VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MINING OF TEXTUAL DATA FROM THE WEB FOR SPEECH RECOGNITION

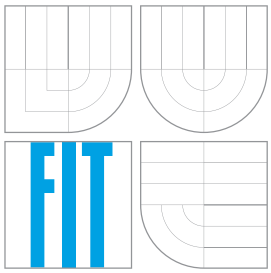
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB KUBALÍK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZÍSKÁVÁNÍ TEXTOVÝCH DAT Z WEBU PRO ROZPOZNÁVÁNÍ ŘEČI

MINING OF TEXTUAL DATA FROM THE WEB

FOR SPEECH RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB KUBALÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ MIKOLOV

BRNO 2010

Abstrakt

Prvotním cílem tohoto projektu bylo prostudovat problematiku jazykového modelování pro rozpoznávání řeči a techniky pro získávání textových dat z Webu. Text představuje základní techniky rozpoznávání řeči a detailněji popisuje jazykové modely založené na statistických metodách. Zvláště se práce zabývá kriterii pro vyhodnocení kvality jazykových modelů a systémů pro rozpoznávání řeči. Text dále popisuje modely a techniky dolování dat, zvláště vyhledávání informací. Dále jsou představeny problémy spojené se získávání dat z webu, a v kontrastu s tím je představen vyhledávač Google. Součástí projektu byl návrh a implementace systému pro získávání textu z webu, jehož detailnímu popisu je věnována náležitá pozornost. Nicméně, hlavním cílem práce bylo ověřit, zda data získaná z Webu mohou mít nějaký přínos pro rozpoznávání řeči. Popsané techniky se tak snaží najít optimální způsob, jak data získaná z Webu použít pro zlepšení ukázkových jazykových modelů, ale i modelů nasazených v reálných rozpoznávacích systémech.

Abstract

The preliminary goals of this project were to get familiar with language modeling for speech recognition and techniques for acquisition of text data from the Web. Speech recognition techniques are introduced and statistical language modeling is described in detail. The text also covers mining models and techniques, information retrieval especially. Specific problems of Web mining are discussed and Google search is introduced. Special attention was paid to detailed description of implementation of the text mining system. However, the main goal of this work was to determine, whether the data acquired from the Web can provide some improvement into the recognition systems. The text is describing experiments, which use the retrieved Web data to update sample language models.

Klíčová slova

Rozpoznávání řeči, Rozpoznávání spojité řeči s velkým slovníkem, Bayesova teorie pravděpodobnosti, Jazykový model, Apriorní pravděpodobnost, Klasifikace do ekvivalentních tříd, N-gram, Smoothing, Entropie, Perplexity, Podíl OOV slov, Word Error Rate, Anotační data, Korpus, Lineární interpolace, Dolování dat, Vyhledávání informací, TF-IDF váha, Dolování webu, Dolování textu, Google PageRank

Keywords

Speech recognition, Large Vocabulary Continuous Speech Recognition (LVCSR), Bayes' probability theory, Language model, A-priori probability, Equivalence classification, N-gram, Smoothing, Information Theory, Entropy, Cross-entropy, Perplexity, Out of Vocabulary rate, Word Error Rate, Annotation data, Corpus, Linear Interpolation, Data mining, Information retrieval, TF-IDF, Web mining, Text mining, Google's PageRank

Citace

Jakub Kubalík: Mining of Textual Data from the Web
for Speech Recognition, diplomová práce, Brno, FIT VUT v Brně, 2010

Mining of Textual Data from the Web for Speech Recognition

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Tomáše Mikolova

.....
Jakub Kubalík
May 25, 2010

Poděkování

Tímto bych rád poděkoval panu Ing. Tomáši Mikolovi za odborné vedení práce, rady a čas, který mi během vypracovávání této práce věnoval. Dále bych rád poděkoval panu Ing. Michalovi Fapšo za vřelou pomoc při prováděných experimentech.

© Jakub Kubalík, 2010.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Contents

Chapter 1

Introduction

The first application of speech recognition appeared already in 70's of the last century. It was not high domain natural language processing, just simple keywords recognition, e.g. single spoken digits. Wider research around speech recognition techniques started in the last two decades. The main stream was supported by military. Typical applications of speech recognition have been voice communication systems, voice-controlled systems, language translation and another voice or language identification systems. The governments are also supporting research in recognition systems helping citizens and more often, we can see automatically transcribed subtitles for TV programs or channels. There are other applications in medical and other areas.

Not only governments are supporting research around speech recognition. Research institutes and companies begin to be interested into this field — especially when multimedia data and information are becoming more relevant. They are more relevant with cheaper computation power and wide-spread voice communication technologies. Recently, huge amount of multimedia data has become available, it is still growing, and it is quite difficult to search through the amount of data. Thus speech recognition is becoming more and more important, as there is need to index such multimedia data.

As the Internet spreads all over the world, a lot of textual information, which could be related to acoustic data, is available. The point is that this textual data could be used to give advices to speech recognition systems. The input of speech recognizer is the information spoken by human. And textual information freely available over the Internet is produced by human as well. So why not use this association?

Chapter 2

Speech Recognition Techniques

For new readers in this field, speech recognition is process in which the acoustic signals (or speech waveforms) are converted into sequence of lexical units. Lexical units can be separate words, phonemes, or even whole sentences, or continuous text pieces. It depends on type of the recognition system. Also, considering start and end of speaker utterance distinguishes different recognition systems. According to [2], speech recognition systems can be separated into the following classes:

- **Isolated Words** - such system usually expects to have enough long silence between each utterance. It does not accept only single words, but does require a single utterance at a time. This class could be also called isolated utterance class.
- **Connected Words** - connected word system allows separate utterances with a minimal pause between them. „The user is allowed to speak in multiple word phrases, but he or she must still be careful to articulate each word and not slur the end of one word into the beginning of the next word.“ [3]
- **Continuous Speech** - this is one of the most difficult. The system must utilize special methods to determine utterance (and lexical units) boundaries. System allows recognized speaker to speak almost naturally. The form of utterance should be more conveniently paraphrased „computer dictation“.
- **Spontaneous Speech** - in comparison with continuous speech, spontaneous one is more benevolent in a flow of speech. Spontaneous speech contains many words which are acoustically tied together or pitch variations. Actually, there is no exact definition of spontaneous speech. Nevertheless, a system processing natural utterance has to be able to manage all speech variations.

For over a decade, continuous speech recognition has been a focal area of research. These systems are preferably based on statistical methods. Therefore, „speech recognition“ in the following text should read „continuous speech recognition based on statistical methods“, or more exactly, „Large Vocabulary Continuous Speech Recognition (LVCSR)“. [6]

Chapter 3

Large Vocabulary Speech Recognition

At first, we need to know, how large vocabulary continuous speech recognition system is built, to understand the idea connecting data mining with speech recognition. As assumed, text is describing speech recognition based on statistical methods.

The whole process of speech recognition is shown in figure 3.1. The recognition process is marked as Decoding in the bottom part of the figure. The Training process is covered as well, which is on the top.

The decoding process can be basically split into two main parts - Acoustic Front-end and Decoder. Acoustic Front-end takes acoustic speech signal Y on input, pre-processes the signal and transforms it into acoustic vectors (or features) X characterizing the input speech. The transformation can be also called feature extraction. It is a process which is trying to suppress irrelevant information from the input signal and keep only the features representing the speech. The frequently used algorithms for feature extraction are Mel Frequency Cepstral (MFC) analysis and Perceptual Linear Prediction (PLP) [6].

The front-end can also filter the signal and relieve the input speech of the background noise, room reverberation or microphone characteristics. This can be also done by acoustic modeling in the training process, thus the ambient noise is considered to be a part of the input speech. It depends how the recognition system is built [10].

The Decoder performs conversion of the acoustic vectors to Speech Transcription. The conversion is based on statistical models created from real textual and speech data. The models are different for each transcribed language and can be separated into Language Model, Acoustic Model and Recognizer Lexicon.

Language model (LM) is probability distribution over some texts well characterizing the transcribed speech. If the recognizer is more universal, then the texts are collected from more different sources and vice versa. The probability distribution is usually evaluated on separated words and word sequences (the word sequence is denoted with W and its probability is denoted with $P(W)$ in the figure), as described in chapter 3.2 about language modeling. Before dealing with language modeling in detail, formulation and basics of probability theory are described from speech recognition point of view in chapter 3.1.

The acoustic model describes the correspondence of acoustic vectors sequence X with word transcription H. Thus the acoustic model evaluates conditional probabilities $P(X|H)$, which says how likely the acoustic front-end produced X, when the speaker uttered word sequence characterized by H.

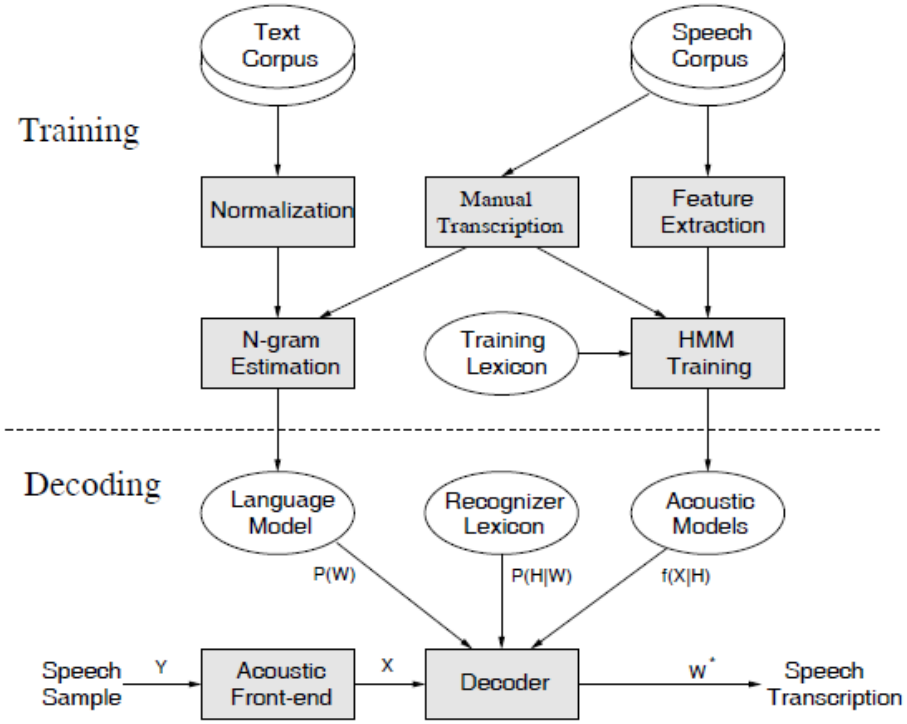


Figure 3.1: System diagram of a generic speech recognizer based on statistical models, including training and decoding processes and the main knowledge sources. [6]

The variation of all the languages is typically very large, thus the acoustic form is not described by acoustic models for each word. The words are rather divided into smaller acoustic units, usually phones or phonemes. This is modeled by the Recognizer Lexicon, which describes word W with phones sequence H. Lexicons and more about pronunciation is discussed in chapter 3.4.

The lexicon is sometimes very closely connected with the acoustic model, thus the acoustic model is evaluating directly probability distribution $p(X|W)$, and the pronunciation variants are looked up in the recognizer lexicon. But there are usually not so many pronunciation variants. „The average number of pronunciation variants per word is less than two.“ [6]

Large vocabulary speech recognition is a statistical method based on statistical models. Thus the language model and acoustic models are trained on some real data. In other words, the models could be considered an approximation of probability distribution of modeled language based on some collected data. Obviously, the precision of the distribution depends on a scope of the data used to train the models.

The training process represents significant part of the recognition system development. The process of models estimation usually takes a lot of time and resources, because the amount of data used to create the models are typically very large. The collection of training data is called corpus. Acoustic model is estimated from speech corpora which consist of speech records associated with manually transcribed texts. The acoustic data of the speech corpus are, at first, pre-processed by Feature Extraction, similarly as in the acoustic front-end before the decoding. As the most common technique to represent the acoustic model, the HMM (Hidden Markovov Model) is used. Acoustic modeling is briefly described in

chapter 3.3.

A base for LM estimation is normalized Text Corpus. „One motivation for the normalization is to reduce lexical variability so as to increase the coverage for a fixed size task vocabulary. The processing decisions are generally language-specific.“ [6] The normalization can consist of numbers or dates to text conversion, treatment of punctuation markers (hyphen, apostrophe), extending abbreviations or capitalization. The texts acquired from the manual transcriptions of speech can be also used to estimate the language model (N-gram Estimation) as marked in the figure with arrow from Manual Transcription. N-grams are covered by chapter 3.2.2. [6]

3.1 Basic Formulation

To understand and discuss the theory of statistical language modeling, we need to define the basic mathematical framework. The following notation will be common for the whole text.

Let define X as the acoustic data - imagine it as the input acoustic signal we want to recognize and transcribe to sequence of words. The input acoustic signal X can be split into a sequence of symbols from alphabet $\mathfrak{X}$. Considering a variety of input speech, $\mathfrak{X}$ could be quite large. Nevertheless for language modeling this fact is not important.

$$X = x_1, x_2, ..., x_m \qquad x_i \in \mathfrak{X} \tag{3.1}$$

Symbol W stands for sequence of words, which is one of all the possible transcribed outputs of the recognition. Each word belongs to finite vocabulary $\mathfrak{W}$. The vocabulary is often constructed for some specific language or some specific application. So it usually contains tens or hundreds thousands of words, but the size of the vocabulary also depends on application.

$$W = w_1, w_2, ..., w_n \qquad w_i \in \mathfrak{W} \tag{3.2}$$

Now, the whole work of speech recognizer could be written as:

$$\hat{W} = arg\ max_W\ P(W|X), \tag{3.3}$$

where $P(W|X)$ denotes the probability that word sequence W was spoken, while input of recognizer was acoustic evidence (data) X. The equation expresses, that the recognizer is searching for the word sequence W, most likely fitting the acoustic evidence X.

Equation 3.3 can be modified by Bayes formula to [10]:

$$P(W|X) = \frac{P(W)P(X|W)}{P(X)}, \tag{3.4}$$

where $P(W)$ is a-priori probability of the word string W (probability that the string W will occur in a speech). $P(X|W)$ is the conditional likelihood of acoustic evidence X, when the word string W was spoken. In other words, it is a likelihood that the acoustic signal X conforms to word string W. And $P(X)$ is the evidence, it is a sum of probabilities $P(W, X)$ over all possible words W:

$$P(X) = \sum_W P(W, X) \tag{3.5}$$

Since the recognizer is (according to equation 3.3) maximizing $P(W|X)$ over word sequence W, $P(X)$ has no effect and can be eliminated — in Bayes formula, it plays a role of normalization constant:

$$\hat{W} = arg\ max_W\ P(W)P(X|W) \tag{3.6}$$

3.2 Language Modeling

To evaluate equation 3.6 we need to know the a-priori probability $P(W)$ of all possible word sequences W or at least those, which speaker is willing to utter, or he wants to transcribe. And that is exactly the purpose of language modeling, to estimate the a-priori probabilities of input word sequences. The a-priori probability $P(W)$ can be decomposed to

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1}), \tag{3.7}$$

where $P(w_i|w_1, ..., w_{i-1})$ is probability that word w_i will follow word sequence $w_1, ..., w_{i-1}$. The sequence $w_1, ..., w_{i-1}$ is usually called history.

Thinking about to calculate probability $P(w_i|w_1, ..., w_{i-1})$ in reasonable time is quite absurd, for instance, for vocabulary size $|\mathfrak{W}| = 10,000$ and $i = 3$ the number of evaluated parameters is $10,000^3 = 10^{12}$. It is also unreasonable to estimate w_i depending on entire history of all the previous utterances. More convenient is put the history into equivalence classes $\Phi(w_1, ..., w_{i-1})$. Then equation 3.7 can be modified to

$$P(W) = \prod_{i=1}^{n} P(w_i|\Phi(w_1, ..., w_{i-1})). \tag{3.8}$$

Therefore calculation of $P(W)$ is easier, it requires only determine classification Φ, and then calculate probabilities $P(w_i|\Phi(w_1, ..., w_{i-1}))$. Evaluating $\Phi(w_1, ..., w_{i-1})$ is usually called equivalence classification described in chapter 3.2.1.

It is also useful to use application specific language model, e.g. medical dictation system can use different language model from an automatic subtitle transcriber for weather forecast. A recognition system with properly constructed language model and vocabulary is able to achieve satisfactory result in recognition and speed as well. [10]

Language models can be constructed just from written materials or even from manually transcribed speech as is indicated in figure 3.1. The second part of the text is focused to text acquisition from the Web and the text discusses possibility to use acquired data as written materials to construct language models or to improve existing models.

3.2.1 Equivalence Classification of History

The first way of classification which would comes out, when thinking about normal language and its grammatical roles, is to split vocabulary into classes denoting different grammatical functions of words. Such classes could be 'noun', 'verb', 'preposition', 'adverb', etc. Grammatical classes might be also split and refined according to semantic interpretation. Example of semantic labels is 'name of country', 'name of month', 'action', 'free time activity', etc. However, performing classification this way is quite complicated, and may contain a lot of exceptions. Moreover there is no definition of such classes. We would need to

specify them and then manually classify the whole vocabulary. A lot of words could also fall within more than one class.

As noted in the previous chapter, evaluating probabilities $P(w_i|w_1, ..., w_{i-1})$ is not possible in reasonable time even for small vocabulary. Actually, it is also not necessary. Thinking a while about usual way of sentence or speech construction, we can find a lot of examples of history sequences $w_1, ..., w_{i-1}$ possibly falling into a same class. Indeed classifying history into the equivalence classes will not cause any harm to the language model, if the classification is properly chosen. Furthermore, a vast amount of word sequences will never occur in any modeled language.

Take a look once more on approximation of history by function $\Phi(w_1, ..., w_{i-1})$. This function denotes mapping of history into some specified set of equivalence classes. Thinking about Φ in context of history processing as a finite state „grammar“, history can be decomposed to states Φ_i. The grammar is in state Φ_i at time i, in state Φ_{i-1} at time $i-1$, and so on. Word w_i then forces state Φ_{i-1} change to state Φ_i. Equation 3.8 can be then modified to

$$P(W) = \prod_{i=1}^{n} P(w_i|\Phi_{i-1}). \tag{3.9}$$

Given some corpus of text to build language model probabilities $P(w_i|\Phi_{i-1})$ might be estimated as follows. The input text sequence is run through a finite state grammar and accumulating counts $C(w, \Phi)$. Counts $C(w, \Phi)$ follows, that word w came when grammar was in state Φ. Estimation of desired probability could be

$$P(w_i|\Phi) = \frac{C(w_i, \Phi)}{C(\Phi)}, \tag{3.10}$$

where $C(\Phi)$ is how many times the grammar was in state Φ.

Not depending on classification method, the estimation must provide sufficient information about history to be able to well predict the consequent word. [10]

3.2.2 N-gram Language Model

One and very simple approach how to classify history is to use N-gram language model. The N-gram language model takes as the parameters a certain number of last words from the history. For instance, bigram model takes only one last word, trigram last two, etc. Estimating the a-priori probability for trigram model is following:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1}) \tag{3.11}$$

In present time, the trigram model is the most frequently used, and moreover the model is surprisingly powerful. Longer N-gram language models are sometimes used: 4-gram or even 5-gram. They usually give small improvement in recognition, but also the complexity of the whole system is increased.

Empirically, it was shown that context longer than 6-grams does not add any improvement into the language model. [11]

3.2.3 Smoothing

Estimation of probability for trigram language model (equation 3.10) is given by:

$$P(w_3|w_1,w_2) \doteq f(w_3|w_1,w_2) = \frac{C(w_1,w_2,w_3)}{C(w_1,w_2)}, \tag{3.12}$$

where $f()$ is the relative frequency function which is expressed with counts $C()$. The counts $C()$ stands for a number of given sequences w_1, w_2, w_3, or w_1, w_2 respectively, which appeared in the training data. Relative frequency is just approximation of the real probabilities as marked in the equation.

Language models are trained on limited amount of textual data (often called text corpus), and although the corpus can be very large, not all the trigrams w_1, w_2, w_3 appear there. A lot of trigrams thus would be assigned probability $P(w_3|w_1,w_2) = 0$ and multiplication in 3.11 evokes that the a-priori probability of the whole word sequence W will be $P(W) = 0$.

To avoid evaluating $P(W)$ to zero, smoothing is typically used. Example of smooth algorithm is simple interpolation, which combines trigram, bigram and unigram relative frequencies,

$$P(w_3|w_1,w_2) = \lambda_1 f(w_3|w_1,w_2) + \lambda_2 f(w_3|w_2) + \lambda_3 f(w_3), \tag{3.13}$$

where λ_1, λ_2 and λ_3 are weights giving importance to each part of model. The weights must satisfy $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

There are a lot of smoothing methods, namely Jelinek-Mercer, Witten-Bell, Good Turing, absolute discounting, Knesser-Ney and others.

The smoothing algorithms are usually performing interpolation of n-grams with lower-order distributions (e.g. trigram with bigram and unigram likewise in equation 3.13). Each smoothing algorithm is only specifying different relative frequency function $f()$ and weights λ_1, λ_2 and λ_3. When performing interpolation, the functions of lower-order distributions are added although the relative frequency of evaluated N-gram is nonzero. There is also another approach called backing-off. Compared to interpolation, backing-off is evaluating the lower-order distribution only when N-gram relative frequency is evaluated to zero. This could be written as:

$$P(w_n|w_1,...,w_{n-1}) = \begin{cases} f(w_n|w_1,w_2,...,w_{n-1}) & when\ \ C(w_1,...,w_{n-1}) > 0 \\ \lambda f(w_n|w_2,...,w_{n-1}) & when\ \ C(w_1,...,w_{n-1}) = 0. \end{cases} \tag{3.14}$$

λ is backing-off probability, which gives lower weight to lower-order distributions.

In trigram example model when $f(w_3|w_1,w_2)$ is evaluated to zero, bigram relative function $f(w_3|w_2)$ is computed. And only when even $f(w_3|w_2)$ is zero, then unigram frequency $f(w_3)$ is used. [5], [8]

3.2.4 LM Training

Three data sets play role in the process of language model creation; training set, development set and testing set. Usually the sets are prepared by splitting the text corpus. The largest part of data is the training set used for to estimate language model. The development data are used to tune parameters of the estimation, e.g. back-off probabilities,

interpolation weights. The quality of the resulting language model is evaluated on the testing set. The testing data must not be a subset of the training data.

A typical language model is estimated on a large text corpus. The text corpus normally consists of texts from various sources even for application specific field. Some data from the text corpus better fit the development data, some worse. Thus the estimation of the language model should consider variety of the corpus content and set the probabilities according to relevance of each source. This is done by evaluating the perplexity on the development data. Evaluating perplexity during the process of language model estimation is quite difficult task. There is another approach called model interpolation. A language model is estimated separately for each source and the models are then interpolated together. The key is how the interpolation weights are set. Development data can be again used for this purpose. [6]

3.3 Acoustic Modeling

Acoustic modeling is not a subject of this work, but refers to pronunciation issue, so we mention it just briefly. As noted before (equation 3.6), the work of recognizer is maximizing the product $P(X|W)P(W)$ over all possible word transcriptions W given acoustic data X. The probabilities $P(W)$ are evaluated by language model as described in chapter 3.2, but to solve the recognition equation the recognizer also need to evaluate likelihood $P(X|W)$. This likelihood is given by acoustic model. A good practice is to model the acoustic with hidden Markov model (HMM). You can find a detail description of HMM for example in [10].

3.4 Phonetic Models

Let us describe acoustic of word w as a sequence of phones $\Psi(w) = \varphi_1, \varphi_2, ..., \varphi_{l_v}$, where φ_i is from predefined phonetic alphabet. The sequence $\Psi(w)$ is also called encoding of the pronunciation of word w. At first, the phonetic alphabet needs to be specified. It usually involves phonetic rules of modeled language, because this step is language dependent. Some nations have such a difficult phonetic system, that others are not able to pronounce their words at all.

To describe phonetic system a universal phonetic alphabet called International Phonetic Alphabet (IPA) developed by the International Phonetic Association. It is mostly based on Latin alphabet and it standardizes symbolic representation of possible sounds of spoken languages. IPA consists of more than 100 symbols (consonants, vowels, diacritic). A phonetic alphabet of recognition system can be based on IPA, whereas systems for specific language normally define their compact versions of the alphabet. „For example, some commonly used phone set sizes are 45 for English, 50 for German and Italian, 35 for French and Mandarin (to which tones may be added), and 25 for Spanish.“ [6] Phone symbols set for English is shown in figure 3.2.

When the phonetic alphabet is specified, the pronunciation dictionary can be created. The pronunciation dictionary defines the encodings $\Psi(w)$ for all possible words w from recognizers vocabulary. The pronunciation of the words can be determined automatically from phonetic and grammatical rules of modeled language, or it can be determined from the training process. But sometimes the pronunciation needs to be manually corrected.

Moreover many words have more different pronunciations variants. This is also specified by the pronunciation dictionary. Each variant can be additionally assigned different probability. But less complex recognition systems do not take other variants in account and model only the basic phonetic form.

Phone	*Example*	*Phone*	*Example*
Vowels		Fricatives	
i	beet	s	sue
ɪ	bit	z	zoo
e	bait	ʃ	shoe
ɛ	bEt	ʒ	measure
æ	bat	f	fan
ʌ	but	v	van
ɑ	bott	θ	thin
o	boat	Plosives	
u	boot	b	bet
U	book	d	debt
ɜ˞	bird	g	get
Diphthongs		p	pet
ɑʲ	bite	t	tat
ɔʲ	boy	k	cat
ɑʷ	bout	Affricates	
Reduced Vowels		ʧ	cheap
ə	xbout	ʤ	jeep
ɨ	dated	Nasals	
ɔ˞	butter	m	met
Semivowels		n	net
l	led	ŋ	thing
r	red	Syllabics	
w	wed	m̩	bott om
y	yet	n̩	button
h	hat	l̩	bottle

Figure 3.2: Set of 45 phone symbols for English with illustrative words, with the portion corresponding to the phone sound underlined. [6]

3.5 Quality of Language Model

The process of the language models creation does not only include basic evaluation of N-gram counts and probabilities. It also covers, or is preceded, by data selection, creation or specification of recognizer dictionary and other pre-processing procedures. A recognition process is difficult task and takes a lot of resources. Thus the language model should be created with respect to complexity of its application.

Usually, contradictory requirements come in mind: the quality and resulting speed of the recognition system. When the user requires a good system recognizing all potential words in the input speech, a large dictionary is usually created. Dictionaries of several hundred thousand words are not an exception. Some complex languages containing difficult grammatical rules, such as Czech, can require dictionaries with more than a million words. But the search space of recognizer, thus speed of the recognition system, also depends on

the size of dictionary.

Question is then what is more important for the user: speed or quality of the recognition result? The answer also depends on the application, but mostly we are trying to do some compromise between these two. In context of language models, there are two important properties rating quality of recognition systems – „Out of Vocabulary (OOV) error rate“ and „perplexity“. Out of Vocabulary error rate denotes how many words in the recognized speech (test data) were not found in the dictionary or in the language model.

Perplexity is also evaluated on test data (eventually, development data during the language model parameters estimation). Perplexity could be explained as a normalized probability how well the testing data matches given language model. Imagine language model as a grammar, which expresses how word sequences are built. Then grammar which better matches given testing data is assigned a higher probability, thus better perplexity [11].

To evaluate perplexity, at first, we need to know something about information theory and entropy.

3.5.1 Information Theory

The roots of information theory were developed by Claude Elwood Shannon. Later, it found application in many fields as mathematics, statistics, physics, electrical engineering or neurology, but mainly computer science. The theory is widely used in data analysis and processing, e.g. compression, cryptography, molecular codes and also statistical and natural language processing.

3.5.2 Entropy

A basic term of information theory is entropy. Entropy is also important property in a lot of other fields than information theory, e.g. thermodynamic entropy is applied in statistical mechanic and other physical or chemical theories. In information theory, entropy is the key term for compression and data or signal processing.

Entropy is based on Shannon's information theory and from that point of view, entropy is a measure of uncertainty or disorder of some random variable. Accordingly, the entropy H for discrete random variable X, defined on a set $x_1, x_2, ..., x_n$, is written as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i), \tag{3.15}$$

where $p()$ is a function defining probability of all possible discrete values of variable X (probability distribution of X). It is similar to probability density function of continuous variable. The base b can be specified according to the purpose of use, usually 2, Euler number e or 10. The most common example explaining entropy is measuring information content of some pre-defined predictable input data. The base b of logarithm in equation 3.15 is substituted with 2, then the entropy H gives an average number of bits needed to encode one symbol from the range given by the variable X. [7]

As defined above, entropy describes information measure of independent variables. But in context of language models we need to evaluate sequence of random variables w_i (sequence of words). We can think of the sequence as a random variable ranging over all sequences of words of length n, which belong to the modeled language L. Thus the entropy of such variable can be written as:

$$H(w_1, w_2, ..., w_n) = - \sum_{(w_1,w_2,...,w_n)\in L} p(w_1, w_2, ..., w_n) \log p(w_1, w_2, ..., w_n), \tag{3.16}$$

and the function normalized to entropy per word as:

$$\frac{1}{n} H(w_1, w_2, ..., w_n) = -\frac{1}{n} \sum_{(w_1,w_2,...,w_n)\in L} p(w_1, w_2, ..., w_n) \log p(w_1, w_2, ..., w_n), \tag{3.17}$$

where $p(w_1, w_2, ..., w_n)$ is a joint probability function characterizing the language L.

In real languages we need to consider word sequences of theoretically infinite length. From that point of view language is a stochastic process. Entropy of the language is then defined as follows:

$$\begin{aligned} H(L) &= - \lim_{n\to\infty} \frac{1}{n} H(w_1, w_2, ..., w_n) \\ &= - \lim_{n\to\infty} \frac{1}{n} \sum_{(w_1,w_2,...,w_n)\in L} p(w_1, w_2, ..., w_n) \log p(w_1, w_2, ..., w_n). \end{aligned} \tag{3.18}$$

Using the Shannon-McMillan-Breiman theorem, the entropy can be rewritten as:

$$H(L) = \lim_{n\to\infty} -\frac{1}{n} \log p(w_1 w_2 ... w_n). \tag{3.19}$$

Thus the probability is evaluated on enough long word sequence $w_1 w_2 ... w_n$. Enough long sequence means, that it contains a lot of shorter sequences according to their probability characterizing the language L. This assumption can be done when we think of the language as a stationary stochastic process, in which probability distribution over word sequences is time invariant. Thus the shorter sequences can appear everywhere in the long words sequence. [11]

Such a simplification of entropy evaluation is not correct, but is a good approximation. To compute probability of a long word sequence cross-entropy is introduced.

3.5.3 Cross-entropy

Cross-entropy is a tool allowing us to compute entropy of some probability distribution p using simplified model m of the distribution p. The definition of the cross-entropy of model m on distribution p is the following:

$$H(p, m) = \lim_{n\to\infty} -\frac{1}{n} \sum_{(w_1,w_2,...,w_n)\in L} p(w_1, w_2, ..., w_n) \log m(w_1, w_2, ..., w_n). \tag{3.20}$$

Both, probability distribution p and its model m, play role in the equation. Using again the Shannon-McMillan-Breiman theorem, the cross-entropy for stationary process can be rewritten as:

$$H(p, m) = \lim_{n\to\infty} -\frac{1}{n} \log m(w_1 w_2 ... w_n). \tag{3.21}$$

As the result, we got that the cross-entropy of some model m on probability distribution p can be estimated on a single, enough long output sequence of the model m.

Any approximation of the distribution p will not be better, thus the cross-entropy is considered to be an upper bound of the distribution. So for every model m the true entropy of p is lower:

$$H(p) \leq H(p, m). \tag{3.22}$$

The cross-entropy can be also used to measure an accuracy of the model m. If we know the entropy of the distribution $H(p)$, then the model m is more accurate when its cross-entropy $H(p, m)$ is closer to the entropy $H(p)$. [11]

3.5.4 Perplexity

The cross-entropy is used to compute perplexity. But the definition in limit is not convenient for numeric evaluation, thus an approximation with a model P on a word sequence W of length N is defined:

$$H(W) = -\frac{1}{N} \log P(w_1 w_2 ... w_N). \tag{3.23}$$

The perplexity of the model P is then defined on a word sequence W from its cross-entropy (the base of the previous logarithms is presumed to be 2 – we measure in bits):

$$\begin{aligned} PP(W) &= 2^{H(W)} \\ &= P(w_1 w_2 ... w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}} \end{aligned} \tag{3.24}$$

After expanding the joint probability:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 ... w_{i-1})}}. \tag{3.25}$$

The perplexity can be translated as a normalized entropy of some model evaluated on a long word sequence W, so how well the model predicts the sequence W. The higher probability is assigned to the sequence W, the lower entropy, and consequently perplexity the sequence acquires. Thus we can compare quality of the language models on some prepared sequence W – test data. Then a language model which assigns lower perplexity to the testing data leads to better recognition system. [13], [11]

3.5.5 Out of Vocabulary Rate (OOV)

There are two types of vocabulary systems, open and closed one. The closed vocabulary system assumes that the vocabulary contains all possible words, which can appear in testing data. But the testing data can be additionally taken from a new source and unknown words can appear there. Such words are called out of vocabulary (OOV) words. The number of unknown words is usually expressed as the percentage of all words and is called OOV rate or OOV error rate. The open vocabulary system is adding the OOV words into the

probabilistic model. An extra meta-word (typically denoted with `<UNK>` as an unknown word) is added. The unknown word is then taken as a normal word during the N-gram probabilities estimation. [11]

Out of vocabulary rate is also quite important property in recognition, since each non-recognized word causes considerable error. One could create a large vocabulary potentially covering all possible fields of input speech. But as discussed before, to lead the recognition system to better performance, reasonably small vocabulary is required. A good practice is to create a vocabulary, while the OOV error rate is minimized on the development data. One option is to order all words by number of occurrences in the development set and take the first N most frequent words. It is obvious, that the OOV error rate is inversely proportional to the size of the vocabulary. [6]

3.5.6 Measuring the Quality of LM

Perplexity is widely used property to measure a quality of the language models. The language model is basically a probability distribution over some texts or sequences of words. Thus the perplexity of the language models is measured per word to get reasonable values. However, we presume that the dictionaries of compared recognition systems are identical, thus they contain the same number of words.

From equation 3.6 we know, that the recognizer is maximizing product $P(W)P(X|W)$, thus the complexity of the recognition task is given by finding probability $P(W)$, but also $P(X|W)$. The probabilities $P(W)$ are given by the language model and conditional likelihoods $P(X|W)$ by the acoustic model. The previous text discussed, that entropy of the probability distribution indirectly measures the complexity of some model. The probability distribution $P(W)$ produces entropy $H(W)$ as a base for perplexity and measure of language model complexity. For acoustic model it is quite difficult to compare conditional entropies $H(W|X)$ and say which model is better. A really bad acoustic model could result in $H(W|X) \cong H(W)$. Then entropy $H(W)$ thus could be considered an upper bound of both entropies. [10]

To measure real quality of the recognition system, the Word Error Rate (WER) is introduced. Word Error Rate represents a number or percentage of words, which were recognized incorrectly in the test speech. The WER is defined as sum of three error counts:

$$WER = \frac{subs + ins + del}{N}, \tag{3.26}$$

where *subs* stands for number of substitutions (words which were substituted with different one), *ins* is number of insertions (additional words which are not present in the original transcription) and *del* is number of deletions (words which are missing in the final transcription).

But since the recognition is really difficult task, it is impossible to develop a recognition system by making a modification in some part (e.g. language model), and run recognition of the whole test speech again to measure the Word Error Rate. Thus perplexity of language model can be regarded as a good property to compare an improvement of the recognition system, while identical acoustic conditions are presumed. [6], [13]

Chapter 4

Text and Web Mining

The mining is quite new area coming with need to retrieve valuable information or knowledge from quantum of data being created and stored every day. Mining is actually interdisciplinary field, which is borrowing from many other related disciplines, e.g. information science, machine learning, statistics, pattern recognition or other disciplines. Usually, a lot of techniques are combined in one system and it depends on specific application, how they are used or combined. A concept of the mining system also depends on information we are interested in or type of the mining source. The data mining systems can be classified according to the source of data to be mined from, according to the kind of mined knowledge or according to the techniques utilized in the mining process. According to the mining source, the mining systems could be classified into the following categories: [4]

- *Structured* - typically classical data mining using warehousing systems, where data are structured to tables or other regular structures (e.g. relation databases).
- *Unstructured* - the source of data does not have strictly defined structure (good example is the Web, where the content is largely unstructured).
- *Semi-structured* - boundaries between the two described categories are not strict. Usually, information retrieval systems are working with semi-structured data sources.

Whether structured or unstructured, the whole mining process could be separated into several phases as shown in figure 4.1. The concept is more suitable for classical data mining, however similar phases can be found in unstructured mining of the Web. As noted in the introduction, we are connecting data mining with language modeling, because the language models are created from text corpus and the desired text can be possibly acquired from the Web with data mining. Thus the parts of the mining process are described and aligned with the text mining applied in practical part of this work.

The first phase, **selection**, is a separation of relevant data from all the available data. The collection of available data is typically very large and not all the documents contain the desired information. In some cases it is also absurd to perform mining on all the available documents, e.g. the Web mining – if we consider the size of the World Wide Web – according to [1], there was over a trillion unique URLs found on the Web already in 2008.

The pre-processing phase can consist of **data cleaning and integration**. Data mining on structured sources typically cleans up the source documents of inconsistent records breaking the predefined structure, or some noisy data are removed. Integration is performed when data are combined from more sources. The correct records are then transformed, if

needed, into format suitable for the following step, data mining. In application of text mining of unstructured Web, the documents need to be filtered out of redundant content not representing meaningful textual data. The documents on the Web are also in a lot of different formats, thus they need to be converted to some unified form.

All previously mentioned phases could be denoted as a pre-processing phase, because all these steps cover preparation operations on the data, from which we want to mine some information or knowledge. Each book on data mining is separating these phases slightly different way, but all describes similar steps. Moreover the order of pre-processing steps is application specific. Figure 4.1 shows one of possible sequences.

And finally, there comes the main part – **data mining**. Data mining does not define any specific algorithm. It could be described as a process, in which the desired information is gained form prepared data. Typically, there are not such data, which could be just taken and the information would be easily extracted. Thus the pre-processing phases should be discussed in context of data mining or they should be considered as a part of the data mining process.

When the desired information was gained by data mining, it can be used for the purpose it was mined for. That is the last step Interpretation/Evaluation. In application of text mining described in this thesis, the valuable text is extracted from the Web documents and clean text is used for the language model estimation and adjustment.

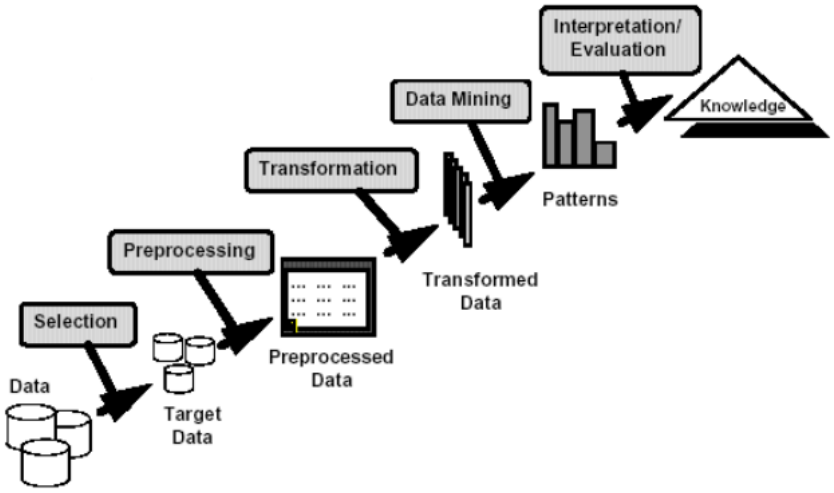


Figure 4.1: Typical data mining process. [14]

4.1 Text Mining

Why speaking about structured and unstructured data sources? Data mining is usually connected with relational databases or other similar sources, where the data are typically structured and the process of mining is quite straightforward. However, text or document databases are also developed. They are storing different articles, papers or even books, e-mail messages or Web pages. Such storages are typically semi-structured or unstructured. We are going to speak about the Web as a source of text. And the World Wide Web can be understood as a large, interconnected, dynamic text database. To search and retrieve

some information from such a huge source of data, a technique called information retrieval is introduced.

4.2 Information Retrieval

Information retrieval (IR) is the crucial technique supporting organization and searching through large amount of textual documents. Examples of IR systems are on-line library catalogs or document management systems, but also the World Wide Web search engines. The aim of such systems is to find documents corresponding to given input query (typically some keyword or document example). Information retrieval is quite complex and difficult problem, it is thus useful to establish a measure of precision and recall for a given algorithm.

$$precision = \frac{relevant \cap retrieved}{retrieved}$$

$$recall = \frac{relevant \cap retrieved}{relevant}$$

The precision equation expresses a rate of how many retrieved documents are relevant to the specified query. Recall denotes the percentage of documents that are relevant to query and were really retrieved. [9]

4.2.1 IR Techniques

The basic concept of IR is that all the documents are described by a set of representative keywords (usually called *index terms*). The index terms are typically automatically extracted from the documents. Usually, as the first, stop words are removed from the documents. Stop words are those words, which are not bringing any information into the document (e.g. prepositions or conjunctions). Now considering typical variation of text, a many different forms of similar word appear in one document. Thus word stems are used to group similar terms.

When we have a set of representative terms, they can be used to compare similarity of two documents, eventually relevancy of the document to given query. The query is typically also a set of keywords. There are more approaches how to measure the similarity. A basic approach is the **boolean model**, which just evaluates the presence of given terms in the document in binary form. The **vector-space model** is more frequently used.

Presuming that each document is represented by t index terms, they can be represented with a vector v in a t-dimensional vector space $\mathbb{R}^t$. And each term in the vector is represented by its frequency or weight. Evaluation of term weights is described in the next chapter. The degree of similarity of two documents or similarity of document to some input query is evaluated as the distance between vectors, representing the documents or the input query. The distance is usually expressed with Euclidean distance or with the dot product of the vectors (a cosine of the angle formed by these vectors).

4.2.2 Term Weighting

A basic property to evaluate the term relevance to the given document is term frequency. It can be expressed binary, thus „1“ when the term is present in the document and „0“ otherwise. Or it can be represented with the relative term frequency, that is, the number of occurrence of given term divided by the total number of all the terms in the document.

An often used measure is the normalized term frequency (TF), which is also used in the Cornell SMART system [9]. The frequency $TF(d,t)$ of term t in document d is computed as follows:

$$TF(d,t) = \begin{cases} 0 & if freq(d,t) = 0 \\ 1 + \log 1 + \log freq(d,t) & otherwise, \end{cases} \tag{4.1}$$

where $freq(d,t)$ denotes number of occurrences of term t in document d.

Another important property is the inverse document frequency (IDF), which gives less importance to the terms, which are covered by all the observed documents. Such terms are not suitable to evaluate the relevance, since they do not distinguish the documents. The IDF of term t used in Cornell SMART system is computed as follows [9]:

$$IDF(t) = \log \frac{1 + |d|}{|d_t|}, \tag{4.2}$$

where $|d|$ is the number of all observed documents and $|d_t|$ is the number of documents containing term t.

The TF and IDF frequency coefficients are typically combined together into the $TF - IDF$ weight in a complete vector-space model:

$$TF - IDF(d,t) = TF(d,t) \times IDF(t) \tag{4.3}$$

4.2.3 Text Indexing

Since text document processing and extraction of terms are quite difficult tasks, there is need to pre-process all the documents, when we want to search in a large amount of documents. The document pre-processing is called indexing. During indexing, the index terms are extracted from all the documents and stored in association tables. The most common used indexing tools are inverted index and signature file.

- **Inverted index** stores the associations in two hash or tree index tables, the document table and the term table. The *document table* associate each document with list of terms related to given document and the *term table*, vice versa, stores association of terms with list of documents in which the term appears.
- A **signature file** creates for each document signatures in form of binary mask, where each bit represent presence of one index term or set of related terms.

4.2.4 Other Techniques

There are a lot of other techniques improving quality of text mining and retrieval systems. Some systems are dealing with **synonyms** — words with similar meaning may be written in completely different form or the same terms may have different meaning in different contexts. Another approach is to assign documents relevance based on users feedback, which is updated while the system is used.

For really huge sources, the text **dimensionality reduction** is performed. The reduction is supported with advanced techniques, like *semantic analysis* of text. Another method adjusting search result is *association analysis*, in which the indexed documents are analyzed to find keywords frequently occurring along with the searched term. The resulting set of documents can be additionally refined by looking up the associated keywords.

The search result may be also enriched with documents related to some search result. Relations between documents are exploited with document classification and clustering.

4.3 Mining the Web

As noted before, the World Wide Web may be considered a large, distributed database (or source) of data. It can be source of textural data, but also different kinds of multimedia data or images. Since the Web is so huge and widely available, it is big challenge to use it as a source for data mining techniques. Before starting to develop any method processing the Web content, the following few facts need to be considered:

- The Web is highly dynamic source. A lot of Web pages are periodically updated and new pages are every day created. News, weather, Internet shops advertisements, and forums – all of these are daily or even more often updated.
- The content of the Web documents possesses a large variety, since the content can be possibly maintained by everyone. Thus there is no guarantee that the content keeps any structure.
- A relevancy or usefulness of the Web content is also disputable. „It is said that 99% of the Web information is useless to 99% of Web users.“ [9] And typical user is interested into a really small portion of the Web content.

Regarding mentioned facts, extracting relevant texts from the Web is then more difficult.

4.3.1 The Web Mining Methods

The diversity of the Web brings need of other methods analyzing the content of the Web documents. In compare with text databases, where the content is usually unstructured, Web content can be considered semi-structured, at least the layout of the documents, since a majority of the Web pages are in HTML or XHTML document format. The structure of such documents follows Document Object Model (DOM). The DOM defines the structure of Web documents as a tree, in which each node correspond to one tag, e.g. `<BODY>`, `<TABLE>`, `<P>`, `<H1>` etc. The DOM tree can be, with advance, used to analyze the content of Web pages, but only when the structure follows some traditional concepts and the tree is defined correctly. There are many documents on the Web which are not even written correctly and the DOM tree cannot be constructed.

Actually, the tree structure of some document does not directly express semantic or visual relation of object displayed on the corresponding page. The more advanced structure analysis is complicated by other widely used techniques, such as Cascading Style Sheets (CSS), which can completely reorganize the visual structure of the document. The visual structure of documents also carry information about object relations, hence technique called VIsion-based Page Segmentation (VIPS) was developed. The VIPS extracts semantic structure from given Web page and represents it with tree, but the tree my completely differ from the DOM tree. [9]

How is the structure of a Web page important for text mining? By analyzing natural relations of objects created on some Web page, the useful content can be identified easily. For instance, in the practical part of this work, the DOM structure of a page is used to find uninteresting content as information bars, menus or advertisements to remove them.

4.4 The Web Search Engines

As the World Wide Web spread all over the world, there was need of search tools to find any useful information. Current searching engines are huge distributed system of procedures, performing regular analysis of the Web content, classifying the content and updating the index databases, while the user interface is providing answers to quantum of search queries per second entered by the clients.

Behind the searcher interface, there is a huge database of index tables providing fast answer to all concurrent queries. Besides that, the index database is created on the fly. The whole examined Web content is analyzed page per page. Thus each Web page is parsed, its structure is analyzed and valuable content is extracted. Then information retrieval helps to extract significant keywords and inverted index or signature file is created. Eventually, some additional procedures are performed, e.g. clustering, association analysis or any form of semantic indexing.

4.4.1 Linkage Structure of the Web

An interesting property of the Web structure is the use of hyperlinks. When author of some Web page places a hyperlink to another Web page, it is probably because the linked page is considered to be interesting destination in mind of the author. Thus authors of the Web documents can support each other by placing a links to other Web documents. The structure of links can be then used to extract useful information about relevancy of Web documents. When there is more links to some documents, the document can be considered more relevant than the others.

One algorithm using hyperlinks is The Hyperlink-Induced Topic Search (HITS). The HITS is evaluating relevancy of Web pages by estimating the authority weight. For this purpose, a hub was created. Hub is a set of documents pointing to authorities. Thus the document can be a list of links pointing to the home pages or commercial sites providing focused topic. The relevancy of authorities and hubs is defined mutually — a good authority is such to which many hubs is referring and a good hub is page pointing to many good authorities. The HITS algorithm iteratively estimates the hub weights and authority weights, and these are then used to choose the most relevant pages to given topic. [9]

4.4.2 Google Search and PageRank

One of the most famous and widely used search engines is Google search. Behind the scenes of Google, a lot of typical techniques discussed above are applied, but in role of linkage ranking, Google's original algorithm PageRank is used.

The concept is quite simple. The relevance (importance) of some document is measured by number of inbound links pointing from other documents. The concept is sometimes called link popularity. The algorithm was described by Lawrence Page and Sergey Brin and the basic formula is [15]:

$$PR(A) = (1-d) + d\left(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)}\right), \tag{4.4}$$

where $PR(A)$ is the PageRank of document A, $PR(T_i)$ is the PageRank of document T_i, which links to document A, $C(T_i)$ is the number of outbound links in document T_i and d is a damping factor from interval $< 0, 1 >$.

You can see from the equation that the result is not just a sum of number of inbound links. Each inbound link is weighted by the PageRank of linked document, so more important documents add more importance to evaluated document. Then there is not possible to improve the PageRank by adding multiple links to one document. Multiple links on the contrary lower the PageRank (division by $C(T_i)$).

While the PageRank of one document is dependent on other documents PageRank, it is not possible to evaluate the result in one step. The PageRank of all the documents is evaluated iteratively. Initially the rank is set to 1 for all the documents and usually 100 iterations is enough to evaluate the PageRank for all the documents.

Benefit of this ranking concept is that every added link to some document raises its PageRank. [15]

Chapter 5

Text Mining Practically

A part of the project was to design and implement a system for acquiring text data from the Web. To be able to construct a good language model, we need to have clear textual information without any punctuation marks, special characters or any graphical decorations. There is a plenty of written information all over the net. The problem is that a vast majority of the Web content is undesirable and could produce inferior model. The designed mining system is principally solving three problems:

- Examine what information could improve the language model of some recognition system. In other words, we need to know which documents of that huge amount available all over the World Wide Web are relevant to add to our text corpus. It is impossible to retrieve the whole Web content and determine which documents are relevant. From the previous chapter we know that the Web documents can be characterized by a set of keywords. In the first phase, keywords well characterizing relevant documents are composed.

- Then the documents corresponding to prepared keywords need to be located and retrieved from the Web. For that purpose a Web search engine can be used. That is the easiest way to determine relevant documents from such large source of data as the Web is. The search engines are performing a large piece of work for this application. The question was: what search engine to choose from those freely available? In this application, the Google search engine was chosen. It is the most widely used engine and the algorithms behind the search engine are proven by long term utilizing and billions users.

- The last, but not less important, problem is how to extract useful text from retrieved Web documents? Typical Web documents contain not only the desired textual information. The documents are full of different banners, advertising content, menu items and other supplementary elements. Such content need to be removed and gained text needs to be post-processed or filtered to give it some structure.

The system is decomposed into the three main blocks or subsystems. All the subsystems are consequently dependent, as the data is processed and floats through all the units of the system. The general scheme of the system is drawn in figure 5.1. The data flow is marked in the figure as well. The following text describes each block separately. A major part of the system is implemented as bash scripts and text filtering components in Perl or AWK. You can also notice that system components are hierarchically organized into separated source

files and the names are marked with small font. An extension of the source file indicates the programming language.

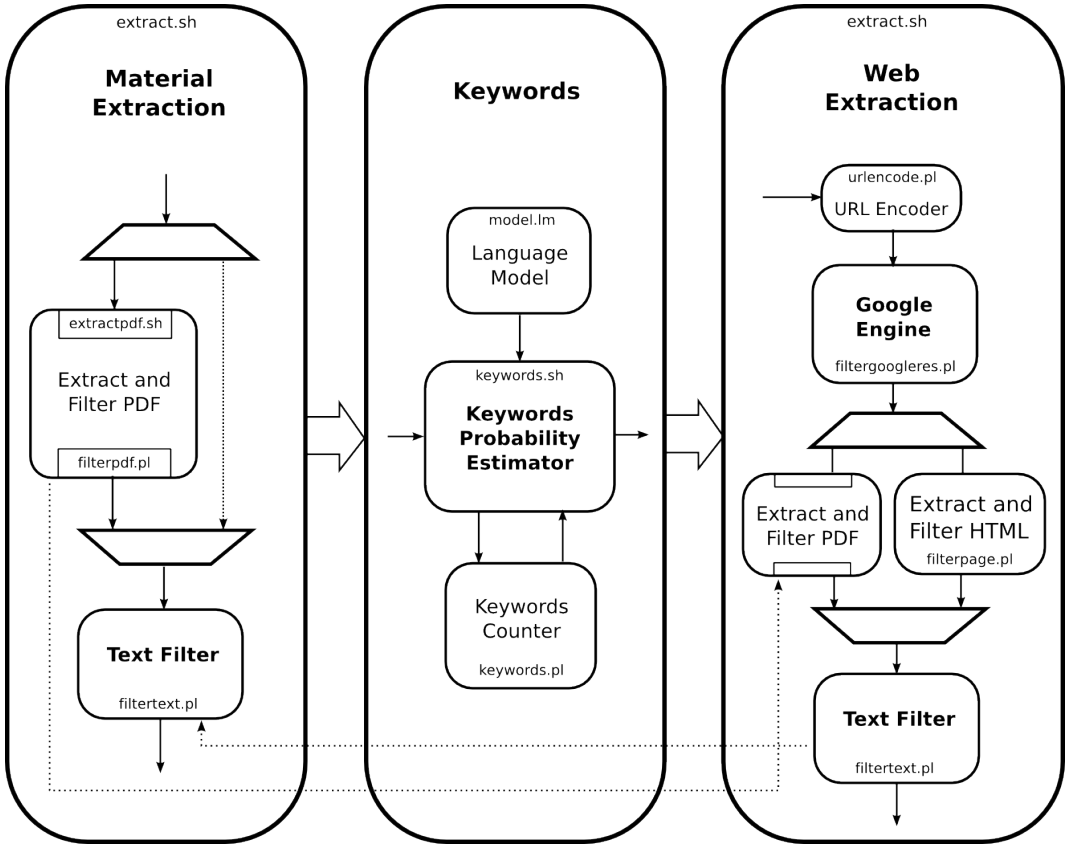


Figure 5.1: A scheme of the mining system for text acquisition.

5.1 Material Extraction

The first block of system (in figure 5.1 named as Material Extraction) performs preliminary modifications on input. The system is designed to be able to accept different form of input data. To understand the concept, we should say, what the typical input is. Purpose of this project is to improve a quality of some language model or create one. From chapter 3.2 we should assume how the language models are created (at least briefly). So what data to use?

5.1.1 Input of The System

To improve existing language model, the input of the system can be basically transcriptions from recognition system, which is using the same language model. Mining the Web should extend the text corpus, from which the language model was created and possibly add non-existing words and word combinations into the corpus, and thus improve the language model. When creating new language model, there are no transcriptions, hence the system also accept different sources of input textual data.

On Faculty of Information Technology (FIT) of Brno University of Technology (BUT), an extensive research on speech recognition systems is performed. We are also experimenting on video records of lectures given at the faculty. The point is, that language models could be created from materials well fitting lecturers speech, e.g. from materials that go along with the lectures. A majority of the available materials are in PDF format, hence the material extraction system also accepts a PDF documents.

5.1.2 Implementation Details

Material extraction consists of three main components. The core is classifying the input data according to its type. The second component performs appropriate extraction or conversion of input. Currently, extraction from PDF documents and plain text documents is available; of course, support for other types of document may be added. The last component filters the extracted text to proper form suitable for the following subsystem, Keywords Extraction.

PDF documents conversion is implemented by Portable Document Format (PDF) to text converter (currently version 3.00), see `pdftotext` manual page[1]. Pay attention to the fact, that the PDF is quite complex format and extraction of the text is not every time so simple. Some of the character encodings are stored in PDF documents in nonstandard way and extraction of the text is almost impossible.

Not only the text extracted from PDF documents may have different character encoding, but also the plain text input is accepted in different encoding formats. The detection and conversion of input text encoding are accomplished by `enca` and `enconv` utility (see `enca/enconv` manual page[2]).

There is nothing more interesting about this part of extraction. I would only emphasize the following bash code,

```
1 script=`which $0`
2 rootdir=`dirname $script`
```

which is used in many bash scripts. The code determines a path of called script. That is because the system consists of many scripts, which are calling each other. The path is prepended to each call of external script and it ensures that the running command can be called from different working directory, than only the one, where the command script is stored.

5.2 Keywords Extraction and Probability Estimation

The input of the second block (in figure 5.1 named as Keywords) is possibly quite large amount of a textual data. Therefore, before starting to mine the Web, we need to know, what information from all the data is important. And because the computers do not understand semantic, we need to specify it. Information retrieval is choosing as representative terms or keywords those phrases, which appear most often in the text. On contrary, when we want to mine text which is missing in the current corpus and which could improve the language model, we extract terms with lower probability — and while working with language models, the terms are evaluated with the language model probabilities.

[1] Manual page available on-line at http://linux.die.net/man/1/pdftotext
[2] Manual page available on-line at http://linux.die.net/man/1/enca

5.2.1 Key Terms

It is quite difficult to determine which words should be tied together to key terms. The described system is trying to use existing language model to estimate probabilities of different word sequences from input text and then compare the language model probabilities of these sequences. While an N-gram language model is used, the word sequences are composed to have the same lengths as the N-grams stored in the model. N-gram language model, which can estimate at least trigram probabilities, is expected. The input textual data is chopped up to word sequences of one, two and three words and the logarithmic probabilities are estimated by specified language model for each sequence. Then the least probable sequences are taken as the key terms.

5.2.2 Probabilities of Terms

The estimation of probability of terms is not so straightforward. When the language model is properly defined, it stores conditional probabilities which satisfy, e.g. for trigram:

$$\sum_{w_3} P(w_3|w_1, w_2) = 1, \tag{5.1}$$

thus all the trigram probabilities with same history w_1, w_2 sum to one. Similarly, for bigram probabilities and unigram probabilities sum to one over all modeled words, plus the special `<UNK>` in open vocabulary language model. Thus if we want to compare probabilities of different terms, we need to know their real probabilities, not conditional as the language model is giving. Computation of real term probability is described in section 5.2.3 about implementation details.

The N-grams probabilities are evaluated by `ngram` SRILM[3] tool. The tool in debug mode gives a verbose output, which contains information about N-gram probabilities and summary for each input line of text. A template for `ngram` command call is the following:

```
ngram -lm $LANG_MODEL_FILE -order N -ppl $KEYWORDS_FILE -debug 2
```

where `$LANG_MODEL_FILE` stands for language model file and `$KEYWORDS_FILE` stands for input list of keywords or text pieces. Argument `-order` specifies maximal order of evaluated N-grams, thus to evaluate unigram probabilities it is `1`, to evaluate bigram `2` and trigram `3`. An example of `ngram` tool output for word „`MODEL`“ with N-gram order set to `1` is following:

```
MODEL
        p( MODEL | <s> )          = [1gram] 0.000200229 [ -3.69847 ]
        p( </s> | MODEL ...)      = [1gram] 0.0792499 [ -1.101 ]
1 sentences, 1 words, 0 OOVs
0 zeroprobs, logprob= -4.79947 ppl= 251.036 ppl1= 63019.2
```

The output contains N-gram probabilities denoted by `p()` for each keyword. In the first square brackets behind the equal sign, the order of listed N-gram is emphasized. Then the positive number denotes normal N-gram probability and the negative number in square brackets is the logarithmic probability. To explain N-gram notation, the word sequences are fractionated by vertical bars (`|`) and `<s>` (respectively `</s>`) denotes the beginning (respectively end) of the sequence (sentence). The first probability 0.000200229 (-3.69847 logarithmic) is the unigram probability of word „`MODEL`“ in which we are interested.

[3]The SRI Language Modeling Toolkit is a package of utilities for building and applying statistical language models. The toolkit is available at http://www.speech.sri.com/projects/srilm/

When evaluating higher order N-grams for longer word sequences, the output of `ngram` tool is similar to the previous one. And the probabilities are evaluated for each input line word by word, as shown in the following example for sentence „`JAZYKOVÝ MODEL JE POUŽIT`“.

```
1 JAZYKOVÝ MODEL JE POUŽIT
2         p( JAZYKOVÝ | <s> )          = [2gram] 4.30891e-07 [ -6.36563 ]
3         p( MODEL | JAZYKOVÝ ...)     = [2gram] 0.0389967 [ -1.40897 ]
4         p( JE | MODEL ...)           = [2gram] 0.0222979 [ -1.65174 ]
5         p( POUŽIT | JE ...)          = [2gram] 0.000414834 [ -3.38213 ]
6         p( </s> | POUŽIT ...)        = [2gram] 0.0471584 [ -1.32644 ]
7 1 sentences, 4 words, 0 OOVs
8 0 zeroprobs, logprob= -14.1349 ppl= 671.401 ppl1= 3417.65
```

The two summary lines contain information about number of Out of Vocabulary words (OOVs) and result perplexity of the input text (ppl).

5.2.3 Implementation Details

The keywords subsystem consists of core bash script `keywords.sh` and supply Perl scripts — Words Counter and Probability Counter. At first, there is performed conversion of input text to encoding, which is used in the language model loaded to evaluate probabilities. By default, the script is looking for language model stored in `model.lm` file but other may be specified in command argument. The default character encoding of the language model is ISO-8859-2, because system was tested on Czech text corpora. The ISO-8859-2 encoding is predefined on many places of the system, but it is possible to specify different encoding by command line switch in some scripts. However, in few supplied Perl scripts the encoding is specified directly by directive „use encoding“. Otherwise the system was not working properly. On these places, the encoding needs to be changed manually.

The input text in proper encoding is then processed by the Words Counter, which generates a list of all words found in the text with their counts. The list of words is passed into the `ngram` utility and unigram probabilities are estimated. The same input text is then again passed into the `ngram` utility and the bigram and trigram probabilities are estimated. The outputs of `ngram` utility (described in the previous chapter) are parsed with fast AWK script[4]. The estimated unigram, bigram and trigram probabilities are finally processed by Probability Counter, which evaluates real logarithmic probabilities of two and three words sequences. As discussed in chapter 3.2 about language modeling, the probability of word sequence W can be decomposed as follows:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1}). \tag{5.2}$$

An N-gram language model is estimating conditional N-grams probabilities, thus the probability of sequence of two words is evaluated following way:

$$P(w1, w2) = P(w_2|w_1) \times P(w_1), \tag{5.3}$$

and probability of sequence of tree words:

[4]The whole script (`ngrams.awk`) can be found on attached DVD, same as all the other scripts and parts of the system.

$$\begin{aligned} P(w1, w2, w3) &= P(w_3|w_1, w_2) \times P(w_1, w_2) \\ &= P(w_3|w_1, w_2) \times P(w_2|w_1) \times P(w_1). \end{aligned} \tag{5.4}$$

Actually, the language model is storing logarithmic probabilities, thus the multiplication is substituted with addition and the resulting probabilities are also evaluated in logarithmic domain.

There is another fact needed to be emphasized. The probabilities of terms with different length cannot be directly compared. For instance, a probability of three words sequence can be computed from probability of the first two words sequence, which is multiplied by another probability as shown in equation 5.4. Thus the ranges of probabilities have to differ. The lists of one, two and three words sequences are therefore processed separately and the result lists are merged on the output.

Before sending out, the resulting lists are sorted by the probabilities. Important is that sorting is ascending, because we are interested in terms with the lowest language probability. The terms with lower probability have higher entropy and they are represented worse by the input language model. Mining data related to these terms could thus improve the language model in field related to such terms.

5.3 Extraction of the Web Content

The text mining is performed by the last subsystem called Web Extraction. The input of the subsystem is a list of key terms produced by the previously described subsystem, one term per line. All the terms are thus consequently, one by one, searched on the Web by the Google Engine.

5.3.1 The Web Search

Each input term is placed into a URL address calling Google search. The result is retrieved by `links`[5] WWW browser called with parameter `-source`:

```
1 links -source http://www.google.cz/search?q=$phrase
```

where `$phrase` stands for an input keyword. Calling `links` with `-source` parameter forces the application to not run in interactive mode, but retrieve the source of the Web page and print it to standard output.

Before searching keyword through Google Engine, all the keywords are converted into UTF-8 character encoding, because Google search accepts queries in Unicode. And the keywords are also converted into URL encoding; URL scheme does not accept all characters. This is done by Perl script `urlencode.pl` and the magic line is search and replace pattern:

```
1 $line =~ s/([^A-Za-z0-9])/sprintf("%%%02X", ord($1))/seg;
```

The retrieved Google search result is a standard HTML Web page, which is parsed by `filtergoogleres.pl` Perl script. A structure of the desired page content could be sketched out as follows:

[5] Links is a text mode WWW browser freely available in the most of Linux distributions. See http://links.sourceforge.net/

```
<li class="g">
    <h3 class="r">
        <a href="desired_link" />
    </h3>
</li>
```

where `desired_link` denotes the desired link. The parser thus goes through the HTML tree and look for similar patterns. An output of `filtergoogleres.pl` is a list of URLs found by Google search. It is usually ten links per page. For each key term the first search result page is taken, therefore we get ten links per term.

5.3.2 Documents Retrieval

The next step in mining is recognizing a type of linked document. Binary formats need to be eliminated, so there exists a list of unsupported formats[6], and these documents are skipped. Supported formats are PDF and all the text based sources as HTML, XHTML, etc. The documents which were not ignored are retrieved again with `links` WWW browser. The PDF documents are processed separately by the same procedure as in Material Extraction subsystem (this is indicated by dotted arrow in figure 5.1).

The PDF documents usually contain mostly relevant textual information, which does not need any more sophisticated selection or filtering. On the other hand common Web documents contains a lot of irrelevant information in form of different advertisement fields, title or menu bars, forms or lists not containing a useful information for language modeling.

5.3.3 Relevant Frames

The HTML based documents are processed by `filterpage.pl` Perl script analyzing the document structure and printing out only the textual fields which are considered to be relevant. To analyse the document structure, the Document Object Model (DOM) is used. For that purpose the Perl class `HTML::TreeBuilder` exists. It creates a DOM tree from given HTML document. The class is quite universal and is able to process other XML based formats, e.g. XHTML.

The document tree is then traversed from the root node to all leaf nodes and the content of each node is analyzed whether it is relevant or not. At first, the typically inappropriate nodes are deleted, for instance, `<SCRIPT>`, `<STYLE>`, or nodes used in the Web forms as `<SELECT>` or `<OPTION>`. Then the system deals with typical menu or advertisement bars. Such frames usually contain too little textual information not closed in `<A>` node[7]. Typical menu bar is a list consisting of links pointing to different places in current Web page. And typical advertisement is a frame with some picture and eventual short phrase, but it is formed as a link to another Web domain. Interactive table of contents may be also eliminated by this quite simple procedure.

5.3.4 Post-processing and Formatting of The Output

The DOM tree analyser is also performing pre-processing for sentence splitting done in the last phase of text mining. This is done by adding new line characters in front of and behind

[6] Current list of unsupported formats comprises DOC, DOCX, PPT, PPS, PPTX and RTF.

[7] The `<A>` node is in HTML documents used to link the user to another document or place in current document.

the HTML nodes about which we know that break the structure of the result text, when it is displayed in a browser, e.g. `<p></p>`, `<br>`, `<hr>`, `<tr></tr>`, `<h1></h1>`, etc. Why new line characters are used to separate sentences? Standard utilities to estimate language model require text corpus in form of textual file containing one sentence per line without any punctuation marks.

The last fragment of the mining process is the text filter (implemented in Perl script `filtertext.pl`). It performs the final text cleaning and separation of sentences. At first, dates, times and numbers are converted to spoken form. The main problem with numbers conversion in Czech mining system was that the language has seven cases and the numbers are spoken different way in each case. Also, the gender of counted object changes pronunciation of the number. Thus the numbers conversion is the most interesting part of the text filter. The conversion is based on Perl libraries made by Jan Černocký, which were extended with case and gender determination to put the number into a correct form. The determination is not correct in all cases, but definitely more correct than no one. The example demonstrating conversion of the date, time and numbers is shown below. The sample input was the following:

```
od 14:35
do 20.5.2010
nad 10t
pod 13%
po 2 nohách
přes 1 kopec
```

Ant the output of the text filter is the following:

```
od čtrnácti hodin a třiceti pěti minut
do dvacátého pátý dva tisíce deset
nad deset tun
pod třináct procent
po dvou nohách
přes jeden kopec
```

The case is determined easily from preposition, if there is any. The problem is that not every time the preposition stays directly in front of the number. Sometimes the case depends on deeper meaning of given sentence. Correct determination of the case would require sophisticated semantic analysis, which is difficult for Czech language.

The gender of object is determined by Czech Morphological Analyzer[8]. The problem is to determine with which object the number is connected in given sentence. It would also require difficult semantic analysis. Currently, the word directly following given number is taken as the object.

Additionally, the numbers may be connected with different metric units[9], which are unwrapped to full form, as they should be correctly pronounced.

The next filter step is unwrapping of known shortcuts and special characters[10]. Then the sentence splitting is done — it is performed simply by separating pieces of text by the punctuation marks (dots, question or exclamation mark, dashes, semicolons, etc.). And finally all non-alphanumeric character and multiple white-spaces are removed. Such filtered text should be acceptable for language model estimation.

[8] Morphological Analyzer based on `libma` library with Czech dictionaries was provided by Stanislav Černý and the whole package is also available on attached DVD.

[9] All supported units and shortcuts can be found in file `csshortcuts.pm` on attached DVD

[10] By special characters are meant operators, equal sign, slashes and others.

Chapter 6

The Experiments

6.1 Quantity and Quality of Retrieved Data

The first experiment was aimed to discover whether the data available on the Web can provide some improvement to language modeling. Actually, we can assume that adding some data into an existing language model will never make the model worse, because an optimized interpolation weight is used. The procedure is the following. New language model from the retrieved text data is created and the model is then interpolated with the input language model. The interpolation weight of the new model is determined iteratively, while optimizing perplexity of the resulting language model on development data. When the retrieved data is useless, it will be assigned a low weight and consequently, it will play small or no role in the resulting model. Thus, new data should never make the language model worse.

The point of the experiment was to gather as much data from the Web, as possible, and then analyze quality of the obtained data. Of course, we cannot make any judgments from one testing case on some restricted input data. There were five cases with language model and development data from different fields to examine whether such method can have any profit.

6.1.1 LM Estimation

The language model from retrieved text documents is estimated using MIT Language Modeling (MITLM)[1] toolkit. We could also use the SRILM toolkit, but MITLM toolkit provides more options for interpolation of models. It is important, that the model estimation is restricted with a vocabulary. If we want to compare perplexities of two language models, the models need to be estimated using the same vocabulary.

When we simplify the general perplexity equation 3.25 to trigram language model (the history of conditional probability for word w_i is limited to two preceding words), it can be written as:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-2}, w_{i-1})}}, \tag{6.1}$$

[1]The MIT Language Modeling toolkit is project of Computer Science and Artificial Intelligence Laboratory. More details about the project and how to use it can be found on http://code.google.com/p/mitlm/

while we need to pay attention at the beginning of the sentence W. Conditional probabilities of the first and the second word are evaluated as $P(w_1| < s >)$ and $P(w_2| < s >, w_1)$.

As already noted in section 5.2.2, trigram language model must satisfy:

$$\sum_{w_3} P(w_3|w_1, w_2) = 1; \tag{6.2}$$

sum over all trigrams with same history w_1, w_2 equals to one. When we estimate two language models from the same text corpus, each using different vocabulary, different amount of words following the same history w_1, w_2 may occur. Consequently, the probability distribution over all possible words following sequence w_1, w_2 may also differ and perplexity of these models may acquire incomparable values. Imagine situation, in which we use a vocabulary with more words: the trigram probabilities $P(w_3|w_1, w_2)$ may be potentially distributed over more words w_3 and the probabilities will acquire lower values. Consequently, while evaluating perplexity of trigram language model, multiplication of inversed probabilities in equation 6.1 will lead to higher perplexity. That is the primary effect of out of vocabulary words. For instance, if we create a language model with tiny vocabulary, it can have low perplexity, but it would be completely useless in any recognition system, because the resulting WER would be very high.

6.1.2 LM Interpolation

MITLM toolkit is also used to interpolate language models. The toolkit provides advanced options to perform optimal language model interpolation. When interpolating language models, the interpolation weights need to be specified. The weights tell us what role will play each model in resulting interpolated model. The interpolation from MITLM toolkit can iteratively determine the weights to optimally model predefined development data. The inputs of interpolation are two language models and development data, on which the interpolation weights are optimized to obtain the best perplexity of resulting interpolated model.

We are using basic linear interpolation in the following experiments:

$$P_i(W) = (1 - \lambda) \times P_1(W) + \lambda \times P_2(W), \tag{6.3}$$

where $P_i(W)$ is probability of N-gram W in the resulting interpolated model and $P_1(W)$ denotes probability of W in the first language model and $P_2(W)$ denotes probability of W in the second model. When interpolating two language models, only one interpolation weight λ need to be specified. The weight of the second model is $(1-\lambda)$, because the weights must satisfy, that they sum to one.

6.1.3 Input Data

Since the text filtering and retrieval covers quite complex procedures, the process of gathering data takes a long time. We have evaluated more testing cases with smaller sparse language models and input data simulating several situations. However, all the used data are provided by speech processing group Speech@FIT[2]. Text corpora were used for input language models estimation from the sources listed in table 6.1.

[2]Speech@FIT is a group active in speech recognition research at Faculty of Information Technology of BUT. More information on http://speech.fit.vutbr.cz/

Internal Name	Description	Tokens	Sentences	Size
cswiki	Texts from Czech Wikipedia	16 212 263	1 115 902	103 M
cz_anotace	Annotation data of short sequences from several courses taught at FIT	44 851	5 546	246 K
cz_podpory_v4	Text acquired from materials for courses taught at FIT	2 573 375	495 340	17 M
lect	Transcriptions from three complete courses (IRP, ISS, MUL) taught on FIT	183 305	13 872	1.1 M
MO_ER	Transcriptions from telephone calls to different radio stations	186 126	20 865	978 K
MO_E123	Various telephone calls	548 846	78 061	2.5M
MUNI	Text materials and notes for lectures taught at Masaryk University	193 866	17 320	1.2 M
MV_EV	Transcriptions of telephone calls	45 297	6 477	223 K
PMK&BMK	Prague spoken corpus + Brno spoken corpus	1 181 689	89 036	5.9 M
subtitles	Collection of movie subtitles	191 861 852	36 169 922	997 M

Table 6.1: Source corporas details

The data from these sources were merged into two corpora splitting general text and more technically oriented text. The composition of corpora is marked in table 6.2.

	Source Corpora
General spoken transcriptions	MO_ER, MO_E123, MV_EV, PMK&BMK
Technical texts	cz_podpory_v4, lect, MUNI

Table 6.2: Testing corpora composition

Using these two corpora, we examined four test cases. One test case estimates input language model from larger part of the general spoken corpora. Then the rest of the data is taken as development data and servers as input for text mining. In the second experiment, the same input language model as in the first experiment is used, but the development data and input for text mining is taken from the technical texts. Such two cases should compare whether the Web based texts can improve only the language models based on data from different domain, or if they can improve language model which already describes given development data quite well. The next two experiments perform similar procedure, but the input language model is estimated from technical texts. This should show whether the Web is better source of spoken form or technical texts.

Finally, we examined a large test case with data and language model estimated for real application. The input language model was created from all text corpora listed in table 6.1 with one exception: the text corpus gathered from subtitles was not directly merged into the model. All the corpora except the subtitles one were merged together and language model was estimated by SRILM `ngram-count` command. From the subtitles corpus a language model was also estimated, but with `-prune`[3] parameter set to 10^{-7}. These two language

[3]The `-prune` parameter forces pruning N-grams with low probabilities. See manual page of ngram-count command. Online available on http://www.speech.sri.com/projects/srilm/manpages/ngram-count.1.html

models were interpolated, where the weight of the first model was set to 0.85. This language model was created for project of Lecture Browser[4] running on FIT. The purpose of this final test case was to improve the lectures recognition system.

The development data and input for text mining for the final test were selected from old annotation corpus[5]. The annotation corpus was randomized. It means that the lines (sentences) of the corpus were randomly reordered to eliminate topics separation, when splitting the corpus. The reordered corpus was split, so that 2 276 sentences were taken as the development data and 2 244 sentences were taken as the input for text mining. More details about input data for all test cases are listed in table 6.3 and details about input language models are listed in table 6.4.

Source	Part	Tokens	Sentences	Size
General spoken transcriptions	LM base	1 865 405	184 530	9.1 M
	Development data	20 280	2 054	101 K
	Text mining input	20 170	2 034	101 K
Technical texts	LM base	1 415 851	252 803	9.1 M
	Development data	15 370	2 771	101 K
	Text mining input	15 382	2 660	101 K
Annotation texts	Development data	18 230	2 276	101 K
	Text mining input	18 324	2 244	101 K

Table 6.3: Input Text Data Detail

Language Model	Bigrams	Trigrams	Size
General spoken form texts LM	614 795	1 210 510	56 M
Technical texts LM	665 228	1 114 456	61 M
Lectures Browser LM	7 820 386	4 334 160	323 M

Table 6.4: Input Language Models

The last important note about the input data is that all the operations (e.g. language model estimation, Web pages content filtering, or selection of keywords) are restricted with the same vocabulary. This vocabulary is the same as the one used in the Lectures Browser project. It contains 313 220 Czech words, including different forms of words coming from diversity of the language.

6.1.4 Results of Text Mining

For each test case, three input files were used — language model or text corpus to create new model, input for text mining and development data. The mining system was given the input for text mining. It filtered the text, extracted keywords and tried to retrieve required amount of text from the Web as described in chapter 5. In the first four test cases, the required amount of data was specified approximately to be twice more than the size of LM base texts. In the last case, around hundred megabytes of plain text was gathered. The baseline Lecture Browser language model is much larger, but acquiring such

[4]The project Lecture Browser can be found on address http://www.prednasky.com

[5]Annotation data are sets of speech audio records tied together with their manual transcription annotated by humans.

amount of text data would take an extremely long time, moreover, this amount of data should tell us something about usefulness of this experiment. The acquired textual data was then randomized similarly as the input corpora and several sets with different length were created. Each set was finally used to estimate new language model and this model was interpolated with the input language model to evaluate perplexity improvement.

The details about all sets and perplexity evaluations are listed in tables 6.5 to 6.9, separately for each test case.

	Input or Web Data			**Interpolated LM**	
Data set	Tokens	Sentences	**Perplexity**	**Weight**	**Perplexity**
Input LM	1 865 405	184 530	334.6		
90 *k*	15 505	1 080	2 592	−3.055	333.0
400 *k*	68 640	4 840	1 813	−2.798	331.8
900 *k*	154 517	10 865	1 499	−2.609	330.8
1 800 *k*	309 491	21 833	1 315	−2.531	330.0
4 *M*	704 391	49 693	1 162	−2.324	327.4
9 *M*	1 585 324	111 939	1 059	−2.157	324.5
18 *M*	3 168 803	224 240	992	−2.015	320.7

Table 6.5: Test 1: Adapting general spoken LM.

	Input or Web Data			**Interpolated LM**	
Data set	Tokens	Sentences	**Perplexity**	**Weight**	**Perplexity**
Input LM	1 865 405	184 530	11 679		
90 *k*	14 554	1 051	6 685	1.079	5 255
400 *k*	64 762	4 579	3 987	1.912	3 649
900 *k*	145 635	10 166	3 095	2.352	2 945
1 800 *k*	291 338	19 944	2 568	2.642	2 482
4 *M*	664 737	45 588	2 117	2.852	2 065
9 *M*	1 495 751	102 188	1 790	2.869	1 749
18 *M*	2 992 418	204 399	1 598	2.765	1 558

Table 6.6: Test 2: Adapting general spoken LM to technical area.

In the results for the first four test cases, we can see that the data retrieved from the Web are providing some improvement in comparison to the input model. In tests 2 and 4, the improvement of perplexity is significant, because we are trying to adapt the language model to completely different domain, than it was originally describing. You can also notice that even the stand-alone language models created from retrieved Web data are better than the input model. The interpolation weights are positive, when the language model from retrieved data better fits the development data than the baseline model. The weights are in logarithmic base. The linear interpolation of two models was already described in chapter 6.1.2. The interpolation is performed by summing weighted probabilities of corresponding N-grams from both models. However, MITLM is representing the weights different way. Weight of the input model is predefined to 0 logarithmically, thus in linear base $t_{LM_1} = 10^0 = 1$, and weight of the second model is in linear base $t = 10^{weight}$. The probability P_i of N-gram W in interpolated model is evaluated as follows:

	Input or Web Data			Interpolated LM	
Data set	Tokens	Sentences	**Perplexity**	**Weight**	**Perplexity**
Input LM	2 950 546	526 532	546.9		
90 *k*	14 788	991	7 519	−5.362	546.8
400 *k*	65 487	4 255	5 136	−4.408	546.1
900 *k*	147 386	9 872	4 131	−3.610	544.4
1 800 *k*	294 611	20 134	3 483	−3.415	543.2
4 *M*	670 518	45 668	2 965	−3.033	539.4
9 *M*	1 510 057	102 514	2 573	−2.696	533.9
18 *M*	3 020 173	204 972	2 296	−2.390	526.4

Table 6.7: Test 3: Adapting technical LM.

	Input or Web Data			Interpolated LM	
Data set	Tokens	Sentences	**Perplexity**	**Weight**	**Perplexity**
Input LM	2 950 546	526 532	3 428		
90 *k*	15 127	1 095	3 141	0.622	2 218
400 *k*	67 822	4 795	2 056	1.338	1 768
900 *k*	152 425	10 716	1 670	1.738	1 520
1 800 *k*	304 989	21 413	1 478	1.972	1 382
4 *M*	693 772	48 798	1 331	2.110	1 260
9 *M*	1 561 595	110 313	1 239	2.112	1 175
18 *M*	3 125 496	221 468	1 182	1.957	1 113

Table 6.8: Test 4: Adapting technical LM to general spoken area.

$$P_i(W) = \frac{P_1(W) + t \times P_2(W)}{1 + t}, \tag{6.4}$$

where $P_1(W)$ and $P_2(W)$ are probabilities of N-gram W in both input models. It is basically the same as in equation 6.3, but λ is rewritten as $\lambda = \frac{t}{1+t}$.

The relation of added tokens with perplexity reduction in test 2 and 4 is shown in figure 6.1. The perplexity in both cases is approaching to the limit of perplexity, which can be achieved by adding Web data. While the Web data dominate in the resulting model, the limit value is basically the perplexity, which Web data can achieve.

In tests 1 and 3, we can see improvement of perplexity, even if the model describes the development data quite well. But the baseline models were made relatively small by purpose – to examine whether such system can provide some improvement. For tiny models, this can be one option, how to easily gather some new data for language model. The perplexity

	Input or Web Data			Interpolated LM	
Data set	Tokens	Sentences	**Perplexity**	**Weight**	**Perplexity**
Input LM	–	–	132.527		
10 *M*	1 693 149	120 241	1 537	−8.968	132.537
50 *M*	8 466 615	599 638	1 292	−8.423	132.536
100 *M*	16 931 970	1 198 525	1 229	−7.856	132.535

Table 6.9: Test 5: Adapting Lecture Browser LM.

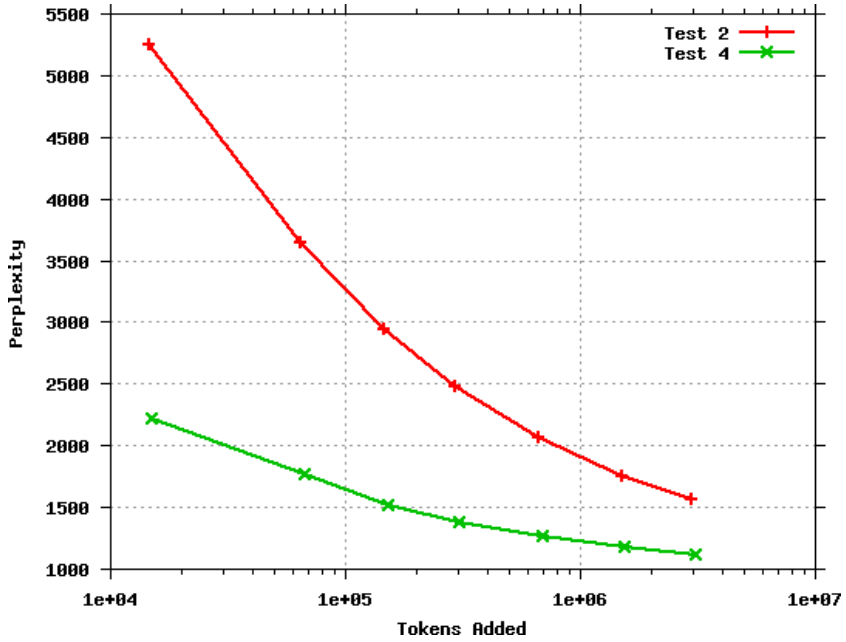


Figure 6.1: Perplexity reduction in test 2 and 4.

reduction in test 1 and 3 is illustrated in figure 6.2. You can notice that the behaviour is slightly different than in case 2 and 4. The Web data are getting more significant as its weight is higher and the amount of tokens exceeds the baseline model. However, with amount of Web data far overriding the baseline model, the situation would be similar to the case 2 and 4.

When examining the result of the last test case, we can see that for large language model which has already quite low perplexity, the retrieved data is useless. More sophisticated filtering methods of the text can probably help, but anyway the required effort would be far from the possible gain.

6.1.5 Experiment on OOV Reduction

In the last text case in the previous section we examined, that to retrieve huge amount of Web data is not a way, to improve the perplexity of a good language model. This experiment is not limiting the language models with predefined vocabulary, but it is trying to improve the model by OOV error rate reduction. OOV error rate was introduced in chapter 3.5.5.

We examined a test case with older lectures language model, which is missing the telephone data and some new lecture materials. The baseline model consist of texts from sources listed in table 6.10.

The same Web data as in the last test case from previous section were reused in this experiment. To examine more situations, we also used a language model estimated from the large subtitles corpus mentioned in table 6.1. It provides a large vocabulary; hence we can examine more significant changes in the results. Three language models were created from described data. More details about the models are in table 6.11. Perplexity and OOV error rate of each model were also evaluated on predefined development data and they are in the table as well. The development data was also reused from the previous test case.

The baseline language model has quite good perplexity compared with the other two

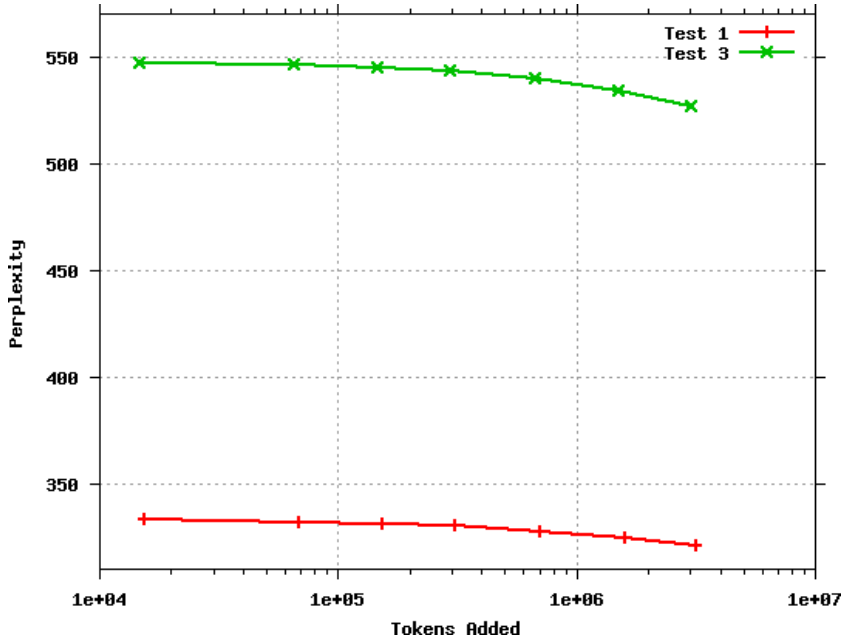


Figure 6.2: Perplexity reduction in test 1 and 3.

models, but the OOV rate is high. The development data based on the annotation corpus has 18 230 words, thus the OOV rate is 8.1%. The model based on Web data has larger vocabulary, hence the OOV rate is lower (3.6%), but the perplexity is worse in compare to the baseline model. The large subtitles model is the worst describing technically oriented annotation data, and although it has large vocabulary consisting of more than a million words, the OOV rate is also quite high (8.2%). However, the content of subtitles is quite general, so it could also help to the final recognition system.

Since we want to interpolate language models based on different vocabularies, the MITLM toolkit cannot be used. MITLM can interpolate only models based on the same vocabulary. The SRILM toolkit was used, which in other hands does not automatically determine the interpolation weight. Thus the weights were determined empirically. We examined several tests, which combine prepared models. The best results are listed in table

Internal Name	Description	Tokens	Sentences	Size
cz_podpory_v2	Text acquired from materials for courses taught at FIT	1 999 832	118 768	12 M
lect	Transcriptions from three complete courses (IRP, ISS, MUL) taught at FIT	185 469	13 872	1.1 M
MUNI	Text materials and notes for lectures taught at Masaryk University	193 866	17 320	1.2 M
PMK&BMK	Prague spoken corpus + Brno spoken corpus	1 181 689	89 036	5.9 M
subtitles	Collection of movie subtitles	3 817 322	618 680	20 M

Table 6.10: Source corporas details

Language Model	Words	Bigrams	Trigrams	Size	OOVs	Perplexity
Baseline LM	240 595	2 219 116	752 191	80 M	1 485(8.1%)	524.1
Web LM	475 033	5 477 534	1 409 012	101 M	654(3.6%)	1 005.5
Subtitles LM	1 000 781	3 433 540	3 185 922	196 M	1 497(8.2%)	2 426

Table 6.11: Input Language Models

6.12 and they are described below.

Case	Models	Weights	Perplexity	OOVs	OOV ⇓
1.	Baseline + Subtitles	0.88, 0.12	624.4	1 213	−1.4%
2.	Baseline + Web	0.65, 0.35	631.4	627	−4.7%
3.	Baseline + Web + Subtitles (1 267 843 words)	0.62, 0.34, 0.04	636.7	608	−4.8%
4.	Baseline + Web + Subtitles (limited to 685 410 words)	0.62, 0.34, 0.04	616.8	649	−4.5%
5.	Baseline + Web + Subtitles (limited to 469 155 words)	0.62, 0.34, 0.04	593.8	707	−4.2%
6.	Baseline + Web (log-linear interpolation)	0.65, 0.35	420.8	627	−4.7%

Table 6.12: Input Language Models

In the first case, the baseline language model was combined with the subtitles model. An optimal interpolation weight of the baseline model was determined to 0.88. The interpolation weight is not logarithmic as in case of MITLM, it is directly λ as described in equation 6.3. We can see, that the OOV rate was slightly reduced by 1.4% and because of that, the perplexity raised.

Then the baseline model was interpolated with model based on retrieved data from the Web. In this case the situation is more interesting, because the OOV error rate was reduced more than twice by 4.7%. In the third case, the OOV rate was reduced even more. All three input models play role in this case. However, because the subtitles language model is interpolated, the resulting model will have a huge vocabulary. This may lead to more frequent confusion of the recognition system using such model.

Regarding this fact, the same experiment was done in fourth case, but the language model estimation was limited by predefined vocabulary. The vocabulary was initialized with all words covered by the input corpora. The words were sorted by their occurrence in the corpora and those which appeared only once or twice were removed. The vocabulary was reduced almost twice by this simple procedure. The OOV rate slightly raised in compare with the third case, but the perplexity was reduced.

In the fifth case, the vocabulary was limited even more than in the previous one. All words which occurred in the corpora less than six times were removed. Thus the final vocabulary consists of 469 155 words. Perplexity of this model is again lower, but still the OOV rate was significantly reduced.

The first five cases were using basic linear interpolation, while the last case shows result of log-linear interpolation of language models. As described in [12], log-linear interpolation is better option, to interpolate independently trained language models. All three input language models were trained separately and each model is based on completely different

source. The OOV error rate was in this case reduced by 4.7% and even the perplexity was reduced from 524 to 421.

6.2 Evaluation of Usefulness of Web Documents

The search engine gives us a set of documents, which are relevant to some specified query. The queries in described mining system are composed from few words, thus the search engine cannot guarantee, that the returned documents are really those, which could be relevant to the development data. And even the queries would be specified more accurately, the World Wide Web is full of different documents from different authors and the search engines are not investigating, whether the documents are correct from the language point of view or not.

Hence a system evaluating relevancy of retrieved Web documents is available. The inconvenient documents are filtered out and only those considered to be good are use to estimate the final language model. The system is described below.

6.2.1 Relevancy Measure System Concept

The concept of system filtering inconvenient documents is demonstrated in figure 6.3. The primary input of the system is the set of text documents retrieved by the mining system described in chapter 5. Beyond that, existing language model need to be specified. The language model is used as a base for relevancy evaluation. Also, some development data and vocabulary are required. The development data is needed to evaluate perplexity of language model and vocabulary is used for language model estimation to be able to compare different models. The process of language models estimation and comparing was already described in chapter 6.1.1.

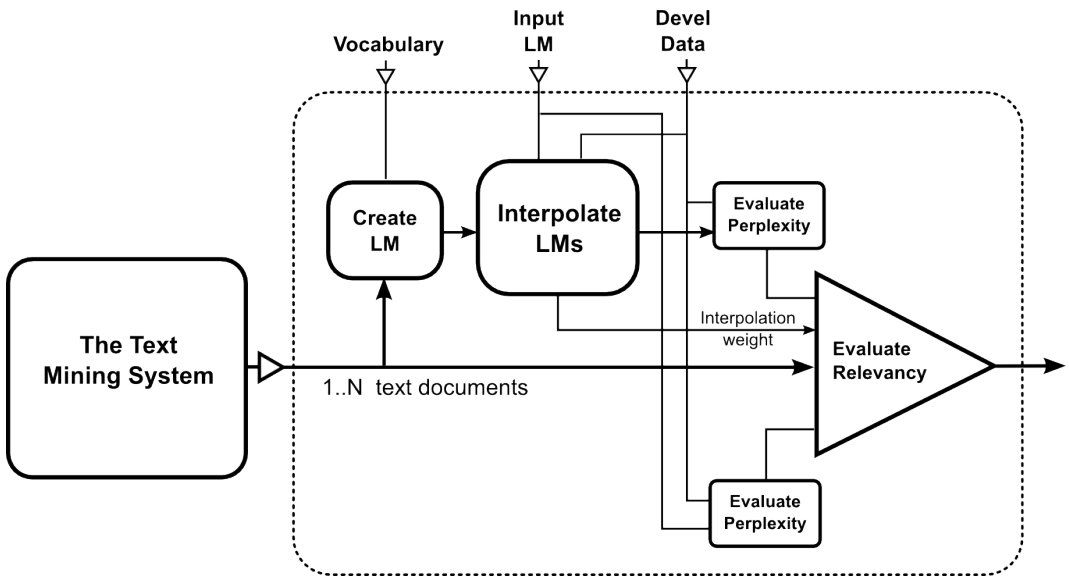


Figure 6.3: The concept of system evaluating relevancy of retrieved text documents and filtering inconvenient documents.

The main idea is that from each clean text document a small language model is created.

This model is interpolated with the input language model, which we intend to improve. The perplexity of the new language model is then evaluated and compared with the perplexity of input model. The interpolation weight of the small language model created from the input text document also plays a role. If the weight is too low, the text document is probably completely useless and can be thrown away.

6.2.2 Evaluation of the Improvement

The final decision, whether the given text document will be included into the output text corpus, can be based on the interpolation weight of the small language model created from the text document. Another option is to make the decision on perplexity of interpolated model evaluated on given development data. The last option is to compare perplexity of the small language model with predefined threshold. All three options are working with relative parameters, thus their values need to be determined empirically. An experiment was done, in which the threshold of small language model perplexity was adjusted. The results of experiment are in table 6.13.

Input LM				Perplexity: 334.599	
Threshold	Documents	Relevant	Weight	Web PP	Final PP
–	826/826	100%	−2.328	1 123	325.837
20 000	582/1 049	55%	−2.274	1 073	325.593
15 000	514/1 177	44%	−2.205	1 024	325.106
11 000	439/1 404	31%	−2.170	1 003	324.678
8 000	335/1 803	19%	−2.127	985	324.353
6 000	213/1 961	11%	−2.141	975	324.959
5 000	164/2 525	7%	−2.106	949	324.409

Table 6.13: Filtering irrelevant documents.

Perplexity of the language model based on the Web document is quite good criterion of document relevancy. The perplexity of Web data model is getting lower as the threshold is adjusted. Perplexity of the final interpolated model also getting lower, but as the number of relevant documents is getting too low, the perplexity even grows. The relation of number of documents needed to retrieve on perplexity reduction is shown in figure 6.4. The documents relevancy filtering is potential option how to increase the quality of retrieved data, but it is compensated for the time required to gather more documents.

6.3 Update of Vocabulary

An intention is also to update the current vocabulary and pronunciation dictionary with data acquired from the Web. The designed system is including several scripts extracting and evaluating new words not covered by the current vocabulary. The scripts are formed into a small subsystem sketched in figure 6.5.

The input of the scripts is text retrieved by the designed mining system from the Web. The text prepared by the mining system consists of plain text without any punctuation markers, numbers, or other special characters, and it is also converted into uppercase. Hence the word units can be identified easily. The Words Counter was already introduced in the mining system described in chapter 5. It separates words in the input text and then it counts occurrences of all words covered by the text. The Filter reads predefined vocabulary

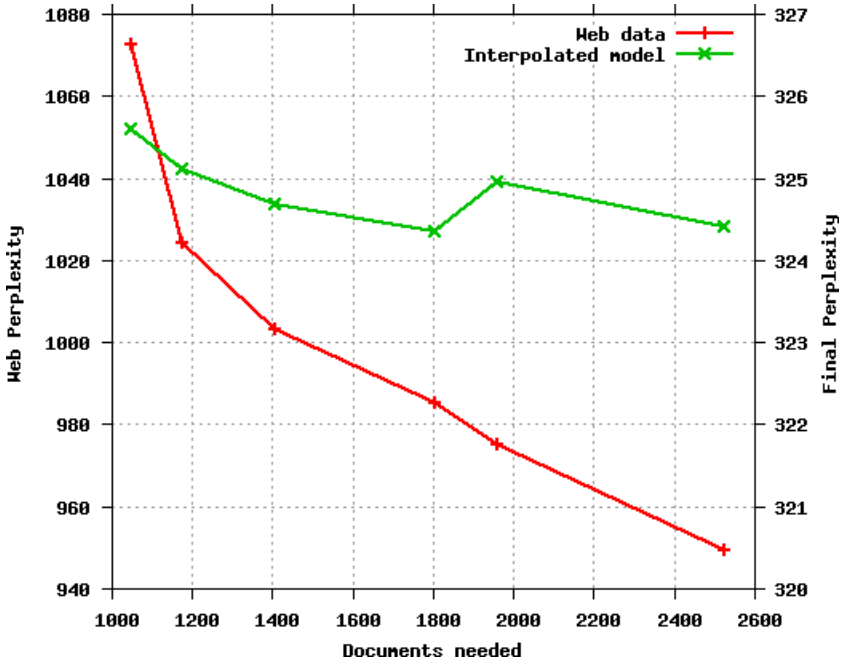


Figure 6.4: Perplexity reduction when filtering irrelevant documents.

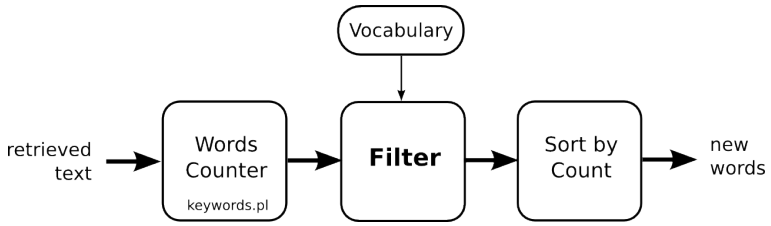


Figure 6.5: The concept of system extracting and evaluating new words.

(that one, which we want to update), and all the words included in the vocabulary are removed from the list, which is produced by the Words Counter. The filtered list is finally ordered by number of occurrences.

6.3.1 Disputable Content of The Web

The data gathered during the tests described in chapter 6.1 were also used in this experiment. Each of the final text corpora from retrieved Web documents were analyzed for new words separately. In the first four test cases, more than 80 thousand of new words were discovered and in the last test case, almost 300 thousand of new words appeared. More details are shown in table 6.14.

Actually, we should not call the acquired tokens words. Mostly, misspelled words or sequences of characters not giving any sense were found. Very often, new shortcuts, which were not unwrapped by the mining system, appeared. However, it would be improper to unwrap some shortcuts, which are usually spelled the same way, as they are written. Majority of the new tokens cover misspelled words or words written without diacritical

Data	Tokens	$> 1000\times$	$> 100\times$	$> 10\times$	$> 0\times$
Test 1	3 301 714	4	112	1 715	91 256
Test 2	3 516 204	18	163	2 320	85 387
Test 3	3 020 210	11	83	1 384	82 603
Test 4	3 125 015	7	119	1 664	86 441
Test 5	16 950 045	63	722	9 000	294 802

Table 6.14: Counts of new words discovered in retrieved Web data.

marks. They were probably found in user comments and forums. There can be found some correctly written words, but they are mostly colloquial or unidiomatic words.

We tried to extract unknown tokens from development data for each of the test case. These tokens were compared with new tokens found in the retrieved data. The vocabulary was updated only with those tokens, which appeared in both sources. And finally, the perplexities and OOV rates of new language models were evaluated with the new vocabulary. This was done for the first four test cases and the results are shown in table 6.15.

		Old Vocabulary		New Vocabulary		
Data	**Words**	Perplexity	OOVs	Perplexity	OOVs	**OOV** $\Downarrow$
Test 1	20 280	322.4	598 (2.95%)	331.1	368 (1.81%)	1.14%
Test 2	15 370	1 558	118 (0.77%)	1 562	106 (0.69%)	0.08%
Test 3	15 370	526.4	118 (0.77%)	526.5	106 (0.69%)	0.08%
Test 4	20 280	1 113	598 (2.95%)	1 161	368 (1.81%)	1.14%

Table 6.15: Perplexity and OOV rates of interpolated LMs evaluated with updated vocabulary.

You can notice that in cases 1 and 4 the OOV error rate was reduced more than in cases 2 and 3. In the first two mentioned cases, the general spoken corpora were used to select the development data. In the next two cases, the technical texts were used. The diversity is caused by the specialization of the vocabulary which was used to estimate the language model. The Lecture Browser vocabulary, which is more technically oriented for purpose of the lectures, was used. Hence there is a lack of colloquial words which were added into the vocabulary. However, in all cases the OOV error rate was reduced, while the perplexity increased. The problem of OOVs was explained in chapter 6.1.1.

Using the described procedure, we can easily reduce the OOV rate of the final language model. But then we need to estimate a new language model with the updated vocabulary. When a new word is added into the vocabulary and language model, it needs to be added into the acoustic model and pronunciation dictionary as well. Otherwise the recognition system is not able to use the new word. The pronunciation of new word can be generated automatically with predefined procedures derived from linguistic rules of given language. But the result may not be absolutely correct.

6.4 Test on Real Recognition System

As already noted in chapter about language modeling, quality of recognition system is measured by Word Error Rate. To measure WER we need to have some annotated data; that is a set of audio records with their literal transcriptions. The audio records are processed

with the recognition system we are evaluating for WER and the produced transcriptions are compared with the original transcriptions.

A set of annotated data for this experiment consists of 16 records of lectures from different courses taught on Faculty of Information Technology. Each record is a short segment of some lecture, which is approximately 5 minutes long. That is 80 minutes of annotated speech.

By measuring WER we can verify whether the improvement on language modeling is important or not. The conclusion of the first experiment was that the text mining can provide some improvement into language model. So the WER was evaluated on recognizer using the baseline model and then on recognizer using model updated with the retrieved data. These two values are compared. The results are not available at this moment, but they will be demonstrated during the defence of this thesis.

Chapter 7

Conclusion

The purpose of this project was to get familiar with language modeling for speech recognition and techniques for acquisition of text data from the Web. At first, the speech recognition techniques were introduced in general. The text focused to statistical speech recognition methods, concretely Large Vocabulary Continuous Speech Recognition. After the basic formulations are introduced, language modeling is explained in detail. Attention is paid to properties for quality measure of the language models, namely OOV error rate, Perplexity and Word Error Rate.

Mining of textual data was introduced in the next chapter. After defining what mining means and what types of mining exist, information retrieval was described. Models of information retrieval are also covered. The last part discusses problems in the Web content mining and Google search with its PageRank algorithm is noted.

The project covers implementation of system acquiring text data from the Web. The architecture of designed system was described in chapter 5. A lot of details are explained with sample pieces of code. The primary problem was to deal with encoding of retrieved documents. The HTML documents specify encoding in header, but for other types of document the automatic detection was built into the system. However, the major part of retrieval system was to determine the relevant and correct textual content of the Web documents.

The experiments described in the last chapter represent the critical part of the work. Several test cases, which evaluate the Web data usability in language modeling, were examined. The experiments demonstrate that the Web based data can adapt language model to another domain. However, the perplexity can be reduced with in-domain textual data as well. For tiny language models the adaptation works quite well. The sample general spoken language model was adapted to technical area and the perplexity was reduced from 11 679 to 1 558. The perplexity in the test case using the in-domain data was reduced from 335 to 321. The Web base data were also used to improve language model on which the Lecture Browser recognizer is based. The OOV error rate of the model was reduced with Web data by more than 4% absolutely. And using log-linear interpolation the perplexity of the model was reduced to 421 from original 524.

In another experiment we examined, how document relevancy evaluation can help to language modeling. Filtering irrelevant documents can reduce the perplexity, however, more than 50% of Web documents were discarded to reduce the perplexity from 325.8 to 325.1. The last experiment shows what the problem of the Web content is. It can reduce the OOV error rate, but improper words may be added.

The whole system implementation is attached on data DVD. Retrieved Web data and

prepared Web corpora are attached as well. The content of the DVD is described in read-me file, which is located in root directory.

Bibliography

[1] Jesse Alpert and Nissan Hajaj. We knew the web was big... http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html, 2008-07-25 [cited 2010-04-17].

[2] Quraish Bakir. Automatic speech recognition and understanding. http://ewh.ieee.org/r10/bombay/news6/AutoSpeechRecog/ASR.htm, [cited 2010-01-02].

[3] Jim Baumann. Voice recognition. http://www.hitl.washington.edu/scivw/EVE/I.D.2.d.VoiceRecognition.html, [cited 2010-01-02].

[4] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan Kaufmann Publishers, Indian Institute of Technology, Bombay, 2003. ISBN-1-55860-754-4.

[5] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. http://research.microsoft.com/en-us/um/people/joshuago/tr-10-98.pdf, August 1998 [cited 2010-04-10].

[6] Wu Chou and Bing Huang Juang, editors. *Pattern Recognition in Speech and Language Processing.* CRC Press, Inc., Boca Raton, FL, USA, 2002. ISBN-0849312329.

[7] Thomas M. Cover and Thomas Joy A. *Elements of information theory / 2nd ed.* Wiley-Interscience, New York, 2006. xxiii. 748 p., ISBN-0-471-24195-4.

[8] Joshua Goodman. The state of the art in language modeling. http://research.microsoft.com/en-us/um/people/joshuago/publications.htm, 2000 [cited 2010-01-04].

[9] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques.* University of Illinois at Urbana-Champaign, 2006, 2nd edition. ISBN-1-55860-901-6.

[10] Frederick Jelinek. *Statistical methods for speech recognition.* MIT Press, Cambridge, MA, USA, 1997, second printing 1999. ISBN-0-262-10066-5.

[11] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Second Edition.* 2008. 1024 p., ISBN-0-131-87321-0.

[12] Dietrich Klakow. Log-linear interpolation of language models. http://www.shlrc.mq.edu.au/proceedings/icslp98/PDF/AUTHOR/SL980522.PDF, [cited 2010-05-24].

[13] Peter E Brown; Vincent J. Della Pietra; Robert L. Mercer; Stephen A. Della Pietra; Jennifer C. Lai. An estimate of an upper bound for the entropy of english. http://acl.ldc.upenn.edu/J/J92/J92-1002.pdf, 1992 [cited 2010-04-15].

[14] Vipin Kumar Pang-Ning Tan, Michael Steinbach. Data mining: Introduction, lecture notes for chapter 1, introduction to data mining. http://www-users.cs.umn.edu/~kumar/dmbook/, 2004-04-18 [cited 2010-01-06].

[15] Markus Sobek. A survey of google's pagerank. http://pr.efactory.de/, 2003 [cited 2010-01-08].