand, this will increase the number of false positive errors. Therefore a compromise has to be achieved by determining the total number of hypotheses to be considered.

4.4 A Procedure for Application Oriented Language Modeling

Suppose we have a working speech recognition system as a basis for the development of application specific systems. We want to adapt the performance scoring, vocabulary, language model, and recognition parameters to the targeted applications. We present a procedure for doing so based on the material presented so far. A full example application of how to perform these steps will be shown in the next chapter.

1. We take the existing speech recognition system. If possible and needed we will adapt its acoustical training set with application specific material. We also need an application specific development set and test set for the later optimization of recognition parameters and evaluation.
2. We perform speech recognition on both the development set and the test sets. Speech recognition parameters like the language model weight and word insertion penalty are optimized using the development set.
3. We evaluate the results. So far we have only performed typical steps for speech recognition. In error analysis we shall not only state results as simply WER, but rather also define an application specific performance measure similar to those defined in the previous section. This should be implemented as an automatic process, since we will need it several times later. If this measure involves the distinction between parts of speech and grammatical categories then MSD tagging of all speech recognition outputs has to be performed first.
4. If the weights of inserted and deleted words could be different, optimization of language model weighted and word insertion penalty shall be repeated as changing their values can have a significant effect of different error types.
5. In determining the value of the new performance measure we shall also perform are more detailed analysis as presented in Chap. 3. The final analysis shall describe the most critical type of errors for the final application.
6. Based on the performed error analysis we shall now define ways to improve performance by adapting and/or adding vocabularies and language models. This is probably the most difficult part of the procedure since it involves understanding of the process of speech recognition, the role of the language model in it as well as the adaption to the target application. In order to build a better language model

we must identify flaws in the language model and devise a new language model to eliminate them.

7. Using the new language model we again perform speech recognition and compare the results. Again an optimization of the speech recognition parameters might be needed. If it would be necessary, we can repeat steps 5–7.

4.5 Examples of Specific Language Modeling

We will show some examples of how language models other than tradition word based *n*-gram models can be build. In each example we first state a category of words of speech for which the language modeling technique is intended. We then continue to specify the structures of these language models.

4.5.1 Spontaneous Speech

Spontaneous speech is problematic in speech recognition because of the differences to prepared or read speech and especially compared to written text, on which language models are often build [2].

A characteristic of spontaneous speech are speech disfluencies. Prepared speech often consist of well formed sentences similar to written text. Furthermore, spontaneous speech is also sometimes characterized by not being grammatically correct. It is therefore harder to recognize as language models are build on written text. Spontaneous speech is also characterized by fillers, false starts and repetitions which make it different from written text.

Filler words (hum, ehm, aha etc.) stand in a spoken sentence in between other words. Removing the filler will result in just the same sentence only spoken more fluently. For the modeling of filler words we can define a class of such fillers and observe their behavior in spoken language.

In the sense of language modeling we have to find out after which words do these fillers appear. We have to use a spoken database. Since such databases are considerably smaller that written corpora we can not expect a good probability estimation for filler based on previous words. A detailed analysis of the appearance of fillers and their preceding words might give us some clues as how to better estimate probabilities for fillers.

With the assumption that the part of the sentence behind the filler simply continues the sentence before it, we can build a rule into our language model:

$$P(w_i|w_{i-1},\ldots,w_{i-n+1}) = P(w_i|w_{i-2},\ldots,w_{i-n+1}) \tag{4.8}$$

This rule shall be used if w_1 is a filler word. Incorporating the rule in above equation into an existing language model would reduce the language model order.

To overcome this, we could expand the conditional probability to include the word W_{i-n}.

For the probability of the filler word itself ($P(w_{i-1})$) a class based approach could be used, e.g.:

$$P(w_i|w_{i-1},\dots,w_{i-n+1}) = P(C|w_{i-1},\dots,w_{i-n+1}) \cdot P(w_i|C). \tag{4.9}$$

Here all typical fillers are combined into the class C. The class probability can be estimated based on spoken corpora. For the probability of the filler given the class, we also shall consider that different speaker use different fillers. Therefore, the probability of the speaker to use a particular filler is higher that what would be suggested by a spoken corpus with several speakers.

A similar rule to that in Eq. (4.8) can be considered for repetitions. Here a word is spoken twice in a row. In spontaneous speech in an inflective language a speaker might say a word with the wrong ending (grammatical error) and then immediately corrects himself. This results in two different words but with the same lemma. In this case we can again use Eq. (4.8), if words w_{i-1} and w_{i-2} are equal or at least have the same lemma.

Modeling spontaneous speech can be useful in a dictation system, but since we do not want repetitions and fillers in the final results an additional step is necessarily to remove them.

4.5.2 Modeling Lemmas and MSD Tags

To build a language model for lemmas is rather straight forward. The training corpus has to be lemmatized and a lemmatizer has to be available inside the recognition processes. This is probably most easily achieved by using a two-pass recognition system. In the first pass a N-best list of hypotheses can be generated. All hypotheses are then lemmatized and a language model for lemmas is used in the second recognition pass.

In the second pass the score (in form of the logarithm of probability) of a hypothesis W can be calculated as

$$\log P(W) = \log P_A(W) + \alpha \log P_W(W) + \beta \log P_L(W') + \gamma N. \tag{4.10}$$

Here P_A is the probability of the acoustic model, P_W the probability of the word based language model, P_L the probability of the lemma based language model and N is the number of words. The coefficients α and β are the language model weights, while the acoustic model weight is 1; γ is the word insertion penalty. We used W' to denote a modified sentence, in this case we replaces each word with its corresponding lemma.

One possibility to implement such a model is to use the concept of factored language models. We define two factors: word and lemma. Every word in the training corpus us factored and presented in this form. Language models are build using appropriate tools and the factored language model is finally used on the N-best list. Using factored language models also opens the possibility of building language models that used mixed factors each.

Using a lemma model might be appealing in applications where we want to identify the right concept that is expressed with a word in any of its forms or in other words when we are looking for the right basic word despite different endings. A sample application is keyword spotting.

The total number of lemmas is smaller than the total number of words, thus enabling better estimates of probabilities, less data sparsity for higher n-grams and therefore the chance of building a higher order n-gram model that outperforms the lower order model, despite the fact that the same order word based model does not perform better.

In a very similar way we can use a MSD tagger instead of or in combination with a lemmatizer. We define the factors: words, lemma and MSD tag. Again factored language models can be build upon these factors. Since MSD tags contain grammatical information usually present in the word ending, a model employing those tags would be designed specifically to improve the recognition of exact word forms.

Using a MSD tagger and lemmatizer introduces another possible source of error in the system as those taggers do not have 100 % accuracy. Even the accuracy reported in research papers, can not be taken as a reliable estimate, as they are usually calculated on written text.

An example on using MSD tags in a similar way in speech recognition of French is presented in [21]. The authors reported improvements in WER and lemma error rate with the best improvements on LER on lexical (mostly inflective) words.

4.5.3 Modeling Minor Parts of Speech and Other Short Words

In the error analysis in Chap. 3 we saw that many errors are associated with short words. Those words typically belong to minor parts of speech (interjections, pronouns, etc.), with a notable exception being different forms of the verb *to be*.

There is a limited number of all words that belong to minor parts of speech and a limited number of word forms of *to be*. Thus, we can define a vocabulary of those short words to which we add tags for other words. This tag can be the same for all words, different based on part of speech, different based on the whole MSD tag, or different based on some other criteria.

This vocabulary would have a very small size compared to vocabularies typically used in LVCSR. A language model build on that vocabulary would therefore be more robust as we would have more data available for estimating individual probabilities. We would be also able to build higher order language models that would involve longer distance dependencies between words.

The resulting language model can be incorporated into a rescoring procedure, similar to that presented for lemmas and MSD tags in Eq. (4.10).

Since this model would include mostly grammatical parts of speech its use would be in applications where we are either interested in the structure of spoken language or in reducing the overall WER, where not so much emphasis is given on lexical words. Another possibility would be to use it in combination with a model for lemmas.

4.6 Summary

We presented several possible aspects of speech recognition results that can be considered in a practical application and also a proposal for a weighted WER that is based on basic types of errors and parts of speech. We can not cover all possible applications with this or any other rule for estimating the performance of a speech recognition system.

Other definitions of a performance rate are also possible and we showed some specific examples. One, who is faces with the challenge of building a speech recognition system for a specific application, has to determine the most appropriate and still feasible performance rate.

The important step in these cases is to determine which errors are important for the targeted application and which are not. A classification of errors is therefore necessary and our the presented classifications were based on lexical features as they will probably give the best clues on how to build or adapt language models to improve performance.

Our main goal was to present a procedure where building and adapting language models is used for a specific application, that incorporates a given performance rate. The processes of determining the appropriate structure, learning data, learning process and incorporation of specific language models is application dependent. We presented a few possibilities of which words could be interesting with proposition on how to build specific language models. A full example on how to use the topics presented in this chapter will be explanation in the next chapter.

References

1. Donaj G, Kačič Z (2014) Manual sorting of numerals in an inflective language for language modelling. Int J Speech Technol 17:281–289. doi:10.1007/s10772-014-9231-y
2. Furui S, Nakamura M, Ichiba T, Iwano K (2005) Why is the recognition of spontaneous speech so hard? In: Matoušek V, Mautner P, Pavelka T (eds) 8th International conference, TSD 2005, Karlovy Vary, 12–15 September 2005, pp 9–22
3. Hashimoto K, Yamagishi J, Byrne W, King S, Tokuda K (2012) Impacts of machine translation and speech synthesis on speech-to-speech translation. Speech Commun 54:857–866. doi:10.1016/j.specom.2012.02.004

4. Petrik S, Drexel C, Fessler L, Jancsary J, Klein A, Kubin G, Matiasek J, Pernkopf F, Trost H (2011) Semantic and phonetic automatic reconstruction of medical dictations. Comput Speech Lang 25:363–385. doi:10.1016/j.csl.2010.07.003
5. Sangeetha J, Jothilakshmi S (2014) A novel spoken keyword spotting system using support vector machine. Eng Appl Artif Intell 36:287–293. doi:10.1016/j.engappai.2014.07.014
6. Steinbiss V, Ney H, Essen U, Tran BH, Aubert X, Dugast C, Knesser R, Meier HG, Oerder M, Haeb-Umbach R, Geller D, Höllerbauer W, Bartosik H (1995) Continuous speech dictation – from theory to practice. Speech Commun 17:19–38. doi:10.1016/0167-6393(95)00012-D
7. Woodland PC (2002) The development of the HTK broadcast news transcription system: an overview. Speech Commun 37:47–67. doi:10.1016/S0167-6393(01)00059-0

Chapter 5
An Example Application

Abstract In this chapter we will present an example of how some of the techniques and ideas presented in the previous chapters can be applied. We will do this in form of a full example. We will perform LVCSR on a Broadcast news database and search for keywords inside *N*-best list results. After a first run we will present optimization to improve the performance of the keyword spotter. Later we will also present some modifications to the language model to improve performance.

5.1 The Target Application

Our target application is the search for any number of arbitrary keywords in a daily broadcast news show. The set of all the keywords will be determined after the speech recognition process and can arbitrary expanded.

When in an acoustical based keyword spotting system the number of different keywords is rising the rate of speech segments that need to be fully transcribed will be also rising, therefore increasing the time demands to the process. As we want to be able to expand the set of keywords and get new results as quickly as possible, we will perform speech recognition on the whole test set. We can afterwards perform the search for keywords in the transcriptions. Although this method is rather slow in the first step (speech recognition), it can later save time, if we want to modify our search parameters several times. This scenario is plausible in an application that is supposed to record and transcribe daily news shows and will be later accessed by different clients, who all have a unique set of keywords and demand a quick performance.

The application will be used in an inflective language so that for any given keyword all inflective forms must be considered. The keywords can be uttered by any speaker, in prepared or spontaneous speech, with or without acoustic background. We will therefore need an appropriate test set.

As a final result of our system, we want the transcriptions of speech segments in which keywords were detected. The results will later be manually reviewed and false positive segments can be at that stage eliminated. Therefore, we want to concentrate on minimizing the number of false negative results.

G. Donaj, Z. Kačič, *Language Modeling for Automatic Speech Recognition of Inflective Languages*, SpringerBriefs in Electrical and Computer Engineering,
DOI 10.1007/978-3-319-41607-6_5

Based on those requirements we will present an experimental setup, with examples of keywords with obtained performance results and methods for improving performance.

5.2 Experimental Setup

For the targeted application, broadcast news type databases are appropriate. They are available in several languages, including some inflectional languages. We will use the Slovene BNSI database [6]. This database consists of a train set (used for acoustic model training), a development set (used for parameter tuning) and a test set (used for evaluation).

The database is manually segmented. As features we used 13 MFCC [2] base coefficients as well as delta and delta-delta coefficients. We have built state-tied Gaussian mixture HMMs as acoustic models.

For language model training the 600 million word FidaPLUS[1] corpus was used. Basic bigram and trigram language models were build using vocabulary sizes from 60,000 to 300,000 words. Good-Turing smoothing and the Katz Back-off algorithm were used in the models. In some previous experiments we have determined that in our system Good-Turing smoothing performs slightly better than the mostly superior modified Knesser-Ney smoothing.

The test set consist of approximately 22,000 words. The speech segments are categorized with different F-classes [3]. That way prepared and spontaneous speech are present in the test set as well as speech with acoustic background. Training and decoding was done using the HTK toolkit [5].

A set of ten different keywords with a total of 230 occurrences in the test set was manually selected. All words are major parts-of-speech that have inflectional forms, and have different frequencies of occurrence in the test set. The words are listed in Table 5.1 along with their frequencies in the test set and the English translation. The frequencies are calculated for all of the words forms.

Table 5.1 Selected keywords

Word	Frequency	English translation
Slovenija	92	Slovenia
predsednik	32	President
minister	26	Minister
sodišče	20	Law court
mednaroden	19	International
republika	12	Republic
Kučan	11	Kučan
Hrvaška	8	Croatia
kloniranje	8	Cloning
Nemčija	2	Germany

5.3 First Results

Of all the build language models we used the 300,000 word trigram model in our final experiments. The overall WER rate on the test set is 25.64 %. The recognition results for the given keywords are significantly better. This is not unexpected as most recognition errors occur with short words and minor parts-of-speech.

Based on methods described in Chap. 3, we analyzed the results. WER on the keywords is 8.70 % and the lemma error rate (LER) is 6.09 %. All results were obtained using only the best hypothesis for each speech segment. Assuming that in a practical application the user will be interested not only in the basic word form, but in any inflected form of the given keyword, it is more appropriate to use results based on lemmas. The basic results are given in Table 5.2. As we see there are no deletions or insertions of the selected keywords. All errors are either substitutions of the keyword with another inflected form of the same keyword, which we later ignore, or substitutions of completely different words. In terms of binary classification we can say that we have a 4.35 % false negative rate (TPR) and 1.74 % false positive rate (FPR) in word level. The results are relative to the number keyword appearances in the test set.

However, if the final application is to detect sentences that contain a given keyword then sentence or speech segment based scoring will be more appropriate. The results shall also be given relative to the number of speech segments. The given keywords appear in 190 different speech segments. The number of speech segments with keywords is smaller as the number of keyword appearances as sometimes more the one keyword appears in a single speech segment. The results on sentence level are in Table 5.3.

We see that 179 speech segments were correctly identified as containing a keyword, which gives us an accuracy of 94.21 %. The results of false positive and false negative speech segments approximately correspond to the number of false positive and false negative keywords. The relative rates in Table 5.3 are calculated with regard to the number of all sentences in the test set.

Table 5.2 Basic results for the selected keywords

Total number of all words	22,743
Total number of all keywords	230
Correctly recognized keywords	204
Deletions of keywords	0
Insertion of keywords	0
Substitutions of keywords with other word form of same lemma	16
Substitutions of keywords with other words (different lemma)	10
Substitutions other words (different lemma) with keywords	4
False positive rate (lemma)	1.74 %
False negative rate (lemma)	4.35 %

Table 5.3 Basic results for the selected keywords

Total number of all sentences	1898
Total number of sentences with keywords	190
False positive sentences	2
False negative sentences	9
True positive rate	95.26 %
False positive rate	0.12 %

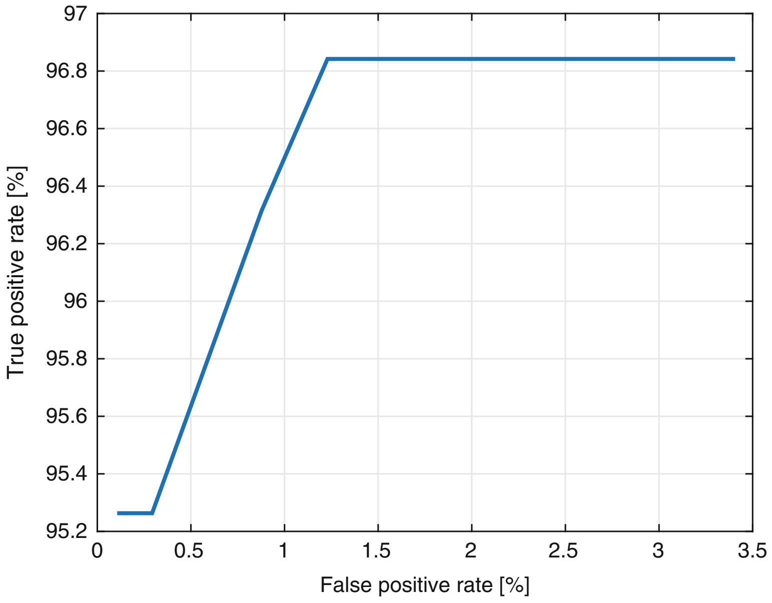


Fig. 5.1 The ROC curve for using several hypotheses

5.4 Improving Results

The first way to improve results is to use several of the best hypotheses instead of the single best one. We denote a sentence positive if at least one hypothesis contains the selected keyword, otherwise the sentence is denoted negative. Using 1, 2, 3, 5, 10, 20, 50, and 100 hypotheses the number of falsely negative sentences decreases from 9 to 6, while the number of falsely positive recognized sentences increases in steps from 2 to 58. Those changes are represented in the ROC curve in Fig. 5.1. Depending on the final application we can then determine the best number of hypotheses to use. If we aim to maximize the true positive rate, as false positive results may be later manually discarded, we may want to select the option of using ten hypotheses as we get a true positive result of 96.84 %, while having 1.23 % false positive rate.

The selection of the number of hypotheses can be more formally expressed by defining a final scoring function based on Eq. (4.6), where a negative weight is assigned to the true positive rate and a positive weight to the false positive rate. The final task is to find the number of hypotheses that minimizes the scoring function.

A second possibility is to use different language model weights and word insertion penalties in the speech recognition run. The weights can be optimized based on results and a scoring function. Based on whether in our target application false negative or false positive errors are more critical or if both types are equally critical. Furthermore, those weights are typically optimized to minimize the overall word error rate on the test set. In our example application we may assume that keywords we most likely be only major parts-of-speech and an error rate on those words should be minimized. Therefore, the weight optimization process should be modified to that end.

In our example changes in language model weight and word insertion penalty gave no significant changes in results and therefore optimization was difficult. However, based on a simple example we can not conclude that changing weights will not have an effect in other application, or even the same application with different basic results.

5.5 Language Model Adaptation

If we consider that false negative errors are more critical that false positive errors, we may alternatively want to perform some sort of language model adaptation that will decrease false negative errors. We can modify a language model and repeat the speech recognition process. However, this method will result in great time demands, as speech recognition must be repeated for any new set of keywords. The other possibility is to perform speech recognition once with a rather long N-best as a result. Hypotheses can be later re-scored with new language models. As this rescoring involves only text processing, it is much faster than speech recognition.

Two methods for increasing the probability of selected keywords in a language model are to increase the probability in the already build model or to perform an interpolation of results with the first and a new language model. The first methods involves the modification of counts of n-gram occurrences. We can artificially increase the number of keywords and n-grams containing them before we build a new language model.

In the second method we can express the final score of a hypothesis with

$$\log P(O|W) + \alpha \log P_1(W) + \beta \log P_2(W) + \gamma N, \qquad (5.1)$$

where P_1 and P_2 are probabilities obtained with two different language models and α and β are their corresponding language model weights. In our experiments the acoustic model weight was 1, the language model of the first language model was 15 and the word insertion penalty was -6.5.

We have added a second recognition pass to our system based on the above equation. The second language model was an unigram model. That model was first build on the same corpus as the original trigram model. It was later modified to increase the probability of the given keywords with different amounts of probability.

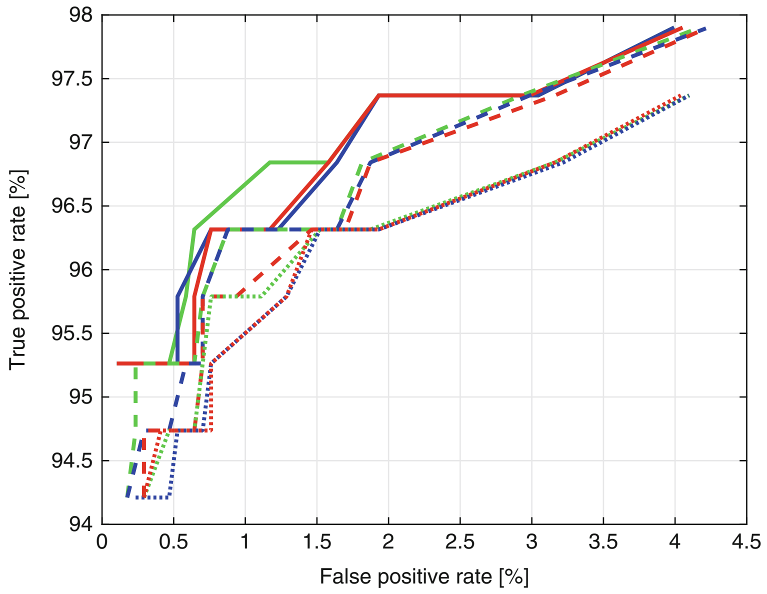


Fig. 5.2 The ROC curve for using different language model weights

The language model weight for the first language model and the word insertion penalty from the first recognition pass were unchanged in the second pass. We have modified the weight for the second model. We again have drawn a several ROC curves for the results obtained with different weights of the language model. Each curve consist of points obtained with different modifications of the probability of keywords. These curves are shown in Fig. 5.2.

We see again that we can obtain best results by using a certain language model weight and model modification. We now have to select weights for the scoring function. Let us consider that false negative errors are ten times more critical than false positive errors. We are the able to form a scoring function and determine its minimum. In our case, we the obtained the minimum point language model weight of 0.5, while the probability of keywords in the language model was increased by a factor of 100. Finally, we achieved a false positive rate of 1.17 % and a true positive rate of 96.84 %.

We can conclude that both methods have shown a performance increase in our application. The final results obtained with the modified language model was better as it had a lower false positive rate, although the difference was negligible. Another possibility might be to combine both methods.

5.6 Conclusion

In this book we concentrated on analyzing speech recognition performance based either on lexical data by differencing between different word forms or on a given application based on a specific scoring function.

This chapter showed a short example on how to use modifications to the language model to improve performance in a specific application with a given scoring function. An example on how to modify language model to increase performance given a specific set words can be found in [4]. This example was held rather short as we want to encourage the reader to thing about methods to further improve the performance.

Any reader who is interested in a more detailed analysis of a given speech recognition system or in the use of a speech recognition system for a specific application can use this book and this example as a starting point for ideas on how to implement language models and evaluate the final performance.

References

1. Arhar Š, Gorjanc V, Krek S (2007) FidaPLUS corpus of Slovenian: the new generation of the Slovenian reference corpus: its design and tools. In: Davies M (ed) Proceedings of the corpus linguistics conference, Birmingham, 2007, pp 27–30
2. Biem A, McDermott E, Katagiri S (1996) A discriminative filter bank model for speech recognition. In: Proceedings of the IEEE, ICASSP-96, Atlanta, May 1996, pp 545–548
3. Chen SF, Goodman J (1999) An empirical study of smoothing techniques for language modeling. Comput Speech Lang 13:359–394. doi:10.1006/csla.1999.0128
4. Donaj G, Kačič Z (2014) Manual sorting of numerals in an inflective language for language modelling. Int J Speech Technol 17:281–289. doi:10.1007/s10772-014-9231-y
5. Young SJ, Evermann G, Gales MJF et al (2006) The HTK book, version 3.4. Cambridge University Press, Cambridge
6. Žgank A, Verdonik D, Zögling Markuš A, Kačič Z (2005) BNSI Slovenian broadcast news database – speech and text corpus. In: Proceedings of interspeech 2005 – Eurospeech, Lisbon, 4–8 September 2005, pp 2525–2528