Západočeská univerzita v Plzni
Fakulta aplikovaných věd

ADAPTACE JAZYKOVÉHO MODELU NA TÉMA V REÁLNÉM ČASE

Ing. Jan Lehečka

disertační práce
k získání akademického titulu doktor
v oboru Kybernetika

Školitel: Prof. Ing. Josef Psutka, CSc.

Katedra kybernetiky

Plzeň 2019

UNIVERSITY OF WEST BOHEMIA
FACULTY OF APPLIED SCIENCES

ONLINE TOPIC-BASED LANGUAGE MODEL ADAPTATION

Ing. Jan Lehečka

doctoral thesis
submitted in conformity with the requirements
for the degree of Doctor of Philosophy
in the field of Cybernetics

Supervisor: Prof. Ing. Josef Psutka, CSc.

Department of Cybernetics

Plzeň 2019

Anotace

Tato disertační práce se zabývá adaptací jazykového modelu na téma v reálném čase. Jde o mechanismus navržený pro snížení chybovosti automatického rozpoznávače řeči v úlohách živého přepisu vícetématických promluv, kde obecný jazykový model není schopen dostatečně popsat rozdílné statistiky posloupností slov v jednotlivých tématech. Základní myšlenka spočívá v dynamickém přizpůsobování jazykového modelu během živého rozpoznávání na základě tématu detekovaném v rozpoznané řeči.

Nejprve je shrnut aktuální stav poznání této problematiky doplněný detailním teoretickým základem pro použité metody a modely. Popsané metody zpravidla kombinují dvě významné výzkumné oblasti: automatické rozpoznávání řeči v reálném čase a automatickou identifikaci tématu.

Poté je navrženo inovativní rozšíření existujícího automatického rozpoznávače řeči o adaptaci jazykového modelu na téma v reálném čase. Originalita navrženého řešení spočívá především v minimalizaci prodlevy adaptace na téma díky paralelnímu běhu dvou dekodérů (obecného a tématického) zároveň a následnému spojení obou výstupů, což vede ke snížení chybovosti slov při živém rozpoznávání řeči.

Navržený adaptabilní systém byl implementován a otestován na dvou vícetématických problémech: živý přepis televizního zpravodajství a živý přepis televizních sportovních přehledů. Experimenty v této práci v obou případech prokázaly, že navržený systém pracuje významně lépe než neadaptabilní systém a že adaptace jazykového modelu na téma snižuje chybovost živých přepisů, zejména pak vlastních jmen úzce spjatých s jednotlivými tématy.

Annotation

The research area of this thesis is online topic-based language model (LM) adaptation. It is a mechanism designed to reduce word error rates of real-time automatic speech recognition (ASR) systems in multi-topic tasks, where a general LM cannot model varying word sequence statistics in particular topics appropriately. The base idea is to dynamically adjust the LM during live decoding based on topics detected in the decoded transcripts.

First, the thesis surveys the state of the art of the problem including also detailed theoretical background of used methods and models. Described methods usually combine two very important research areas: real-time automatic speech recognition and automatic topic identification.

Next, an innovative solution to extend existing real-time ASR system by online topic-based LM adaptation is proposed and described in details. The originality of proposed solution lies primarily in minimizing latency of the topic-based adaptation by using two parallel decoders (general and topic-specific), and online merging their outcomes in order to reduce word error rate during online speech recognition.

The proposed adaptable system was implemented and tested for two multi-topic real-time ASR problems: live transcription of TV news and live transcription of TV sports summaries. For both problems, experiments in this thesis showed that proposed system performs significantly better than a system without LM adaptation, and that topic-based LM adaptation can reduce error rates of live transcripts, especially by better recognizing topic-specific proper nouns.

Declaration of Authorship

I hereby declare that this doctoral thesis is completely my own work and that I used only the cited sources.

Pilsen, February 25, 2019

. . . .

Jan Lehečka

Acknowledgements

On this place, I would like to thank to my supervisor, prof. Ing. Josef Psutka, CSc., for giving me the opportunity to study this interesting problem and for the motivation to my work. Big thanks belongs also to my consultant-specialist, Ing. Aleš Pražák, Ph.D., whose valuable advices during numerous consultations were essential for the development of this work. Thanks also to my colleagues from Department of Cybernetics for their cooperation, support and ideas.

Finally, I would like to thank a lot to my family, especially to my wife Zdeňka, for all the support and almost limitless tolerance during my long-lasting doctoral study.

The work has been supported by the grant of the University of West Bohemia, project No. SGS-2016-039.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

Contents

Contents **i**

List of Figures **v**

List of Tables **vii**

Abbreviations **ix**

1 Introduction **1**

1.1 Outline of the Thesis 2

2 Objectives of the Thesis **5**

3 Theoretical Background **7**

3.1 Automatic Speech Recognition 7

3.1.1 Acoustic Features Extraction 8

3.1.2 Acoustic Model 9

3.1.3 Language Model 10

3.1.4 Lexicon 12

3.1.5 Decoder 13

3.1.6 Implementation 14

3.1.7 Evaluation 15

3.2 Topic Identification 16

3.2.1 Text Preprocessing 17

3.2.2 Document Representation 18

3.2.3 Text Classification 19

3.2.4 Topic Models 22

3.2.5 Multi-label TI 24

3.3 LM Adaptation 25

3.3.1 Linear Interpolation 26

3.3.2 MAP Adaptation 26

3.3.3 Language Model Fill-Up 27

3.3.4 Constraint Specification 28

4 Existing Methods **29**

4.1 In-Domain Corpus 30

4.2 Offline Multi-pass Method 31

4.3 Methods Using Directly ASR Output 31

4.4 Methods Extracting Semantic Knowledge 32
4.5 Topic-based Methods 35

5 Proposed Solution and Implementation 37
5.1 The Scheme of the Solution 37
5.2 Online ASR 39
5.3 Hypotheses Merger 40
5.4 Topic Identifier 42
5.4.1 Result Encoder 43
5.4.2 Soft Predictor 46
5.4.3 Hard Predictor 47
5.5 Topic-change Detector 47
5.6 LM Mixer 48
5.6.1 Prepared Mode 49
5.6.2 Online Mode 50
5.6.3 Offline Mode 50
5.6.4 Mix LM block 51
5.6.5 Speed of LM Mixer 53
5.6.6 Type of Storage 54

6 Experiments on TV News Domain 55
6.1 Audio Data 55
6.2 Text Data 56
6.2.1 LMs 57
6.2.2 Topics 57
6.3 Performance Measures 58
6.4 Baseline Performance 59
6.5 Supervised Text Classification 59
6.5.1 Topic Tree 59
6.5.2 Topic Clustering of Unlabeled Data 60
6.5.3 Online Topic Identification 65
6.5.3.1 Classifier and Thresholding Strategy 66
6.5.3.2 Cropping of Hypotheses 68
6.5.3.3 Weighting of Recently Recognized History 69
6.5.3.4 From Words to Lattices 70
6.5.4 Topic-change Detection 70
6.5.5 General LM weights 71
6.5.6 Adding New Topic-related N-grams 71
6.6 Unsupervised Topic Models 74
6.6.1 Adapted LMs Preparation 75
6.6.2 Number of Topics 76
6.6.3 Fine-Tuning of Parameters 77
6.7 Results Summary 77

7 Experiments on TV Sports Domain 81
7.1 Audio Data 82
7.2 Topics 83

7.3 Roster 84
7.3.1 Derived Forms of Names 85
7.3.2 Phonetic Transcriptions of Names 86
7.4 Text Data 86
7.5 Performance Measures 87
7.6 Adapted LMs Preparation 88
7.7 Online Sport Identification 91
7.8 Weights of LMs 93
7.9 Fine-Tuning of Parameters 95
7.10 Using Original Audio 96
7.11 Results Summary 97

8 Experimental Results Summary **99**
8.1 Test of Significance 100

9 Discussion **101**

10 Conclusion **103**

A Default Configuration File of Proposed System **105**

Bibliography **109**

List of Author's Publications **121**

List of Author's Other Works **123**

List of Figures

3.1 General framework for ASR system 8
3.2 Example of HMM word model 9
3.3 Example of a trellis 13
3.4 A hyperplane in SVM 20
3.5 The scheme of LSTM cell. 21
3.6 Plate notation of LDA 23
3.7 Basic framework for LM adaptation 25

5.1 The basic scheme of proposed solution 38
5.2 Example of merging two hypotheses 41
5.3 The detailed scheme of topic identifier 42
5.4 Weighting functions in the result encoder 45
5.5 The detailed scheme of topic-change detector 48
5.6 The detailed scheme of LM mixer 49
5.7 The detailed scheme of mix LM block 51
5.8 Processing time of LM mixer with increasing size of vocabulary 53
5.9 Processing time of LM mixer with increasing number of mixed LMs . . . 54

6.1 Topic cloud generated from labeled data (in Czech) 60
6.2 Example of fully expanded topic node "economy, business and finance" in our topic tree (in Czech) 61
6.3 20 top-level nodes in our topic tree (in Czech) 62
6.4 Scheme of topic-based data clustering in the news domain 63
6.5 Results with different topic clustering of unlabeled data in the news domain 64
6.6 Results with different online topic identifiers in the news domain 66
6.7 Results with different cropping of hypotheses in the news domain 68
6.8 Results with different weighting functions of recognized words in the news domain 69
6.9 Results with different steadiness of the topic-change detector in the news domain 70
6.10 Results with different weight of general LM in the news domain 71
6.11 Results with different n-grams count limitations in the news domain . . . 73
6.12 Scheme of LDA-based data clustering in the news domain 75
6.13 Results with different number of topics in the LDA model 76
6.14 Word clouds for individual topics in trained LDA model (in Czech) 78

7.1 Mapping from summer Olympics disciplines to topic clusters 84
7.2 Scheme of adapted LMs preparation in the sports domain 89
7.3 Scheme of online LM adaptation in the sports domain 91

7.4 Results with different weight of news LM in single-topic LM in the sports domain 93
7.5 Results with different weight of general LM in the sports domain 94
7.6 Results with different weighting functions for recognized words in the sports domain 95
7.7 Results with different steadiness of the topic-change detector in the sports domain 96

List of Tables

6.1 Test audio records from TV show ”Události” 56
6.2 The amount of labeled and unlabeled text data. 59
6.3 Details about LMs with different minimal n-gram counts in the news domain. 72
6.4 Results summary for the news domain 79

7.1 Test audio records from sports summary TV shows 82
7.2 Comparison of different sport identifiers 92
7.3 Results summary for experiments with shadow speaker in the sports domain 96
7.4 Results summary for experiments without shadow speaker in the sports domain 97
7.5 Results summary for the sports domain 98

8.1 The main results of the thesis 99

Abbreviations

AM	**A**coustic **M**odel
ASR	**A**utomatic **S**peech **R**ecognition
BOW	**B**ag **O**f **W**ords
CC	**C**losed **C**aptions
DNN	**D**eep **N**eural **N**etwork
G2P	**G**rapheme to **P**honeme
GMM	**G**aussian **M**ixture **M**odel
HDD	**H**ard **D**isk **D**rive
HMM	**H**idden **M**arkov **M**odel
IR	**I**nformation **R**etrieval
LDA	**L**atent **D**irichlet **A**llocation
LM	**L**anguage **M**odel
LSA	**L**atent **S**emantic **A**nalysis
LSTM	**L**ong **S**hort **T**erm **M**emory
LVCSR	**L**arge **V**ocabulary **C**ontinuous **S**peech **R**ecognition
NER	**N**ame **E**rror **R**ate
NLP	**N**atural **L**anguage **P**rocessing
NN	**N**eural **N**etwork
OOV	**O**ut-**O**f-**V**ocabulary
PNER	**P**roper **N**oun **E**rror **R**ate
RNN	**R**ecurrent **N**eural **N**etwork
RTR	**R**eal **T**ime **R**atio
SSD	**S**olid **S**tate **D**rive
SVM	**S**upport **V**ector **M**achine
TI	**T**opic **I**dentification or **T**opic **I**dentifier
WER	**W**ord **E**rror **R**ate
WFST	**W**eighted **F**inite **S**tate **T**ransducer

Chapter 1

Introduction

Automatic speech recognition (ASR) systems have become very useful in people's everyday lives. ASR system produces a digital transcript of uttered speech automatically without the need of typing words manually. The transcript can then be used to extract various information in order to analyze the speech. Thus, it can save a lot of time and work in human-to-computer interactions, which are increasingly important in present days.

Typical applications, in which ASR is indispensable, are: (1) chatbots, virtual assistants and dialogue systems, where people are conversing with computers, (2) dictating systems, where people need to note a large amount of information without typing it, (3) in-vehicle systems, where hands-busy and eyes-busy drivers need to operate onboard devices, (4) multi-media searching in large audio or video databases, where manual transcription would be unfeasible, (5) forms with voice input allowing people to fill any text field (e.g. queries to search engine or SMS content) by speaking on a mobile phone or computer, or (6) closed captioning of live videos, which helps deaf and hard-of-hearing audiences to access the speech in videos.

However, if the domain of the recognized speech does not match the domain the ASR system has been trained for, the resulted transcript is usually errorful. For example, if we take ASR system trained for medical dictating system and use it to transcribe political debates, we will most likely obtain unintelligible transcript, because politics and medicine have markedly different vocabulary and word sequence statistic. Such domain mismatch often arises when dealing with *multi-topic* speech, in which the speaker switches from one topic to another frequently.

A standard mechanism used in the literature to reduce effects of the domain mismatch in multi-topic tasks is a *language model (LM) adaptation.* The principle of LM adaptation is to use a domain-independent general language statistics in the first pass to get a base transcript of the speech. This transcript is then analyzed and a topic is predicted for each utterance. Based on predicted topic, each utterance is then recognized again in the second pass using system adapted to the particular topic.

Much more complicated situation occurs when the ASR system must transcribe a multi-topic speech *online*, i.e. in the real time. In such tasks, the transcribed words must be available at the very same time (or with reasonably small delay) as the words are uttered. The online LM adaptation is required especially in dictating systems, where the user can change the topic of his speech anytime and the system must adapt quickly, or during closed captioning of live multi-topic TV shows. For example, in the task of closed captioning of live TV news, each individual report can be about completely different topics and the captions must be shown to the viewer very soon after uttered in order to match text with the underlying video. In such online tasks, the standard two-pass approach cannot be used. Instead, the topics must be dynamically identified in just-uttered words and whenever a topic change is detected, the system must adapt as soon as possible.

This thesis deals with the problem of *online topic-based language model adaptation*. The proposed solution combines two very important research areas: real-time automatic speech recognition and automatic topic identification. The objective is to build a complex ASR system capable of real-time speech recognition and low-latency topic-based LM adaptation in order to reduce word error rates in multi-topic tasks. The originality of this work lies in minimizing latency of the adaptation by using two parallel decoders (instead of a common two-pass approach, which cannot be used online) and online merging their outcomes. The way the change of topic is detected from partial hypotheses is also innovative in this work.

1.1 Outline of the Thesis

After this introduction chapter, the main objectives of the thesis are listed in chapter 2. Then, in chapter 3, the theoretical background for approaches discussed in this thesis is summarized. The chapter covers three key research areas this thesis is building on: automatic speech recognition (section 3.1), automatic topic identification (3.2) and language model adaptation (3.3). Chapter 4 sums up the state of the art of the problem including all inspiring and relevant approaches we are aware of.

An innovative solution to extend existing online ASR system by topic-based language model adaptation is proposed and described in chapter 5. Proposed solution was tested in two challenging multi-topic tasks: online transcription of TV news (chapter 6) and online transcription of TV sports summaries (chapter 7). The most important results are tabulated in chapter 8 along with a test of their statistical significance, and discussed in chapter 9.

Finally, the last chapter 10 concludes the thesis and suggests future work.

Chapter 2

Objectives of the Thesis

The main objectives of this thesis are:

1. Survey the problem of language model adaptation with accent on online and topic-based methods.
2. Propose a solution to extend existing online ASR system by online topic-based language model adaptation. The proposed system should be able to detect a change of a topic in a real-time speech and adapt the language model online with the latency as low as possible.
3. Implement proposed solution in suitable programming language. The implementation should be fully modular allowing user to easily change component modules. Default parameters should be set to reasonable values. By changing default values, the user should be able to experiment with the system in order to fine-tune parameters for particular task.
4. Select suitable multi-topic real-time ASR problems to test the proposed system.
5. On selected tasks, use existing online ASR as a baseline system and evaluate error rates.
6. On selected tasks, use proposed ASR system and evaluate error rates.
7. Compare error rates and evaluate the improvements of the proposed system over the baseline system.

Chapter 3

Theoretical Background

This chapter provides reader with theoretical basis and relevant mathematical formulas from three key research areas this thesis is building on: automatic speech recognition (section 3.1), automatic topic identification (section 3.2) and language model adaptation (section 3.3).

3.1 Automatic Speech Recognition

The goal of *Automatic Speech Recognition* (ASR) is to automatically convert spoken language into a text using a computer. The research of ASR started with recognition of isolated words with a small vocabulary (e.g. digits) and continued to a *large-vocabulary continuous speech recognition* (LVCSR) systems, where by "large-vocabulary" researchers usually mean hundreds of thousands words and by "continuous" that words in the speech are running together naturally (Martin and Jurafsky, 2000). Nowadays, almost every modern ASR system is LVCSR, especially for highly inflected languages, where the vocabulary have to incorporate even few million words and word forms in order to avoid high OOV-rate.

In present days, ASR has many useful applications, for example in dictation systems, dialogue systems, virtual personal assistants, home assistants, multi-media archives, voice-navigated systems, in-vehicle systems, closed captioning systems and many others. ASR is especially useful in hands-busy or eyes-busy applications, where the user needs to interact with the computer, but cannot concentrate on typing (e.g. when driving a car).

Mathematically, ASR system can be defined in the following way. Given the observed acoustic signal represented as a sequence of observation vectors O, we are searching for a sequence of words W^*, which is the most probable one of all possible word sequences in a language $\mathscr{L}$:

$$W^* = \underset{W \in \mathscr{L}}{\arg\max}\, P(W|O). \tag{3.1}$$

Using a Bayes rule, we obtain

$$W^* = \underset{W \in \mathscr{L}}{\arg\max}\, \frac{P(O|W)P(W)}{P(O)}. \tag{3.2}$$

Since $P(O)$ is a constant value for all word sequences W, it can be ignored, and we can write a key formula for ASR system

$$W^* = \underset{W \in \mathscr{L}}{\arg\max}\, P(O|W)P(W). \tag{3.3}$$

$P(W)$, the prior probability of word sequence W, is modeled by a *language model* (LM) and $P(O|W)$, the probability that given the word sequence W, the observation sequence is O, is modeled by an *acoustic model* (AM).

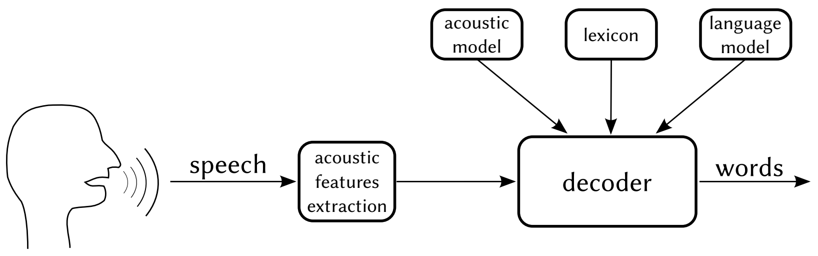

FIGURE 3.1: General framework for ASR system.

A typical framework for an ASR system is depicted in figure 3.1. The speech signal is processed by *acoustic features extraction* block, which extracts characteristic features from the observed signal. Then, the acoustic and language model together with a *lexicon* (defining the pronunciation of all words in the vocabulary) contribute to solving the equation 3.3, which is a problem usually referred to as a *decoding.* The part of ASR system dealing with the decoding problem is called a *decoder.* In following subsections, these blocks will be discussed in more detail.

3.1.1 Acoustic Features Extraction

Acoustic features extraction is a task of converting an acoustic signal into a sequence of feature vectors (observations). Typically, the input speech signal is sliced into short time-segments, called *frames.* A typical duration of one frame is 10 ms. In such a short time interval, the acoustic signal is considered to be a stationary signal, from which characteristic spectral features are extracted. The most popular features used in ASR systems are *Perceptual Linear Predictions* (PLPs) and *Mel-Frequency Cepstral Coefficients* (MFCCs). To put it simply, these features are reflecting how much energy in the signal is at different frequencies. For detailed comparison of various feature types see (Davis and Mermelstein, 1980).

It has been shown (Furui, 1986) that the performance of ASR system can be greatly enhanced by concatenating the feature vector with its time derivatives. These derived features are known as *delta* coefficients. The delta coefficients c_t^Δ of a basic (static) coefficients c_t in the time (frame) t can be computed as

$$c_t^\Delta = \frac{c_{t+\Delta} - c_{t-\Delta}}{2\Delta}, \tag{3.4}$$

where Δ is the size of the frame window which is usually set to 1. Also, the second-order derivatives known as *delta-delta* or *acceleration* coefficients can be calculated and concatenated with the feature vector in the same way, but they are calculated from the deltas, not from the static coefficients.

3.1.2 Acoustic Model

The *acoustic model* (AM) models the relationship between the sequence of observations (parameterized audio signal) and the basic phonetic units in the language (e.g. phones, triphones etc.). The most common way how to do that is using *hidden Markov models* (HMMs).

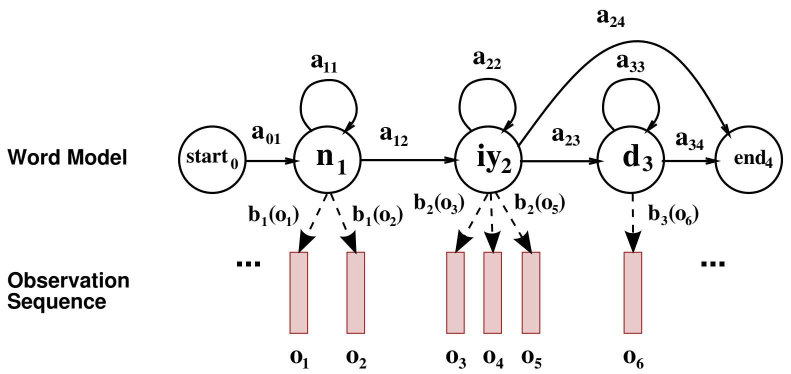

FIGURE 3.2: An example of a HMM word model generating the observation sequence $o_1, o_2, ..., o_6$ taken from (Martin and Jurafsky, 2000). The underlying word is "need" consisting of phonemes n, iy, d. Transition probability from state i to state j is denoted as a_{ij} and observation likelihood of k-th observation vector being generated from a state i is denoted as $b_i(o_k)$. Note that self-loops a_{ii} are here to model variable phone duration. For instance, phoneme iy is spread across 3 frames in the audio signal and phoneme d fills in only one frame.

HMM, as depicted in figure 3.2, is basically a stochastic automaton with a stochastic output process attached to each state. HMM is defined by a set of possible states, transition probabilities between these states and observation likelihoods expressing the probability of an observation being generated from a state. The term "hidden" refers to a fact that we can observe only the output of the process, not the states.

In HMM used in ASR systems, each possible phonetic unit in the language is represented by one or more states, and within each state, a probability distribution over the observation space is stored. To describe the probabilities of observations for each state, one can employ *Gaussian mixture models* (GMMs) or, recently much more popular, *Deep Neural Networks* (DNNs) (Hinton et al., 2012). Parameters of AM have to be estimated from (large enough) annotated audio corpus.

Since this work is focused mainly on the language models, we will not discuss AMs any deeper here. A detailed overview about acoustic processing of a speech and HMMs in ASR can be found e.g. in (Rabiner, 1989; Martin and Jurafsky, 2000).

3.1.3 Language Model

The goal of a *language model* (LM) is to estimate a probability of any sequence of words W, i.e. $P(W) = P(w_1, w_2, ..., w_N)$, in the language. The most popular solutions in present days are statistical *n-gram LMs*, which are based on a statistical evidence of word sequences in the training corpus, and LMs based on recurrent neural networks, *RNN LMs* (Mikolov et al., 2010). Since the adaptation of *RNN LMs* is still an open problem due to a high computational complexity of training and updating (Chen et al., 2015; Deena et al., 2019), in the rest of the thesis, we will focus only on statistical n-gram LMs.

Statistical LM is based on a decomposition of a joint probability of a contiguous sequence of N words $P(W) = P(w_1, w_2, ..., w_N)$ into a product of conditional probabilities. The decomposition is performed via the statistical chain rule

$$\begin{aligned} P(w_1, w_2, ..., w_N) &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_N|w_1, w_2, ..., w_{N-1}) \\ &= \prod_{i=1}^{N} P(w_i|w_1, ..., w_{i-1}), \end{aligned} \tag{3.5}$$

where each component $P(w_i|w_1, ..., w_{i-1})$ is a probability of seeing the word w_i given its history $w_1, ..., w_{i-1}$.

Since the number of possible word sequences W (and thus the number of parameters to be estimated) is immense, there will never be enough training data for estimating all probabilities (components) in 3.5. Instead, in n-gram LMs, each probability is approximated to the probability conditioned with a shorter word sequence. Taking only k words from the history into account, the probability of seeing the next word w_i can be approximated as

$$P(w_i|w_1, ..., w_{i-1}) \approx P(w_i|w_{i-k}, ..., w_{i-1}). \tag{3.6}$$

Using such approximation, the formula for n-gram LM can be rewritten as

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

where the probability of each approximated component can be estimated from a training corpus as

$$P(w_i|w_{i-k}, ..., w_{i-1}) = \frac{c(w_{i-k}, ..., w_{i-1}, w_i)}{c(w_{i-k}, ..., w_{i-1})}, \tag{3.8}$$

where $c(W)$ denotes observed frequency (count) of a word sequence (n-gram) W in the corpus.

The simplest n-gram LM is *unigram* (1-gram), where no history of words is taken into account, i.e. $P(W) = \prod_i P(w_i)$. However, in practice, usually *bigram* (2-gram) LM

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-1}) \tag{3.9}$$

or *trigram* (3-gram) LM

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-2}, w_{i-1}) \tag{3.10}$$

is typically used.

Since the training corpus is usually sparse, it can be expected that in testing data, there will be n-grams never seen in the training data. Such unseen n-gram would obtain zero probability from the LM which would lead to a zero probability of whole word sequence 3.7. To deal with this *zero-frequency problem*, some additional parameters are usually computed in order to *smooth* the LM:

- Some probability mass is taken from observed n-grams (so-called *discounting*) and it is assigned for all unseen n-grams to avoid the zero-frequency problem.
- Lower-order n-grams observed in the training data are also incorporated into the LM. When calculating the probability of n-gram, also information from lower-order models can be taken into account. There are two main ways how to combine information from higher- and lower-order models: *linear interpolation* (Jelinek and Mercer, 1980) and *back-off* model (Katz, 1987), which is "backing-off" to

lower-order model whenever there is not enough evidence about the n-gram in the higher-order model.

The most popular smoothing techniques in present days are *Good–Turing discounting* and *modified Kneser-Ney smoothing*. The detailed description of them along with many other smoothing techniques can be found e.g. in (Chen and Goodman, 1999).

A drawback in using n-gram LMs is that they are not able to capture long-range dependencies. This drawback can be slightly suppressed by using *maximum entropy LM*, which can combine more knowledge sources (Wu and Khudanpur, 2002), or by using some topic-based adaptation technique discussed later in chapter 3.3.

3.1.4 Lexicon

Lexicon, also referred to as *pronunciation dictionary*, is a connector between words from the LM and phonetic units from the AM. The pronunciation is usually defined as a phonetic transcription of each word, i.e. as a sequence of phonemes that are expected to be observed when correctly pronouncing the word.

Obtaining the lexicon with correct pronunciations is not a trivial task. Phonetic transcriptions can be obtained either manually by a human expert leading to accurate, but expensive and time-consuming solution, or automatically which is cheap and fast, but the transcription is usually errorful when transcribing words with irregular pronunciation.

Building an automatic phonetic transcriber is a challenging task. Many language-specific *rule-based* automatic phonetic transcribers have been built. However, most of them cannot correctly transcribe words with irregular pronunciation, words adopted from foreign languages etc.

Some attempts to train a stochastic so-called *grapheme-to-phoneme* (G2P) model from an existing lexicon in order to estimate a phonetic transcription for arbitrary word have been reported. The most popular stochastic G2P model is *Phonetisaurus* (Novak et al., 2012). However, not even G2P model is usually able to cover rare language-specific pronunciation irregularities, since it gives priority to the most frequent pronunciation seen in the training lexicon. The author of this thesis showed in (Lehečka and Švec, 2013), that enriching the LM with inclusions from other language (English) and updating the lexicon automatically using G2P might lead to more accurate speech recognition. (Rao et al., 2015) showed, that G2P model based on sequential learning of recurrent neural network outperforms stochastic models and that best results are obtained when combining both approaches.

3.1.5 Decoder

Finally, in the decoding stage, the acoustic model, language model and the lexicon are used to build a recognition network and to find the sequence of words (i.e. a hypothesis), which has the highest probability given the acoustic signal, i.e. to solve equation 3.3.

Since the number of possible sentences in the language is immense, an exhaustive computation of all of them and selecting the best one is not the right way to go. Instead, the space of all possible sentences should be searched more efficiently disregarding unlikely hypotheses. Typically, the searched space is represented as a trellis, which is an oriented graph, where the nodes (states) are organized vertically into connected time-frames, as depicted in figure 3.3. In the speech recognition tasks, each time-frame of the trellis is moreover associated with one observation (parameterized frame of audio signal).

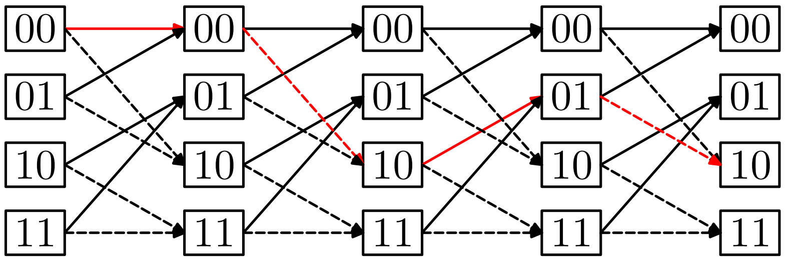

FIGURE 3.3: An example of a trellis downloaded from `http://en.wikipedia.org/wiki/Trellis_(graph)`. At each time-frame, which is formed of vertically organized set of all possible states, each state can be associated with any state from the previous time-frame and the next transition can lead to any state from the future time-frame. In this figure, there are four possible states (00,01,10,11) and five time-frames. When each transition (arrow) is associated with some transition probability (weight), the most likely sequence of states can be found as a path with the highest join probability of transitions along the path through the trellis. Note that this example shows only the simple trellis, but in ASR task, nodes (states) are additionally associated with an observation likelihoods expressing the probability of time-synchronized observation being generated from a state.

The most popular decoding algorithm is *Viterbi algorithm*. It uses forward dynamic programming to find the most likely sequence of states in a trellis, that results in a sequence of observations. It exploits the Markov property[1] and recursively performs an efficient search while remembering the backpointers in order to backtrack the most likely sequence of states after passing through the whole trellis.

When using simple Viterbi algorithm, the output of the decoder is one most likely sequence of words. However, when ASR is a part of some more complex system, usually more informative output is required. Following outputs are used frequently in the literature:

[1]The future state depends only upon the present state, not on the history.

- *N-Best List* — Viterbi algorithm can be extended to find not one but N-best paths (hypotheses) in the trellis.
- *Word Lattice* — a labeled, weighted, directed acyclic graph in which each complete path represents an alternative hypothesis, weighted by its recognition score for a given utterance. In a *word lattice*, each arc represents a word associated with a score from decoder and each state (node) represents a point in time. The basic idea behind lattices is that they can represent a large number of alternative hypotheses in more compact way than N-best list can.
- *Sausages* — a compact representation of a word lattice, which can be seen as a word confusion network (Mangu et al., 1999).

3.1.6 Implementation

The mostly used paradigms to implement an ASR system require a combination of some HMM toolkit (e.g. HTK (Young and Young, 1993)), LM toolkit (e.g. SRILM (Stolcke et al., 2002)) and a special file containing the lexicon. However, in present days, there is a significant trend in using also alternative implementations, which are usually based on one unifying framework capable of representing all the different knowledge sources needed for ASR system. The most popular of them are presented in following paragraphs.

WFST

An ASR system implemented as a *Weighted Finite State Transducer* (WFST) offers unified framework to efficiently represent major components of speech recognition systems, including HMMs, context-dependency models, pronunciation dictionaries, statistical grammars, and word or phone lattice (Mohri et al., 2008). Speech recognition with WFST is used e.g. in Kaldi speech recognition toolkit (Povey et al., 2011).

The typical way of using WFST-based ASR system is as a static recognizer. The main advantage of using static WFST is that the graphs can be heavily optimized (i.e. determinized and minimized) before the recognition starts, therefore at the decoding time, more compact recognition network is to be searched.

Neural Networks

With increasing computational power of today's computers and GPUs, training neural networks (NNs) capable of learning complex problems related to ASR system has become popular.

The problem of representing an acoustic model with NN or building a hybrid model combining HMM with NNs has been studied mostly in 80' and 90'. Proposed methods used mainly time-delay and recurrent NNs. The survey of the most important methods can be found in (Trentin and Gori, 2001). Most of them were designed to replace only one part of the speech recognition pipeline with NNs. In recent years, a significant improvement in acoustic modeling has been reported e.g. in (Hinton et al., 2012), where GMMs were replaced with deep feed-forward NNs. The problem of implementing a language model with NN has been studied intensively in recent years. The most promising method has been proposed in (Mikolov et al., 2010).

However, the incorporation of all ASR components into one global end-to-end NN-based ASR system, which would take the acoustic signal on the input while outputting a sequence of decoded words, is still a challenging task (Graves and Jaitly, 2014). Recently, there has been reported several promising approaches, such as *Deep Speech* (Hannun et al., 2014), *Listen, Attend and Spell* (Chan et al., 2016) or a multi-head attention architecture with sequence-to-sequence models (Chiu et al., 2017).

3.1.7 Evaluation

To measure how well the ASR system performs, the common approach is to count differences between the reference text (i.e. what was the true sequence of words) and the recognized sequence of words. The most popular performance measure is a *Word Error Rate* (WER) or its complement to 100 % known as a *Word Accuracy* (Acc).

Since the two examined sequences of words (reference and ASR output) can have different number of words, *Levenshtein alignment* is usually used to align them. Levenshtein alignment is based on *Levenshtein distance*, which is the minimum number of operations needed to convert one sequence to another. Possible operations are insertion, deletion and substitution. If we denote the total number of insertions as I, deletions as D, substitutions as S and the total number of words in the reference transcript as N, then we can define WER as

$$WER = \frac{I + D + S}{N} \cdot 100\,\% \tag{3.11}$$

and

$$Acc = 100 - WER = \frac{N - I - D - S}{N} \cdot 100\,\%. \tag{3.12}$$

Since there can be more operations needed than words in the reference transcript, WER can be higher than 100 % and Acc can be negative. For instance, if the reference transcript is "hello word" and the ASR output is "hi our low word", there are 3 operations needed to convert the reference into the ASR output (substitute "hello" with "hi" and insert "our" and "low", thus $I = 2$, $D = 0$, $S = 1$ and $N = 2$), which leads to $WER = 150\,\%$ and $Acc = -50\,\%$. To overcome this problem, a *Correctness* ($Corr$) can be measured as

$$Corr = \frac{N - D - S}{N} \cdot 100\,\%, \tag{3.13}$$

i.e. all insertions are ignored, which keeps the value within the interval $[0, 100]$.

3.2 Topic Identification

The goal of *Topic Identification* (TI) can be defined as follows: given an unstructured text corpus, automatically associate each document in the corpus with a relevant label(s) identifying the topic(s) of the document. The set of target topics depends on the particular domain and objective.

First, let us clarify the terminology. In the literature, several terms for the task of automatic document labeling can be found, namely *text classification*, *text categorization*, *topic identification*, *topic detection*, and *topic spotting*, all of them meaning roughly the same. According to the survey on text categorization (Sebastiani, 2002), terms *text categorization*, *text classification* and rather historical term *topic spotting* share the same definition of "labeling natural language texts with thematic categories from a predefined set". However, this definition excludes unsupervised text clustering, which is very popular document-organizing method, in which the set of categories is not known in advance (see section 3.2.4). On the other hand, term *topic detection* usually stands for a task of detecting, whether some new topic (event) is present in a stream of text documents (e.g. tweets (Cataldi et al., 2010) or broadcasting news (Allan, 2012)), or if the arriving document falls into some existing topic. The term *topic identification*, in my opinion, fits best to the definition of the task of associating each document with a relevant topic-label while covering all important approaches (including unsupervised text clustering), which is the reason this term will be used to address the task in this thesis.

The TI task is closely related to several other research areas, mainly *information retrieval* (IR), *machine learning*, *natural language processing* (NLP), *information extraction* and *text data mining.* Note that borders between these disciplines are not always exactly delimited. TI makes use of NLP techniques and *information extraction* methods to preprocess the text and convert each document to a vector of relevant features, *machine learning* algorithms are often used to build a classifier assigning topics for unlabeled documents, and the whole process of TI can be viewed as an instance of both *text data mining*, which aims to discover or derive new information by analyzing large quantities of text, and IR, which is usually defined as the process of finding documents that satisfy user's information needs (asked in the form of queries) (Hearst, 1999).

Depending on the data, the TI task can be divided into several cases:

- **Supervised text classification** requires training data with correct label(s) assigned to each text document. Usually, one of popular classifiers from the machine learning area is adopted. See subsection 3.2.3 for details.
- **Unsupervised topic modeling** does not require correct labels, it aims to uncover hidden (latent) semantic relationships between words and documents and thus reveal hidden "topics" or substantial keywords of documents. See subsection 3.2.4 for details.
- **Semi-supervised learning** requires a part of training data to be correctly labeled and the rest unlabeled. Since semi-supervised learning usually extends some of the methods mentioned above to fit appropriately to some particular data, it will not be discussed in this work. An overview about existing semi-supervised learning methods can be found e.g. in (Zhu, 2005).

Depending on the application, the TI task can be of two cases: (1) *single-label* TI, in which exactly one label must be associated with each document, and (2) *multi-label* TI (see section 3.2.5), in which any number of topics can be present in each document.

3.2.1 Text Preprocessing

First of all, some text modifications are usually done in each document to obtain the text cleaner, the vocabulary smaller or the corpus more suitable according to the problem to be solved. These modifications are called *text preprocessing*. Usually, some of the following text preprocessing methods are employed:

- **cleaning** — remove boilerplate, comments, tags etc.,
- **tokenization** — convert text into a sequence of tokens[2],
- **true-casing** — determine the correct capitalization of each word,

[2]Tokens are usually words, but can be also phrases, symbols, syllables, or other language elements.

- **word normalization** — convert inflected forms of words into stems[3] (*stemming*) or into lemmas[4] (*lemmatization*),
- **text normalization** — convert written forms of abbreviations, numbers, dates, acronyms etc. into their spoken form, i.e. into a sequence of words,
- **stop words removal** — remove undesired or topic-neutral words.

3.2.2 Document Representation

When dealing with a text, direct use of text representation (i.e. a sequence of tokens) in TI task is impractical, because of a variable length of each document. Therefore, it is desirable to convert raw text data to a so-called *vector space*, where each word or document in a corpus is represented as a vector of predefined features.

Inspired by debates whether the knowledge in the human brain is stored in some specific regions or it is rather spread across the entire cortex, document representations can be divided into two groups:

(1) **local representation** — only several features encode the document while the majority of features is not activated, i.e. the representation is sparse,

(2) **distributed representation** — each feature contributes to the representation, i.e. the representation is dense (Hinton et al., 1986).

Typical example of (1) is *tf-idf*[5], which is a *bag-of-words* (BOW) model converting the corpus into a document-term matrix, or *1-of-V* (also known as *one-hot*) encoding, where each word is represented as a vector full of zeros with 1 only on the position of the word in the vocabulary of size V. A typical examples of (2) are *autoencoders* (Hinton and Salakhutdinov, 2006), *word2vec* (Mikolov et al., 2013) and its subword extension *fastText* (Bojanowski et al., 2016), *GloVe* (Pennington et al., 2014) or paragraph vectors (Le and Mikolov, 2014). Moreover, topic models discussed later in subsection 3.2.4 can be also used to generate distributed representations of documents.

Recently, it has become very popular to use word-based representations (such as *1-of-V*, *word2vec* or *GloVe*) as inputs to TI models with sequential learning (e.g. recurrent neural networks (Dai and Le, 2015; Johnson and Zhang, 2016)), or to view the sequence of word vectors as an "image" and adopt popular image-processing models, such as convolutional neural networks (Johnson and Zhang, 2015; Conneau et al., 2016).

[3]Stem is a common part of all inflected variants of the word.
[4]Lemma is a base lexical form of the word.
[5]`https://en.wikipedia.org/wiki/Tf%E2%80%93idf`

3.2.3 Text Classification

The *Text Classification* (TC) task aims to label natural language texts with thematic categories from a predefined set (Sebastiani, 2002). This is usually done by a classifier trained from some amount of data with correct labels, i.e. supervised learning is used to set classifier parameters.

Because of the vast number of papers about TC task published in recent decades, it is almost impossible to discuss all different algorithms in this short survey. Therefore, we will mention only several most popular classifiers and for those used in this thesis, we will also provide detailed mathematical background.

To solve a TC problem, one can employ for example *decision trees* (a hierarchically organized set of rules which are used to decide about the topic of a document) such as *C4.5 algorithm* (Quinlan, 2014), *probabilistic classifiers* (models that aim to estimate the probability that a document belongs to a topic) such as *naive Bayes classifier* (Zhang, 2004) or *proximity-based classifiers* (models assuming documents from the same topic are likely to be close to each other in the vector space) such as *k-nearest neighbor classifier*[6]. Another two popular classifiers, which are used in this thesis, are described in following paragraphs.

SVM

A *Support Vector Machine* (SVM) classifier (Joachims, 1998) aims to divide the space of input vectors by a hyperplane which separates the positive examples from the negative ones with the widest possible margin, as depicted on figure 3.4. Training vectors most difficult to classify, i.e. vectors lying on the very edge of the margin, are called *support vectors.*

Given a training dataset with n samples $(x_1, y_1), ..., (x_n, y_n)$, where $y_i \in \{1, -1\}$ denotes the class membership of the sample x_i (positive of negative class), SVM classifier solves the following primal problem: minimize

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\zeta_i \tag{3.14}$$

subject to $y_i(w \cdot x_i - b) \geq 1 - \zeta_i$ and $\zeta_i \geq 0, i = 1, ..., n$, where the hyperplane is defined by its normal vector w and bias b $(w \cdot x - b = 0)$, ζ_i is hinge loss function, i.e. $\zeta_i = \max(0, 1 - y_i(w \cdot x_i - b))$ and $C > 0$ is the penalty parameter of the error term.

[6] `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`

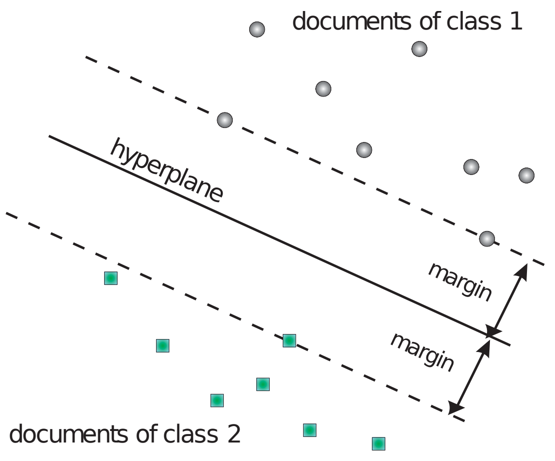

FIGURE 3.4: A hyperplane in SVM taken from (Hotho et al., 2005). The hyperplane is dividing documents in the vector space into two classes. The hyperplane has the widest possible margin.

Its dual problem is to minimize

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i\alpha_i K(x_i, x_j) y_j \alpha_j - \sum_{i=1}^{n}\alpha_i \tag{3.15}$$

subject to $\sum_{i=1}^{n}\alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C, i = 1, ..., n$, where $\alpha_1, ..., \alpha_n$ (αs for short) are dual variables and K is a kernel function. Most of the αs will turn out to have the value zero. The non-zero αs will correspond to the support vectors. Knowing the αs, we can compute $w = \sum_{i=1}^{n}\alpha_i y_i x_i$, i.e. w is a linear combination of support vectors. Selecting some support vector x_i, the bias can be computed as $b = \sum_{k=1}^{n}\alpha_k y_k K(x_k, x_i) - y_i$.

Finally, the decision about some new sample x is made by computing signum of the decision function

$$y = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i\alpha_i K(x_i, x) - b\right). \tag{3.16}$$

Note that most of αs will be zero, and so the decision is made using only several support vectors. When classifying text documents, usually the linear kernel $K = (x_i \cdot x_j)$ is used.

SVM classifier is designed to solve a dichotomy[7] problem. If there are more than two classes to be distinguished among, usually *one vs. the rest* strategy is employed. It trains one SVM classifier per topic. During the training phase, documents belonging to the topic are presented as positive examples while all other documents are considered to be negative examples. During the classification, the topic with the highest decision function should be assigned.

[7]Classification into two classes: positive and negative.

Multi-class SVM produces an uncalibrated soft prediction (decision function output) that is not a probability. Based on (Niculescu-Mizil and Caruana, 2005), better result can be achieved when calibrating predictions in order to obtain a posterior probability distribution over classes. In this work, we used sigmoid method described in (Platt et al., 1999) to calibrate SVM's decision function outputs.

LSTM

Long Short Term Memory (LSTM) classifier is a member of RNN classifiers family designed to avoid the vanishing and exploding gradient problems[8]. Each LSTM cell in the network consists of four neural network layers connected as depicted in the figure 3.5.

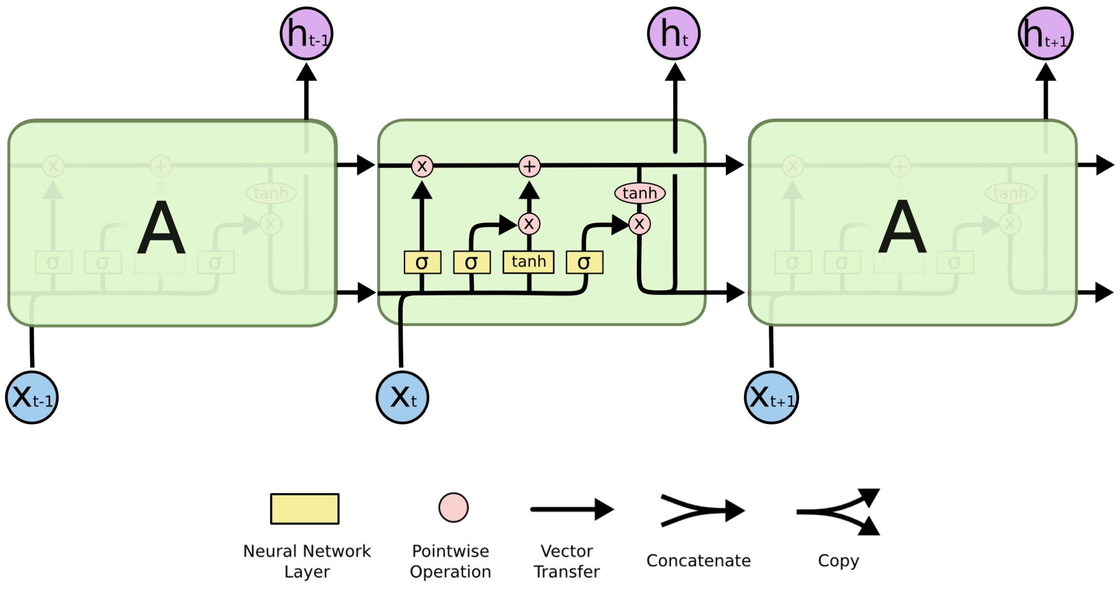

FIGURE 3.5: The scheme of LSTM cell, adopted from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Mathematically, LSTM cell can be defined as follows. Given the cell output h_{t-1} from the previous time step and current input vector x_t, the *forget gate* layer, parameterized by its weight matrix W_f and bias vector b_f, is

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{3.17}$$

[8]The exploding gradients problem refers to the large increase in the norm of the gradient during training. Such events are due to the explosion of the long term components, which can grow exponentially more than short term ones. The vanishing gradients problem refers to the opposite behavior, when long term components go exponentially fast to norm 0, making it impossible for the model to learn correlation between temporally distant events. (Pascanu et al., 2013)

where σ is a sigmoid activation function and $[\,.\,,\,.\,]$ means concatenation of two vectors. This gate decides what information to throw away from the cell state. Analogously, the *input gate* layer

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.18}$$

decides which values in the cell state to update with new candidate values

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{3.19}$$

Now, the cell state C_{t-1} from the previous time step can be updated by computing

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{3.20}$$

where $*$ is element-wise matrix multiplication. Having a new cell state, the *output gate* layer

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.21}$$

decides what information will be outputted from the cell state, and finally, the new output is computed as

$$h_t = o_t * \tanh(C_t). \tag{3.22}$$

In this way, LSTM network itself decides which information to store and which information to forget. The sigmoid function in each gate ensures each output is between zero and one, describing how much of each element should be let through the gate.

In text classification tasks, the NN architecture usually consists of several layers of LSTM cells followed by some flattening layer (e.g. pooling over time or connecting only the last output from the last LSTM cells) followed by several fully connected feedforward layers, where the last one is a sigmoid layer with the same number of neurons as the number of target classes. The sigmoid activation function at the end of the network maps logits[9] into interval between zero and one, encoding how much the input sequence belongs to each class.

LSTMs have been successfully used to solve text classification problems e.g. in (Dai and Le, 2015) or (Johnson and Zhang, 2016).

3.2.4 Topic Models

Unsupervised TI task aims to organize a set of documents according to discovered topics without any supervisor information. Common statistical models solving such document

[9] The vector of raw (non-normalized) predictions that a classification model generates.

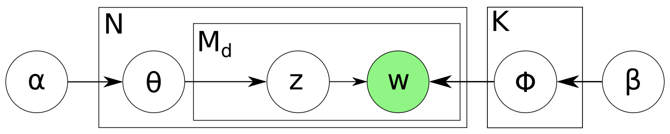

FIGURE 3.6: Plate notation of LDA model. In this model, word w is the only observable variable, N is number of documents, M_d is number of words in document d, K is number of latent topics and a value in the upper left corner of each rectangle means repetition count of its content.

clustering problem are called *topic models*. They aim to uncover hidden semantic relationships between words and documents in a corpus and thus reveal hidden topics or substantial keywords of documents.

In the last three decades, there has been published a lot of studies dealing with unsupervised topic modeling. The earlier approaches used mainly *Latent Semantic Analysis* (LSA) (Deerwester et al., 1990) or *probabilistic LSA* (pLSA) (Hofmann, 1999). However, the leading paradigm in unsupervised topic modeling during the last 15 years is *Latent Dirichlet Allocation* (LDA) (Blei et al., 2003).

In LDA, documents are random mixtures over latent topics generated from Dirichlet distribution with parameter $\alpha = (\alpha_1, \alpha_2, ..., \alpha_K)$ where K is the number of latent topics, and latent topics are random mixtures over words generated from Dirichlet distribution with parameter $\beta = (\beta_1, \beta_2, ..., \beta_V)$, where V is the size of corpus vocabulary.

The plate notation of LDA is shown in figure 3.6. Each document is represented by a topic distribution $\theta \sim \text{Dirichlet}_K(\alpha)$, while each topic is represented by a word distribution $\Phi \sim \text{Dirichlet}_V(\beta)$. For each word position in each document, a latent topic z is drawn from θ, corresponding word distribution Φ_z is found in Φ and word w is drawn from Φ_z. In order to predict the topic distribution for an unseen document, we are using online variational inference (Hoffman et al., 2010) in this work.

Since topic models are searching for hidden relationships without any knowledge about the corpus, a topic is usually represented as a probability distribution over vocabulary with no human-understandable label. A very illustrative visualization of latent topics are for example word clouds[10]. Also, comparing more topic models is difficult, because without supervisor's information and without human-understandable labels, it is not clear which topic is correct for a document (Wallach et al., 2009; Lau et al., 2014).

[10]An electronic image that shows words of different sizes according to how often they are used in the text.

3.2.5 Multi-label TI

In the case of *multi-label* TI, any number of topics can be present in each document. Based on the predicted vector of per-topic scores $s = (s_1, s_2, ..., s_K)$ called in the literature *categorization status values* or *soft prediction*, the classifier must decide, which topics are relevant for the document and which are not. In the literature, this decision is usually called *thresholding strategy* and the result of the decision is a vector of per-topic binary values (assign the topic or not), called *hard prediction.*

The first published thresholding strategies were very simple, e.g. assign only topics with probability higher than one fixed threshold (*probability thresholding*) or assign only k best topics (*rank-based thresholding*, *RCut*). The first paper addressing the thresholding strategy as a non-trivial problem, was (Lewis, 1992), where proportion-based thresholding *PCut* had been introduced. In *PCut*, each topic is assigned to its top scoring documents in proportion to the frequency of the topic in the training corpus. The drawback of this strategy is that it expects all the testing data to be fully available at the prediction time.

(Yang, 1999) presented more complex thresholding strategy *SCut*, where an optimal (in terms of F_1 measure) per-topic thresholds had been trained using validation data. The results in the paper indicated that *SCut* outperforms both *RCut* and *PCut*, which have been confirmed later by (Montejo-Ráez and Ureña-López, 2006). A deep analysis of *SCut* thresholding behavior in different tasks can be found e.g. in (Pillai et al., 2013).

In (Largeron et al., 2012), a thresholding strategy *MCut* (*Maximum Cut*), where thresholds are estimated automatically without training, was presented. For each tested document, let us denote sorted list of scores s as $S = (s(l), l = 1, ..., k)$. The highest difference between successive scores is at position

$$t = \text{argmax}\{(s(l) - s(l+1)), l = 1, ..., k-1\} \tag{3.23}$$

and the threshold is set to

$$MCut = \frac{s(t) + s(t+1)}{2}. \tag{3.24}$$

Experiments in the paper did not confirm that *MCut* performs better than other thresholding strategies, however roughly similar results were obtained in completely automatic way without any learning phase.

In *WISE 2014 Challenge* (Tsoumakas et al., 2014), many interesting thresholding strategies have been tested. The strategy of the first two winning teams was simple, yet effective: assign i-th topic, if

$$\frac{s_i}{\max(s_1, s_2, ..., s_K)} \geq p \tag{3.25}$$

with threshold p optimized on validation data.

In this section, only the most popular thresholding strategies were presented, although many other threshold selection and optimizing algorithms exist and can be found in the literature, e.g. in (Draszawka and Szymański, 2013).

3.3 LM Adaptation

The core idea behind LM adaptation is to use a small amount of domain-specific data to adjust the general LM and thus, reduce the impact of linguistic differences between the training and testing data (Chen et al., 2004). After LM adaptation, the LM should be more suitable to recognize a domain-specific speech, but not overfitted to this domain.

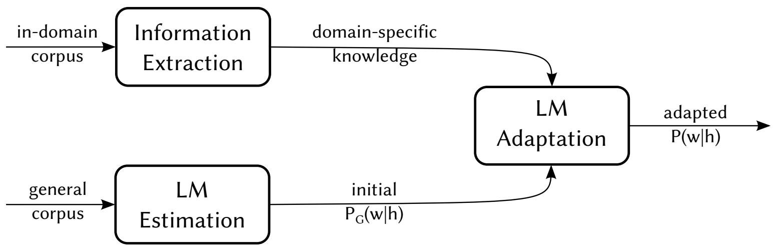

FIGURE 3.7: Basic framework for LM adaptation (Bellegarda, 2004).

Typically, LM adaptation approaches follow a basic LM adaptation framework depicted in figure 3.7, where one large general corpus is used to model the domain-independent language and another in-domain corpus is used to adapt the LM. Existing LM adaptation approaches differ mainly in what the domain-specific knowledge extracted from in-domain corpus looks like and how to combine this information with the general LM during the LM adaptation.

Based on the time, when the adaptation is applied on the general LM, two modes of LM adaptation can be employed:

- *offline LM adaptation* can be used in ASR systems, where the real-time performance is not required and the recognition can wait until the adapted LM is ready; the most popular offline method is multi-pass speech recognition (see section 4.2),

- *online (on-the-fly, dynamic, just-in-time) LM adaptation* is applied dynamically during the recognition; this mode should be used when the system must run in the real time and the topic of the speech is highly changeable.

More challenging is, of course, the online mode, since tasks such as knowledge extraction, in-domain data selection or LM interpolation have to be usually done simultaneously with the recognition as fast as possible. This thesis is focused on the online mode.

In the rest of this section, commonly-used techniques of how to combine in-domain corpus with general LM will be presented along with its mathematical background.

3.3.1 Linear Interpolation

Linear interpolation is the most common LM adaptation technique. It first trains domain-specific LM $P_D(w|h)$ from the in-domain corpus and then interpolates it with the general LM $P_G(w|h)$ at the n-gram level:

$$P(w|h) = \lambda P_G(w|h) + (1 - \lambda)P_D(w|h), \tag{3.26}$$

where $\lambda \in [0, 1]$ is an interpolation coefficient and h is history of word w. λ can be set empirically or more often, it is estimated using Expectation-Maximization (EM) algorithm to maximize the likelihood (or minimize the perplexity) of some held-out text data (Schwarm et al., 2004; Martins et al., 2007; Tur and Stolcke, 2007).

Linear interpolation technique can be used also in tasks, where there is a set of K domain-specific LMs, $P_{D_1}(w|h), ..., P_{D_K}(w|h)$, which we want to combine in order to compute the adapted LM:

$$P(w|h) = \sum_{k=1}^{K} \lambda_k P_{D_k}(w|h), \tag{3.27}$$

where $\sum_{i=1}^{K} \lambda_i = 1$ and λs are mixture coefficients, which can be set e.g. according to scores obtained from a topic identifier or by using EM algorithm to get optimal weights for so-far recognized words (Clarkson and Robinson, 1997). These adapted LMs are typically referred to as *mixture models*.

3.3.2 MAP Adaptation

When the interpolation is not done at the n-gram level, but earlier, at the corpus level, i.e. a weighted mixture of raw n-gram counts is used to train adapted LM, the adaptation approach is known as *maximum a posteriori* (MAP) adaptation.

Let $C_G(h, w)$ and $C_D(h, w)$ be a raw count of n-gram consisting of history h followed by word w in the general and in-domain corpus, respectively, then the mixture count is

$$C(h, w) = \lambda C_D(h, w) + C_G(h, w) \tag{3.28}$$

and the adapted LM is estimated as

$$P(w|h) = \frac{C(h, w)}{\lambda C_D(h) + C_G(h)}, \tag{3.29}$$

where λ is a mixture coefficient which can be estimated empirically (Bellegarda, 2004). Note that when λ is not an integer, after this count merging method, the counts are not necessarily integers and hence any smoothing algorithm that relies on counts of counts (such as Kneser-Ney or Good-Turing) does not apply (Wang and Stolcke, 2007).

A comparison experiment (Chen et al., 2004) indicated, that using MAP adaptation can bring modest reduction of perplexity, but no improvement in terms of WER in an ASR system.

3.3.3 Language Model Fill-Up

Language model fill-up adaptation strategy, proposed in (Besling and Meier, 1995), is an alternative to the interpolation strategy. It is not combining probabilities from both models as the interpolation does, but it is rather "filling up" the sparseness of the in-domain corpus using the general LM:

$$P(w|h) = \begin{cases} P_D(w|h) & \text{if } C_D(h, w) \geq T; \\ \beta P_G(w|h) & \text{otherwise,} \end{cases} \tag{3.30}$$

where T is an empirical threshold, and the back-off coefficient β is calculated to ensure that $P(w|h)$ is a true probability (Bellegarda, 2004).

This strategy prefers in-domain LM probabilities for n-grams that were seen frequently in the in-domain corpus while falling back to general LM probabilities for the rest of n-grams. This adaptation strategy can be used especially when the adaptation model has been obtained on insufficient amounts of training material (Besling and Meier, 1995; Klakow, 2006).

3.3.4 Constraint Specification

Constraint-based approaches are using extracted knowledge from in-domain corpus to compute constraints the adapted LM must satisfy. The most popular constraint-based approach is known as *Minimum Discrimination Information* (MDI) adaptation. The goal of MDI adaptation is to find out a new LM whose probability distribution satisfies some constraints derived from a specific task and whose relative entropy with the baseline LM is minimal (Lecorvé et al., 2009).

In MDI, K constraints on the adapted joint probability distribution are defined as

$$\sum_{\{(h,w)\}} I_k(h,w)P(h,w) = \alpha_D(\hat{h}_k, \hat{w}_k), \tag{3.31}$$

where I_k is the indicator function selecting an appropriate subset of n-grams and $\alpha_D(\hat{h}_k, \hat{w}_k)$ denotes the relevant empirical marginal probability estimated from the in-domain corpus (Bellegarda, 2004). Also, its minimum Kullback-Leibler distance from the joint probability distribution of the general LM is required.

It can be shown (Darroch and Ratcliff, 1972), that in MDI adaptation, we are searching for a joint probability distribution belonging to the exponential family

$$P(h,w) = \frac{P_G(h,w)}{Z(h,w)} \prod_{k=1}^{K} \exp\{\lambda_k I_k(h,w)\}, \tag{3.32}$$

where $Z(h,w)$ is a normalization factor, $P_G(h,w)$ is the joint probability distribution in the general LM and λ_k is MDI parameter. Since the solution is searched in the family of exponential distributions, LMs adapted with MDI are often referred to as *exponential models.*

Chapter 4

Existing Methods

In this chapter, existing methods dealing with language model adaptation, which are relevant to this thesis, are briefly summarized. In the literature, LM adaptation has been successfully employed in three main multi-topic speech recognition tasks:

(1) *offline recognition* — the whole audio record can be processed in multiple passes using progressively more and more domain-specific LMs,

(2) *online recognition with LM adaptation controlled by user* — the online ASR system adapts the LM on demand, i.e. the user decides about change of topic in the speech and triggers the LM adaptation,

(3) *online recognition with automatic LM adaptation* — fully automated online ASR system, where LM adaptation is handled by the system itself.

This thesis aims to deal with the task (3), which is the most difficult of all three tasks mentioned above. However, since methods used in other two tasks are often highly related to this work, these methods will be also briefly described in this chapter.

The first section of this chapter lists various approaches of how to accumulate the in-domain corpus, which is an important problem common to all LM adaptation methods (except for methods described in section 4.3). The second section presents commonly used multi-pass offline method, and in the last three sections, existing LM adaptation methods are grouped based on what information they use to adapt the LM. More detailed review of existing methods has been written down by the author of this thesis in his report for State doctoral examination (Lehečka, 2016).

4.1 In-Domain Corpus

Since each domain has its specific word and n-gram usage, the relevance, suitability and recency of in-domain corpus is crucial for the LM adaptation task. In the literature, a paradigm to accumulate relevant in-domain corpus according to the speech to be recognized, which can incorporate also tasks from different research areas (e.g. topic identification, information retrieval or text data mining), usually follows these steps:

1. Put all available knowledge about the speech together, e.g.:
    - use known metadata about the speech if available, e.g. announced theme of the speech, presentation slides (Maergner et al., 2012), information about the speaker, expected style or previous audio records (Tur and Stolcke, 2007),
    - use all information from the recognizer's output, which is available at the time of required LM adaptation, e.g. n-best list or lattices of a recent speech or a transcript obtained with general LM in the first decoding pass.
2. Use the knowledge about the speech to accumulate in-domain data; this is where tasks from different research areas are usually applied, e.g.:
    - the topic of the speech is identified in the available data, and based on this topic, corresponding prepared subcorpus in a topic-clustered database is selected (Shi et al., 2012; Echeverry-Correa et al., 2015),
    - the knowledge is used to retrieve relevant documents from some (external) data source, e.g. substantial keywords are extracted and used as queries to retrieve documents from a large text corpus (Chen et al., 2001, 2003) or when the domain can be absent from the corpus, documents can be retrieved from web via search engine (Berger and Miller, 1998; Zhu and Rosenfeld, 2001; Monroe et al., 2002; Sethy et al., 2005; Ito et al., 2008; Lecorvé et al., 2012; Maergner et al., 2012),
    - some distance between available documents and knowledge about the speech can be defined and only documents with minimum distance are used to form the in-domain corpus (Nanjo and Kawahara, 2003; Bigi et al., 2004),
    - some kind of data augmentation is used to obtain more data, e.g. based on semantic similarity of words used in the speech, new n-grams can be artificially generated without observing them (Janiszek et al., 2001).
3. Combine in-domain data with the general LM, e.g.:
    - use some adaptation technique described in section 3.3,

- select only relevant novel words and incorporate them into the general LM (Auzanne et al., 2000; Venkataraman and Wang, 2003; Federico and Bertoldi, 2004; Schwarm et al., 2004; Allauzen and Gauvain, 2005; Boulianne et al., 2006; Martins et al., 2007; Maergner et al., 2012).

4.2 Offline Multi-pass Method

Multi-pass speech recognition has been originally used to reduce computational demands of ASR systems with large LMs. The idea was to apply simple models earlier to prune the search space for subsequent passes using more computationally expensive models (Bates et al., 1993; Richardson et al., 1995).

This approach was then adopted into LM adaptation frameworks. The idea is to apply a general domain-neutral LM in the first decoding pass and based on information extracted from decoded result, find domain-coherent segments and apply progressively LMs more and more suitable for particular domains. This approach was used e.g. in (Martins et al., 2007; Lecorvé et al., 2012).

4.3 Methods Using Directly ASR Output

The main idea behind using ASR output directly to adapt the LM is that if the word or n-gram has been recently used, it is very likely to be used again in the near future. It is also the only possible adaptation method, if no in-domain corpus is available. Typical examples of these methods are *dynamic cache LMs* (Kuhn and De Mori, 1990; Jelinek et al., 1991; Clarkson and Robinson, 1997), where an unigram LM built from words recognized in a recent history (i.e. a cache) is dynamically interpolated with the general LM. However, experiments in (Iyer and Ostendorf, 1999) showed that static topic-specific LM can outperform the dynamic cache LM.

In the ASR output, there is usually a lot of errors and misrecognized words, which deteriorate the positive impact of adaptation using directly ASR output. One attempt to deal with this problem is to use more hypotheses from n-best list or from lattice and weight them appropriately. For instance in (Souvignier et al., 2000), each hypothesis in the n-best list is weighted by its confidence score. The idea behind this approach is that well-understood parts of a sentence will occur in most of the hypotheses of an n-best list, whereas for misrecognitions there will usually be several alternatives. Thus, the effect of a recognition error is distributed over several competing hypotheses and does not result in a strong error reinforcement.

Another attempt is to detect misrecognitions and do not use them for the adaptation. For instance, (Gretter and Riccardi, 2001) used word lattices to estimate an error probability distribution over recognized words, which is then used to filter possible misrecognitions by a probability threshold.

4.4 Methods Extracting Semantic Knowledge

Adaptation approaches taking advantage of semantic knowledge are usually based on finding semantically close words, n-grams or documents to the underlying discourse and tuning the LM appropriately.

One possible approach is to use *triggers* (Lau et al., 1993; Rosenfeld, 1996), usually in a form of *word trigger pairs*, which are pairs of words that co-occur frequently close to each other. The main idea behind this approach is that if one word from a trigger pair occurs in the speech, the second word is very likely to be used in a short time future, therefore its probability in the LM should be tuned. In practice, usually the mutual information has been used to reveal the relevant trigger pairs and exponential models, where each trigger pair forms a constraint to be satisfied, have been used to implement trigger pairs.

In present days, triggers are obsoleted by more systematic frameworks, which are able to reveal and learn relevant semantic relations between words and documents, such as LSA, LDA, word vectors and many other methods.

An example of an adapted LM can be e.g.

$$P(w|h,\tilde{h}) = \frac{P(w|h)\rho(w,\tilde{h})}{Z(h,\tilde{h})}, \tag{4.1}$$

where $\tilde{h}$ denotes the global bag-of-words representation of the history, $\rho(w,\tilde{h})$ is the correlation between the word w and history $\tilde{h}$ in the low-order vector space (e.g. LSA) and $Z(h,\tilde{h})$ is the normalization factor (Bellegarda, 2004). In contrast to standard n-gram LMs, LM 4.1 can encode much longer history into $\tilde{h}$ with no additional computational cost, however, the order of words in the history is lost since the bag-of-words representation is used.

Also, all methods searching for similar documents, which do not identify the topic of the speech explicitly can be put into this group of LM adaptation methods. In such approaches, usually some kind of IR task is performed in some data source. For instance, an ASR output (or only selected keywords (Chen et al., 2003; Ito et al., 2008)) can be used as a query to retrieve relevant documents to accumulate the in-domain corpus (Berger and Miller, 1998; Schwarm et al., 2004; Maergner et al., 2012), or the ASR output can be used to find most similar documents based on some distance measure (or similarity), such as cosine similarity (Ananthakrishnan et al., 2011), Kullback-Leibler distance between unigram probabilities (Bigi et al., 2004), perplexity of training documents with LM built from the ASR output (Nanjo and Kawahara, 2003; Haznedaroglu and Arslan, 2014) etc.

In (Korkmazsky et al., 2007), ASR lattices are used to form word clusters reflecting the mutual information between the words and distance in the lattice (in terms of speech frames), i.e. not only semantic, but also syntactic information is extracted. The best clusters are used to adapt the probabilities in the background LM. Very modest reduction in WER was achieved (from 59.2 % to 58.8 %) using this method.

In (Wiggers and Rothkrantz, 2006; Shi et al., 2011), an alternative to classical n-gram models using dynamic Bayesian networks is proposed. In contrast to n-gram models, it can take additional semantic information into account, since it is a generalization of the n-gram models and HMMs. For instance, the topic state or POS (part-of-speech) state can be explicitly modeled as a variable in the dynamic Bayesian network LM.

As an online semantic-based adaptation method, we can also count LM adaptation methods proposed for spoken dialog systems such as (Lucas-Cuesta et al., 2013), where the LM is dynamically adapted at each dialogue turn based on the knowledge extracted from the context of the ongoing dialogue. Here, the knowledge is in a form of a dialogue concept or the goal of a dialogue.

Most Relevant Methods

In this place, let us describe in more details several papers which fall into this group of methods and which are in my opinion most relevant to this thesis.

In (Shi et al., 2012), an online method for recognition of the spoken Dutch was presented. In this work, the most suitable LM is being dynamically selected from a set of prepared topic-specific LMs. The estimating of interpolation coefficients during the speech and LM interpolation itself is omitted. All LMs are prepared in advance and based on the recognized speech, the ASR system dynamically switches between prepared LMs. At each point in time, the LM with the highest probability of the current history is used in the ASR. Each topic-specific LM is interpolated from the complete corpus and topic-specific subcorpus, thus all LMs share the same vocabulary to avoid overfitting. Experiments showed roughly 12% reduction in perplexity, especially in the broadcast news domain, but reduction in WER was not evaluated. The main difference from our approach is that our system allows also to interpolate new topic mixture on demand during the recognition and TI is used to extract topic information from the recent hypotheses.

In (Maergner et al., 2012), LM adaptation via interpolating background LM with topic-specific LM was applied on lecture speech recognition. In advance of each lecture, the text from the presentation slides had been used to select similar documents on the web. Documents were searched based on queries in the search engine. Queries were short text sequences from the slides not containing common words. After some amount of documents had been collected, a vocabulary of the most relevant words was created and background topic-independent LM was adapted, which resulted in 57% relative reduction of OOV-rate and 12.5% relative reduction of WER. The LM adaptation in this work was not applied during the recognized speech, but in advance.

(Martins et al., 2007) presented an offline adaptation of LM in WFST-based ASR system recognizing Portuguese broadcast news on daily basis, i.e. once a day it processes all video records from the last 24 hours. Each day, as the background data, general corpus is interpolated with web news articles from the last one week. Since the size of the vocabulary is fixed in this work, new words found in new articles replace low-frequency words from the background vocabulary. Not all words are added into the vocabulary, but a sophisticated algorithm based on the distribution of POS classes in the in-domain corpus is used to select only the most relevant words to be added. Hypotheses from the first decoding pass are heuristics-based segmented into topic stories and used as queries to the IR database to retrieve relevant documents to mix a topic-specific LMs for the second decoding pass. Experiment results on two hours long evaluation speech showed that a significant reduction in OOV-rate and WER is obtained after the first decoding pass and some more reduction after the second pass. From the baseline, a reduction of 65% in OOV-rate and 6.5% in WER was achieved while covering almost every topic-specific words and keeping the vocabulary size fixed ($57k$).

In (Federico and Bertoldi, 2004), an offline LM adaptation in Italian broadcast news transcription system is presented. On a daily basis, news reports are downloaded from the web servers, novel words are extracted, their phonetic transcriptions are automatically generated and the background LM is interpolated with small adaptation LM trained on selected news texts from the repository. Experiments showed relative reductions of 58% in OOV-rate, 16% in perplexity, and 4% in WER. Basically, it is very similar work to (Auzanne et al., 2000), but the adaptation is done not on weekly basis, but on daily basis.

4.5 Topic-based Methods

In topic-based LM adaptation approaches, the general LM is adapted in order to adjust to the underlying topic of the speech. In online topic-based methods, topics must be identified at the precise time when they appear in the speech. Since this thesis is focused on online topic-based LM adaptation, papers mentioned in this section are the most related existing approaches we are aware of.

A work which is to my knowledge the most related to this thesis, is (Echeverry-Correa et al., 2015). In this paper, various topic identification methods and various dynamic language model adaptation methods are tested on TI and ASR task using data from Spanish partition of the European Parliament Plenary Sessions. The proposed system has 2 recognition passes. In the first one, the general LM is used to generate hypotheses, from which the topics are identified. Then, the general LM and previously trained topic-specific LMs are used to interpolate the adaptation LM, which is then used to re-decode the audio signal in the second pass. In the TI experiments, two different document representations have been tested: BOW and LSA. Results suggested, that LSA improves the TI error over the BOW-based approaches. The improvement is significant especially for shorter audio segments (roughly 1 minute). In the dynamic LM adaptation task, 3 different interpolation schemes have been tested: (1) hard (interpolate 1 topic LM), (2) soft (interpolate all topics), (3) top10 (interpolate ten top-scoring LMs). Surprisingly, the language model was adapting better to short segments even though the topic identification error is increased for those segments. The best results have been obtained when using top10 interpolation scheme. Another surprising result in this paper is that topic-specific LMs based on automatic document clustering (k-means with LSA representation) perform better than LMs based on manual labeling of the corpus. The main difference from our approach is that presented method is adapting the LM only on speaker-turns or after roughly 1-minute audio segment and then the whole segment is re-decoded, while we are aiming at an online ASR system, which analyzes the ASR output at each time step and adapts LM without re-decoding the signal.

In (Boulianne et al., 2006), a system for French closed captioning of live TV broadcasts in Canada was presented. The recognizer is WFST-based 1-pass system and its speed is 0.7xRT with maximum delay of 2 seconds between the input (speech of shadow speaker) and the text output. Before a live captioning session, a number of configurations are preloaded in memory, so that switching between topics and speakers happens instantly during the session. Topic LMs are interpolated in advance based on optimizing the perplexity of held-out data. The client software suggests before each session novel words, pronunciation and association with topics to be validated by a shadow speaker. During live production, the shadow speaker can change topics and insert punctuation or other symbols. After the session ends, the shadow speaker listens again to his voice and correct all recognition errors to produce data for supervised training. The web-crawler, which feeds the text database on a daily basis, retrieves about 1 million words of text every night. On average, 6000 of these are unknown to the system, but only 200 or so will survive the garbage filters and be added to the database and proposed to the user for verification. The main difference from our approach is that in the paper, the topic identification step is done manually by a shadow speaker, while we are aiming at fully automatic system.

In several papers, the TI block has been implemented as a topic model (see section 3.2.4). Here, we will briefly describe those of them that are most related to this work.

- In (Tam and Schultz, 2005), topic mixture weights are dynamically updated based on decoded words (each word is in LDA modeled as a mixture of latent topic) and the in-domain LM is interpolated with the background trigram LM.
- In (Hsu and Glass, 2006), an extension of LDA, HMM-LDA, was used. In this work, LM is adapted not only to the underlying topic, but also to the style of the speech modeled with latent syntactic states.
- In (Watanabe et al., 2011), the TI block is implemented as a latent topic tracking model, which is an extension of LDA estimating additional parameters (Dirichlet priors) of word-topic probabilities depending on the previous time steps, which implements the time-dynamic. However, in this work, the improvement in terms of WER is very modest at the cost of slowing the recognition to 2xRT.
- In (Chen et al., 2015), adapted RNN LMs are (offline) trained using vectors from topic models as an additional input.
- An online fully-unsupervised LDA-based approach was published also by the author of this thesis (Lehečka and Pražák, 2018).

Chapter 5

Proposed Solution and Implementation

In this chapter, an innovative solution to the problem of online topic-based LM adaptation is presented. The solution is a complex system, in which the existing online ASR system (Pražák et al., 2012) was extended with an online topic-based LM adaptation feature by the author of this thesis. The scheme of proposed solution will be outlined and broken down into component blocks. Then, we will look into each component block and describe details about how it works, how it is implemented, what is the default setting and which parameters are optional to be experimented with. A complete list of configurable parameters can be seen in appendix A.

5.1 The Scheme of the Solution

The basic scheme of the solution is outlined in figure 5.1. The core of the system is online automatic speech recognizer (*Online ASR* block described in section 5.2), which is processing the input audio stream in the real time and generating partial hypotheses about the content of the speech so far.

My first improvement over the existing (*baseline*) ASR system was to add one more LM into the decoder resulting in a *Parallel Decoder* generating two different partial hypotheses at each time step: one using a general LM and one using an adapted LM, which can be replaced on the fly.

My second improvement is a *Hypotheses Merger*, in which information from both partial hypotheses is merged together while favoring the information from adapted-LM decoder (details in section 5.3).

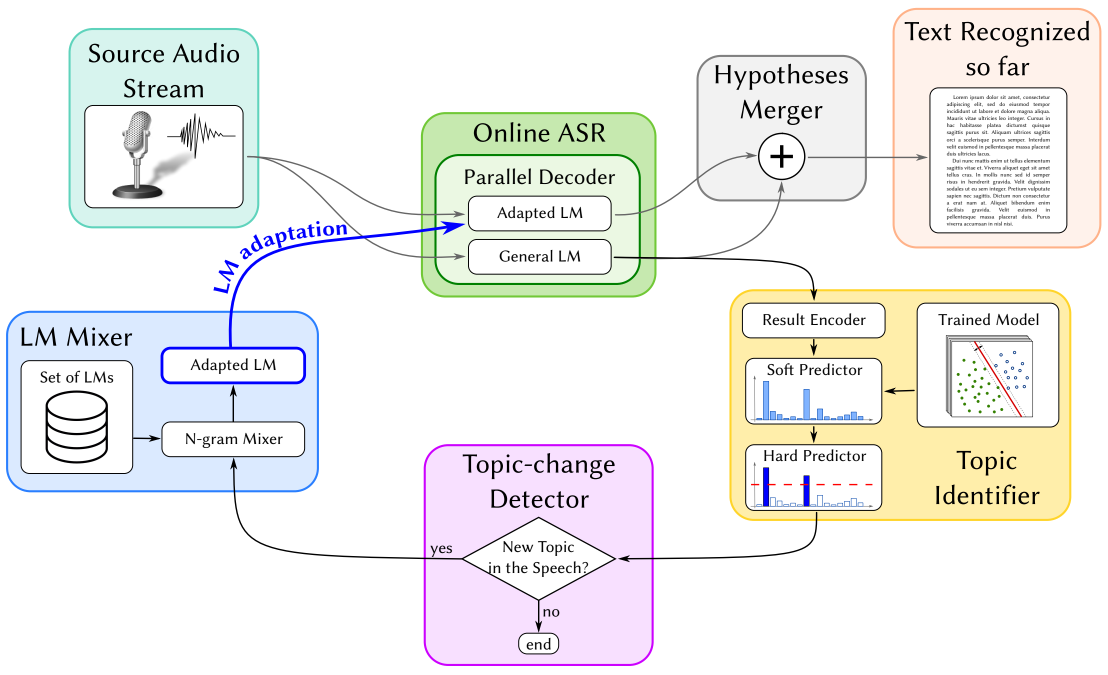

FIGURE 5.1: The basic scheme of proposed solution.

My final improvement is a sort of "feedback" adaptation loop around the ASR system, in which current-speech topics are identified (*Topic Identifier* block, section 5.4), checked if they are new in the speech (*Topic-change Detector* block, section 5.5) and if so, a suitable adapted LM is prepared (*LM Mixer* block, section 5.6) and used to replace the older adapted LM in the ASR's decoder (*LM adaptation* arrow). After that, the ASR immediately starts to generate decoded hypotheses using new adapted LM. In this way, the system is being adapted online based on the current (or very recent) topics in the speech.

The originality of this solution lies in minimizing latency of the adaptation by using two parallel decoders (instead of commonly used offline multi-pass approaches) and online merging their outcomes. The *Topic Identifier* and *LM Mixer* blocks incorporate mostly publicly known models and approaches. However, the way the *Topic-change Detector* decides in the real time about the change of the topic and the connection of all blocks in the adaptation loop is an innovative contribution of this thesis, since we are not aware of any published work with similar scheme.

The processing time needed for all computations in the adaptation loop depends on how fast a computer is able to process particular blocks. In the *Topic Identifier* block, only a short part of incoming hypothesis (only the most recent part) is typically analyzed in order to identify current topics, therefore this block is causing only a small delay depending on the complexity of the trained model. Similarly, the block *Topic-change*

Detector is usually evaluated very fast, since it typically consists only of several *yes-or-no* rules. The *LM Mixer* block, on the other hand, is the critical time-consuming part of the adaptation loop, where a lot of computations need to be done as fast as possible (see section 5.6.5).

In some special cases discussed later in section 5.6, the *LM Mixer* block can be accelerated by precomputing a restricted set of LM images[1] leading to a dramatically fast adaptation with almost no delay caused by slow computing.

I decided to write the system in *Python* programming language[2], because there has been developed a vast amount of easy-to-use modules for fields like topic modeling, machine learning, NLP, neural networks training etc., all written in Python and publicly available. Also, our online ASR system has a Python API, so decoding process can be controlled directly from a Python code and returned partial results can be Python objects. The implementation of the system is fully modular and hence it can be easily extended by plugging in any algorithm other than those presented in following sections, if the need be.

5.2 Online ASR

The cornerstone of proposed system is an LVCSR system developed on our department for low-latency real-time ASR (Pražák et al., 2005). This system is being continuously developed and extended with new methods to perform state-of-the-art recognition with accent on online decoding and live captioning of TV shows (Pražák et al., 2012). It is written in *C++* programming language with an API wrapped into Python.

For acoustic modeling, the ASR system uses three-state HMMs with output probabilities modeled by a Deep Neural Network (DNN).

When the system starts, a parallel decoding using 2 decoders at the same time is initialized. The main purpose of particular decoders is following:

- Decoder D_G uses a general LM. It is immutable during whole recognition and thus it provides continuous stable stream of partial results (hypotheses) to the adaptation loop in order to identify topics dynamically. It is also used in *Hypotheses Merger* to fill potential gaps in decoded results emerging from changing adapted LMs.

[1]A special data structure suitable for fast loading into the decoder in ASR system
[2]`https://www.python.org`

- Decoder D_A uses an adapted LM. It is a variable decoder, where the LM adaptation takes place during the recognition on the fly.

After each LM adaptation, D_A can start decoding with a feature called *retro-recognition*. It means that it first re-decodes several last seconds of the audio stream and then continues with online recognition. *Retro-recognition* can be used for online corrections of the last decoded words using the new adapted LM and thus it can compensate short time-delays emerged from the topic-change detector or LM mixer. However, *retro-recognition* can be used only for several seconds back into history. Re-decoding longer epochs is impractical, since the decoder must wait until *retro-recognition* finishes establishing real-time decoding, and so, partial results would be too delayed behind the speech.

To implement the change of LM in the decoder D_A whenever the change in the topics of the speech is detected and corresponding LM is prepared, following sequence of commands is used:

1. Function *StopParallelRecognition()* is called to stop decoding with D_A. Decoder D_G continues unchanged.
2. Function *ChangeLanguageModel()* is called to switch the LM in D_A. Optional arguments *LanguageModelWeight* and *WordInsertionPenalty* can be specified here.
3. Function *StartParallelRecognition()* is called to start again decoding with D_A. An ID of a frame, from which D_A starts recognizing, must be specified as an argument. Here, *retro-recognition* can be applied by passing ID of a frame from some (reasonable) past time.

Additionally, LVCSR has many parameters which can influence the decoding process, the acoustic model behavior, the audio processing etc. To name few of them which are important for our work, one can set *ResultPeriod* (how often partial results are reported), *ProcessorNumber* (how many cores are used) and the type of returned results (lattices, 1-best hypotheses, ...). A complete list of configurable parameters can be seen in appendix A.

5.3 Hypotheses Merger

The *Hypotheses Merger* combines hypotheses from decoders D_G and D_A while favoring the information from D_A, if available. Let us denote 1-best hypothesis from decoder D_G as H_G, 1-best hypothesis from D_A as H_A and the combined (merged) hypothesis as

H_M. The merger uses always H_G as a base information for H_M and whenever H_A is available, it sticks H_A to H_G and thus overlays information from the general LM with information from adapted LM in the output H_M.

An example of merging H_G with H_A is depicted in figure 5.2. Without the knowledge about the topics, the decoder D_G is not able to decode correctly topic-related words like "Apple", "Jobs" or "NeXT", because the general LM assigns higher probabilities for homonyms[3] "apple", "jobs" and "next" which are used more often in a common speech. However, when our system sees the word "computer" and bigram "Steve Jobs", it likely triggers LM adaptation to the IT domain, where IT-related n-grams have much higher probabilities. As long as the speech is related to the IT domain, both decoders run in parallel and the merger gives preference to D_A over D_G (words with blue background). Whenever the topic-change detector decides the IT domain is no longer in the speech, because another topics have just appeared, the system stops D_A, merges H_A with H_G and starts LM adaptation to the new topics.

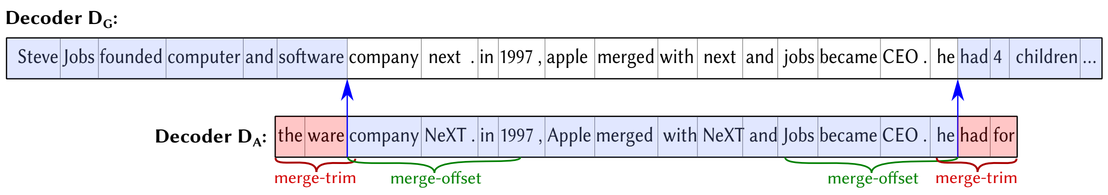
Decoder D_G:
Steve Jobs founded computer and software company next . in 1997 , apple merged with next and jobs became CEO . he had 4 children ...
Decoder D_A:
the ware company NeXT . in 1997 , Apple merged with NeXT and Jobs became CEO . he had for
merge-trim
merge-offset
merge-offset
merge-trim

FIGURE 5.2: Example of merging two hypotheses. For better legibility, punctuation was added into decoded text.

Based on my observations, the decoded words at the very beginning and end of H_A can be unreliable due to the lack of context and because D_A can be started or stopped in the middle of some word. For that reason, few seconds from both ends of each H_A are dropped (red words in the figure 5.2) and the best place to cut a gap in H_G and stick H_A to it is found. Finding the best place to cut and tie both hypotheses is based on finding two words with the same (or very close) start-timing at the beginning of H_A and two words with similar end-timing at the end of H_A.

The Merger has two optional parameters depicted also in the example 5.2: ***merge-trim*** defines how many seconds to drop from the beginning and end of H_A (default is 2). Only words lying entirely in the trimmed interval are dropped. Parameter ***merge-offset*** defines the width of time-offset at the beginning and end of the rest of H_A, in which the place to cut and tie both hypotheses must be found (default is 3 seconds). If words with the exact match of timings are not found, the merger selects two words with minimum timing-difference within the allowed time-offset. This ensures that information

[3]Words which sound alike or are spelled alike, but have different meanings.

from H_A will be used even if there are completely different words with different timings in both hypotheses.

5.4 Topic Identifier

Now, let us dive into the adaptation loop. The aim of the first block, the *Topic Identifier* (TI), is to take the partial result from the general-LM decoder D_G and decide which topics are present in the current speech. The input result can be either a lattice or 1-best hypothesis (see section 3.1.5). The detailed scheme of topic identifier implemented in the system along with all parameters is depicted in figure 5.3.

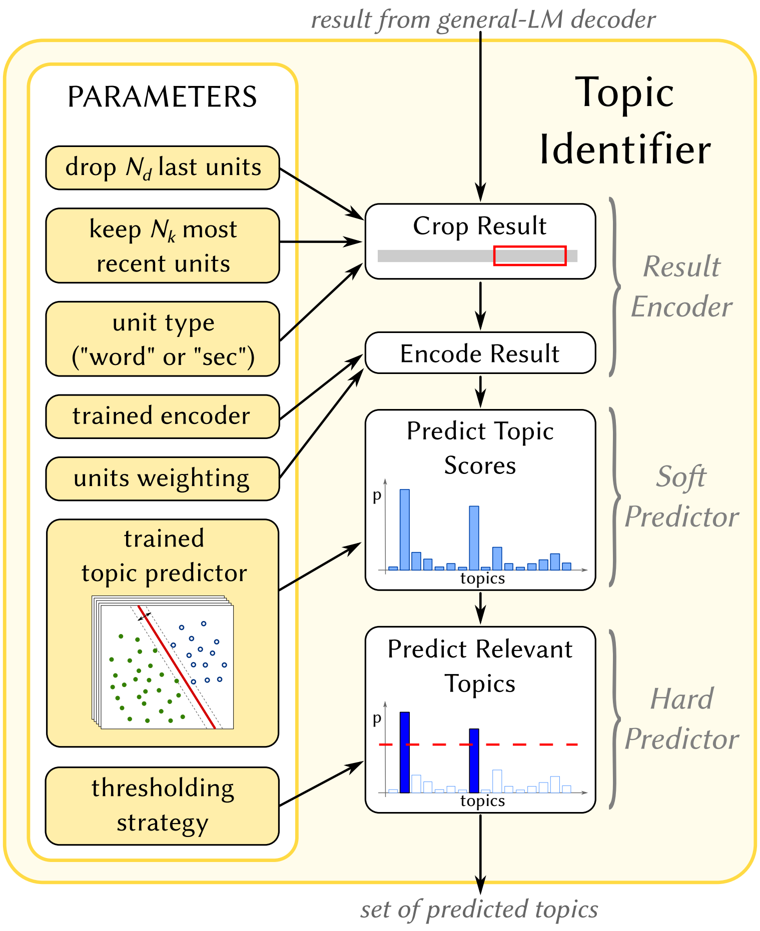

FIGURE 5.3: The detailed scheme of topic identifier.

In this place, let us clarify the terminology of using terms *topic* and *topics*. Since our system works with both single- and multi-label topic identifiers, the speech can theoretically contain any number of topics at any time. When speaking generally about some TI without specifying if it solves single- or multi-label problem, we might use terms like "topic or topics", "one or more topics" or similar. However, since this term will be very frequent in this thesis, we decided to use rather the short term *topics* whenever speaking generally about some TI, meaning any number of topics. On the other hand, we will use the term *topic* to underline that we mean exactly one sole topic.

5.4.1 Result Encoder

The first part of the TI subsystem, the result encoder, aims to convert the important part of the input result into a suitable numerical representation (vector or matrix).

Crop Result

First of all, the input result needs to be cropped in order to select only the important part of it. Based on my experience, it is reasonable to throw away the last few words of the result as there is usually high uncertainty about them in partial hypothesis, and focus only on short recent window. In some applications, it is desired to define boundaries of the window in number of words (e.g. when a fixed number of words is required by a predictor), sometimes it is better to use a time-window (e.g. when there are long non-speech parts in the speech between topics).

To control boundaries of the window in focus, three optional parameters were added: **unit type** (”word” or ”sec” defining whether boundaries are word counts or seconds), **drop N_d last units** (how many last units are thrown away) and **keep N_k most recent units** (how many last units are selected from remaining result, i.e. the width of the window).

When the result is a lattice, boundaries must be set in seconds. When the result is 1-best hypothesis, both words and seconds can define boundaries. The default setting is to drop last 2 words and then select window of fixed size 50 most recent words.

Encode Result

Next, the cropped result is processed by a **trained encoder** to produce a numerical representation of the cropped result. It must be fully compatible with the trained topic predictor to produce correct prediction, that is why these two blocks (encoder and predictor) are usually trained together and used as one model. However, if units weighting (discussed later) is used, it must be applied on the numerical representation, i.e. after encoding result and before predicting scores. That is why the result encoding is split into encoder and predictor blocks.

Following text encoders were incorporated in the system:

- ***tf-idf*** — bag-of-words model combining sublinear cosine-normalized *tf* with *idf*; it must be used in combination with SVM-like models; it is based on python implementation from Scikit-learn package (Pedregosa et al., 2011);
- ***word2vec*** — sequence of *fastText* word vectors (Bojanowski et al., 2016); it must be used in combination with LSTM-like models; to handle word vectors, python Gensim package (Řehůřek and Sojka, 2010) is used and during encoding, word vectors are assembled into a matrix representation capturing also the order of words.

If the result is in form of a lattice, one of following alternative encoders must be used:

- ***lattice_tf-idf*** — probabilities of each term are aggregated across whole lattice and used as term frequencies to build a *tf* vector;
- ***lattice_word2vec*** — the lattice is converted to word confusion network (also known as a sausage) and for each node in the sausage, a weighted sum of word vectors is computed based on probabilities of outgoing arcs.

Units Weighting

By default, if the result encoder does not take the order of words into consideration (e.g. *tf-idf*), all words within the selected window are equally important. However, since more recent words are naturally more significant for evolving topics, it seems reasonable that the very recent words should have higher weights than older ones.

In our system, there is parameter **units weighting** to define which weighting function should be applied on the selected window. By *units* we mean all words (in case of 1-best hypothesis) or nodes (in case the result is a lattice) within the window.

Let $U = (u_1, u_2, ..., u_N)$ be the sequence of N units we want to analyze, u_1 the oldest unit from this sequence and u_N the most recent one. We are looking for a function $f : u_i \mapsto \alpha_i$, $i \in [1, ..., N]$, $\alpha_i \in \langle 0, 1 \rangle$, where α_i is the weight of unit u_i (the higher, the more important the unit is). I have experimented with several functions plotted in figure 5.4.

Mathematically, these functions are defined by following formulas:

- **constant** (no weighting, default):

$$\alpha_i = 1, \tag{5.1}$$

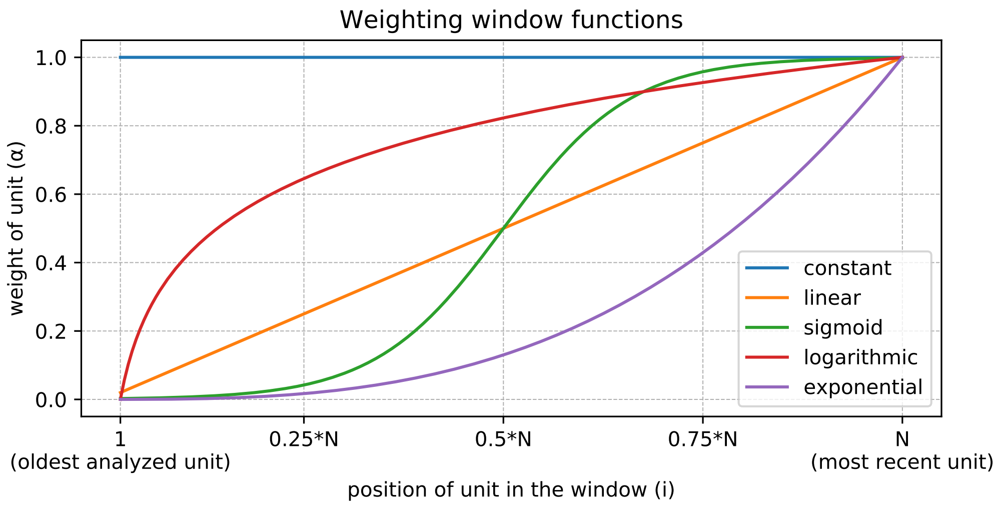

FIGURE 5.4: Weighting functions which can be applied on a result-window. Higher weight means higher importance of the unit in the encoded result. This graph has been plotted for $N = 50$ units.

- **linear**:

$$\alpha_i = \frac{i}{N}, \tag{5.2}$$

- **sigmoid**:

$$\alpha_i = \frac{1}{1 + e^{-\lambda(i - \frac{N}{2})}}, \tag{5.3}$$

- **logarithmic**:

$$\alpha_i = \log_N(i), \tag{5.4}$$

- **exponential**:

$$\alpha_i = \frac{1}{10^p} \cdot i^{\frac{p}{\log_{10}(N)}}. \tag{5.5}$$

Based on my expectation of how the weights should evolve in time, I set parameter $\lambda = 0.25$ in 5.3 and $p = 5$ in 5.5. As can be seen from figure 5.4, all functions assign minimal weight for the oldest unit and $\alpha_N = 1$ to the most recent one, but each function with a different trend.

These weights are then used to scale corresponding term frequencies in the *tf* weighting scheme or corresponding word vectors in ***word2vec*** matrix, so the recent words influence the predictions more than older ones according to selected function.

5.4.2 Soft Predictor

The aim of a soft predictor is to convert encoded result into one vector characterizing how likely is each topic present in the speech, i.e. to predict the per-topic scores. In the literature, there can be found a vast collection of such predictors (see section 3.2), each having many hyperparameters, therefore it is not feasible to experiment with all of them. Based on my experience and knowledge, several most promising and time-tested predictors were selected and included into the system. However, the system is fully modular and hence it can be easily extended by plugging in any other predictor, if need be.

For datasets with topic-labeled data, I experimented with following supervised classifiers:

- **SVM** — Support Vector Machine classifier with linear kernel and *one vs. the rest* multi-class strategy; the input to the model must be tf-idf vectors; python implementation from Scikit-learn package (Pedregosa et al., 2011) is used;
- **calibSVM** — the same as *SVM*, but extended with a probability calibration wrapper, so the output is a probability distribution over topics (see section 3.2.3 for details);
- **LSTM** — recurrent neural network with 3 LSTM (Long Short Term Memory) layers, each with 512 neurons (including max-pooling and dropout), followed by one dense output layer with sigmoid activation function; the input sequence is limited to the first 500 words[4], shorter documents are zero-padded; the input to the network must be a sequence of word vectors, e.g. (Mikolov et al., 2013), (Bojanowski et al., 2016) or similar; python implementation based on Tensorflow library (Abadi et al., 2015) is used.

For datasets without any topic-labels, I experimented with following unsupervised topic model:

- **LDA** — Latent Dirichlet Allocation model with the number of clusters as a hyperparameter, trained in 5 passes over training data; the input is a plain text, i.e. cropped 1-best hypothesis without any encoding; python implementation from Gensim package (Řehůřek and Sojka, 2010) is used.

[4]Based on my experiments, by looking at the first 500 words of a document, LSTM classifier gets enough information to decide about the topic.

5.4.3 Hard Predictor

The per-sample prediction from a soft predictor is a vector of per-topic scores $s_1, s_2, ..., s_K$, which has to be thresholded to obtain binary topics assignment. See section 3.2.5 for details.

Based on my experiments and knowledge, following thresholding strategies were included into the system:

- **FixedCut(t)** — fix the threshold globally to the same value t for all topics and samples, i.e. all topics with $s_i > t$ are assigned; the natural strategy for SVM classifier is FixedCut(0);
- **MCut** (*Maximum Cut*) — try to select only topics which have "really higher" scores than the others by identifying the highest difference between successive scores (Largeron et al., 2012);
- **RelCut(p)** (*Relative Cut*) — set the threshold for each sample relatively to the maximum predicted score, i.e. i-th topic is assigned, if $\frac{s_i}{\max(s_1, s_2, ..., s_K)} > p$; the most common setting is RelCut(0.5);
- **RCut(k)** (*Rank-based Cut*, also known as *k-per-doc*) — for each sample, assign exactly k topics with the highest scores; for example, RCut(1) ensures exactly one topic is assigned for each sample.

The output from the hard predictor is a set of identified topics which is being online monitored by a *Topic-change Detector*.

5.5 Topic-change Detector

The aim of a *Topic-change Detector* is to decide whether the current and past topics identifications provide sufficient evidence of a change in the speech topics. The detailed flowchart of topic-change detector implemented in our solution along with all parameters is depicted in figure 5.5.

Since topics are identified from only a short word history, there is usually some noise in each individual prediction which must be somehow smoothed out over longer time period. In our system, the key parameter which controls the smoothing of predictions is **topic steadiness**. It defines, how many seconds into the history a topic must have been continuously identified to be marked as a *steady* topic in the speech. For example,

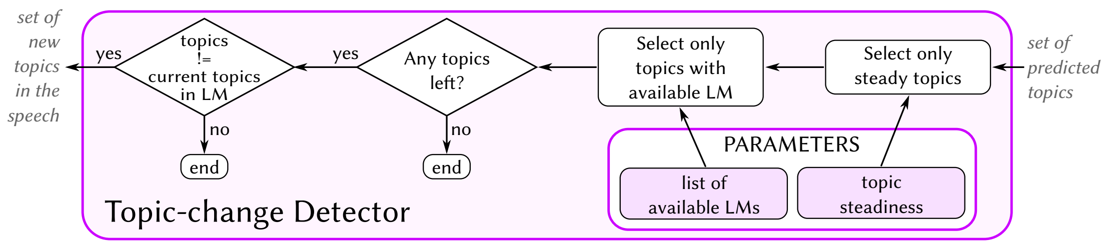

FIGURE 5.5: The detailed scheme of topic-change detector.

if topic steadiness is set to 5, only topics which occurred in every topic identification during the last 5 seconds are considered as *steady*.

By increasing the steadiness, the topic-change detection becomes more smooth and safe, but also more delayed behind the real changes of topics. This delay can be partially counterweighted by a retro-recognition, but only few seconds of retro-recognition is possible to recompute in real-time applications.

After selecting steady topics, the existence of a single-topic LMs is assured by checking the **list of available LMs** and if there is no topic left, it means that right now, there is no steady topic in the speech the system can be adapted to, and the algorithm ends the loop here.

Remaining steady topics are compared with current topics in adapted LM. If they are equal, the topics in the speech still correspond with the adapted LM in the decoder and there is no need for any change. However, if they differ, new topics in the speech have just been detected and adaptation is triggered by sending new topics to a *LM Mixer* block.

5.6 LM Mixer

The aim of a *LM Mixer* is to prepare a mixture of LMs suitable for LM adaptation based on detected topics in the speech. The detailed flowchart implemented in our solution along with all parameters is depicted in figure 5.6 and unfolded *mix LM* block later in figure 5.7. The *LM Mixer* can work in 3 modes: *online*, *offline* and *prepared*. Based on selected working mode, LMs are prepared differently.

Since preparation of LMs is computationally intensive process and topics are likely to re-occur in the speech, each adapted LM is cached (stored on disk) for future usage. The amount of lastly-used LMs stored in the cache is controlled by **cache size** parameter. The *Update cache* block maintains the list of lastly-used LMs in the speech and removes

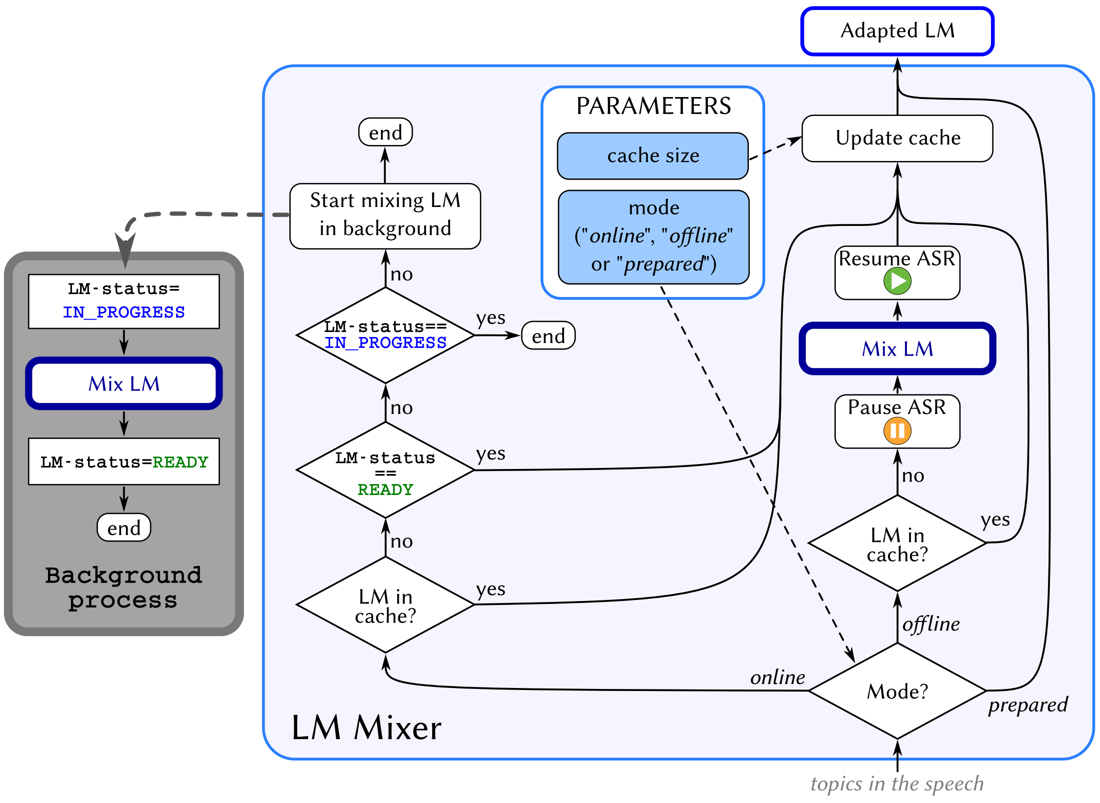

FIGURE 5.6: The detailed scheme of LM mixer.

the old ones which have not been seen for a long time in order to limit the disk usage. The only exception is *prepared* mode, in which caching is not necessary as all LMs are supposed to be already prepared (cached) on the disk.

5.6.1 Prepared Mode

The simplest and fastest working mode is *prepared mode*. It assumes all adapted LMs have been prepared offline and are stored on the disk to be used, therefore real-time recognition with almost no adaptation delay can be achieved using this mode. However, it can be used only when a reasonable number of distinct LMs is to be used during the recognition.

For example, if we have 20 topics and we know there will never be a combination of more than 3 topics at the same time in the speech, there are $\binom{20}{3} + \binom{20}{2} + \binom{20}{1} = 1350$ distinct combinations of topics. For each one, we want to train an adapted LM with size about 500 MB[5], then we would need 675 GB of disk space. That is possible, but now imagine 1000 topics (which is nothing unusual in the real world applications): we would

[5]A typical size of LMs used in our ASR system

have to save $\binom{1000}{3} + \binom{1000}{2} + \binom{1000}{1} \approx 167$ million LMs on a storage with 83 PB free disk space available. And of course, this is the simple uniform combination not taking various weights of topics into consideration. It is obvious, that with increasing number of topics and possible topic-combinations, the required disk space very quickly exceeds the capacity of any affordable storage. That is why *prepared mode* can be used only in tasks, where the number of distinct adapted LMs is reasonably small.

5.6.2 Online Mode

The *online mode* is designed for online applications, where the real-time ASR runs continuously and there is no way all possible adapted LMs can be stored on the disk. Let's denote newly discovered set of N topics in the speech as $T = \{t_1, ..., t_N\}$. Each T gets a *LM-status* variable S_T holding information about the preparation progress. The mixing of adapted LM itself runs as a background process in parallel with the recognition. First, S_T is set to "*IN_PROGRESS*", and as soon as the mixed LM is prepared, S_T is set to "*READY*".

When *LM Mixer* enters the online mode, it first checks whether any computing is necessary. If the desired LM has been recently used, the cached LM is used. If $S_T =$ "*READY*", it has been prepared recently and is ready to be used. If $S_T =$ "*IN_PROGRESS*", the request to prepare this LM is already in progress and the algorithm can not use it right now. If all these 3 decisions are negative, a new background process is triggered and the *LM Mixer* ends here waiting for the future topics identifications while the new mixed LM is being prepared silently in the background and the ASR continues with no changes. In this way, many various LMs can be preparing at the same time in parallel and whenever one of them is ready, it can be immediately used in the decoder, if the corresponding topics are present in the speech.

5.6.3 Offline Mode

The *offline mode* is designed for offline applications where the real-time operation is not required, e.g. to recognize an audio record. In such case, with each new set of topics in the speech, the ASR is paused, the desired LM is prepared (whatever time it takes) and after that, the ASR is resumed with the new LM ready to be used.

This mode can also simulate how an online application would work if the conditions were ideal, i.e. if all LMs can be prepared in advance and stored on disk or if the LM preparation takes no time at all.

5.6.4 Mix LM block

The *Mix LM* block is the main core of *LM Mixer* in both *online* and *offline* modes. It executes the most computationally intensive parts as its subprocesses and waits for completion of the mixed LM. The scheme of this very important block is in figure 5.7.

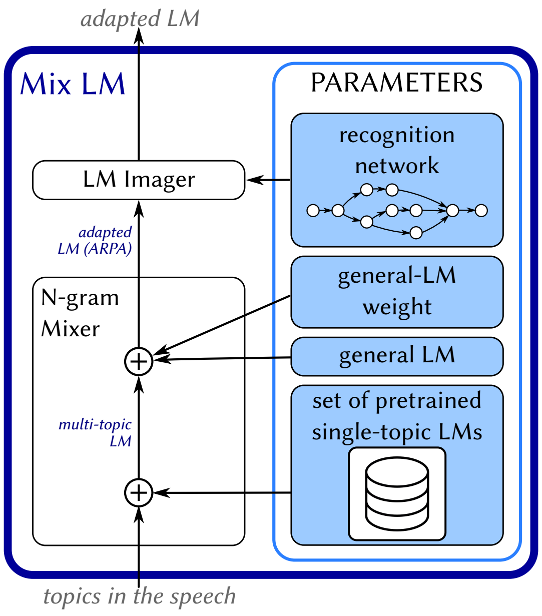

FIGURE 5.7: The detailed scheme of mix LM block. The round block with "+" symbol denotes linear interpolation of static n-gram LMs.

In ***N-gram Mixer*** block, *single-topic LMs*, i.e. LMs trained from documents related to only one topic, are selected from a *set of pretrained single-topic LMs* (technically a mapping from topics to LM paths) based on topics identified in the speech. Then, a command to mix them together is assembled and run as a subprocess. The result of this process is a *multi-topic LM*, which is a LM interpolated from one or more single-topic LMs.

For any sequence of words W, let us denote multi-topic LM as $P_{mt}(W)$, number of identified topics in the speech as N, single-topic LM for i-th identified topic as $P_i(W)$ and its mixture weight as λ_i. Assuming that $\sum_{i=1}^{N} \lambda_i = 1$, the multi-topic LM can be written as linear interpolation

$$P_{mt}(W) = \sum_{i=1}^{N} \lambda_i P_i(W). \tag{5.6}$$

It might seem reasonably to set $\lambda_1, ..., \lambda_N$ based on scores obtained from the soft predictor and thus, reflect the various importance of detected topics in the speech. However, unlike binary predictions, topics scores obtained from the soft predictor (usually float numbers) can be highly changeable and it is not very likely that the exactly same scores will be seen more than once during the recognition. The effort to prepare an adapted LM reflecting exactly identified topics and predicted scores can lead to: (1) new LM adaptation at each single time step (in the *offline* mode) or (2) no LM adaptation at all, because any prepared LM will never match predicted scores again (in *online* mode).

To avoid problems (1) and (2), I decided to use always uniform weights for all single-topic LMs, i.e.

$$\lambda_i = \frac{1}{N}, \quad i = 1, ..., N. \tag{5.7}$$

When using uniform weights, once prepared adapted LM can be used even in situations, when the topic identifier predicts the same topics, but with different scores.

Based on my experiments, it is usually better to add also a general topic-independent LM into each mixture, because it guarantees the complete vocabulary and general n-grams of the language to be present in the adapted LM. Therefore, after mixing the multi-topic LM, it is interpolated with *general LM*, $P_G(W)$. The path to the general LM and its mixture weight $\lambda_G \in \langle 0, 1 \rangle$ are optional parameters. The default value is $\lambda_G = 0.5$. If $\lambda_G = 0$, the general LM is not added into the mixture. The result of this interpolation is the desired *adapted LM*, $P_A(W)$:

$$P_A(W) = (1 - \lambda_G)P_{mt}(W) + \lambda_G P_G(W). \tag{5.8}$$

When put equations 5.6, 5.7 and 5.8 together, we can write a formula to interpolate all LMs in one single step:

$$P_A(W) = \frac{1 - \lambda_G}{N} \sum_{i=1}^{N} P_i(W) + \lambda_G P_G(W). \tag{5.9}$$

This single-step interpolation is implemented in our system to minimize the mixing time. However, in experiment schemes in this thesis, usually 2-step interpolation (i.e. compute first eq. 5.6 and then eq. 5.8) is depicted in order to preserve the concept of single- and multi-topic LMs, which is in my opinion more intuitive. It is just for the sake of visual clearness in figures, resulted LMs are the same for both single- and 2-step interpolations.

To apply static linear interpolation 5.9 in our system, SRILM toolkit (Stolcke et al., 2002) is used, specifically *ngram* program. The output from *n-gram mixer* is the adapted LM in a binary ARPA backoff n-gram format[6].

[6]`http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html`

Finally, in the ***LM Imager*** block, the LM is combined with a *recognition network* and saved as a LM image, which is a special data structure suitable for fast loading into the decoder in ASR system. This image can efficiently replace current adapted LM in the decoder on the fly.

5.6.5 Speed of LM Mixer

It has been already mentioned that the most computationally intensive part of the whole adaptation process is *LM Mixer*, specifically *N-gram Mixer* and *LM Imager* blocks. It is worth a short analysis of processing time these blocks are consuming.

We took a sample 3-gram LM containing 1.2 million words, 27 million bigrams and 20 million trigrams (1.2 GB on disk), performed several experiments and measured consumed CPU time of *LM mixer* running on 1 core of Intel Core i5-3470 machine.

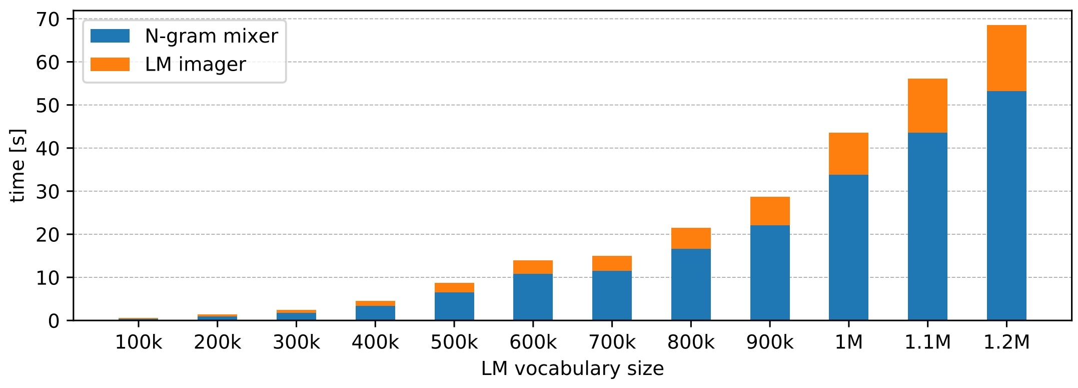

FIGURE 5.8: Processing time of LM mixer with increasing size of vocabulary. The total time consists of mixing together two identical n-gram LMs and converting the mix into a LM image suitable for the decoder in ASR system.

Figure 5.8 shows how the LM mixer scales with enlarging LM. For each particular run, we created a new LM with limited vocabulary, where only first N words (alphabetically) were included and all other words were replaced with *unk* token in all n-grams. The request to mix this reduced LM with itself was sent to LM mixer and the consumed time of particular blocks was measured. As can be seen, small LMs are mixed together very fast, but from the vocabulary size of about 600 thousand words, the consumed time starts to be impractical, especially for *online* mode. The main portion of processing time is consumed by *N-gram Mixer*, i.e. by the SRILM toolkit.

Figure 5.9 shows how the *LM mixer* scales with increasing number of mixed LMs. We took LM restricted to the first 500 thousand words from the previous experiment and sent a request to mix N identical LMs together. As can be seen, processing time of

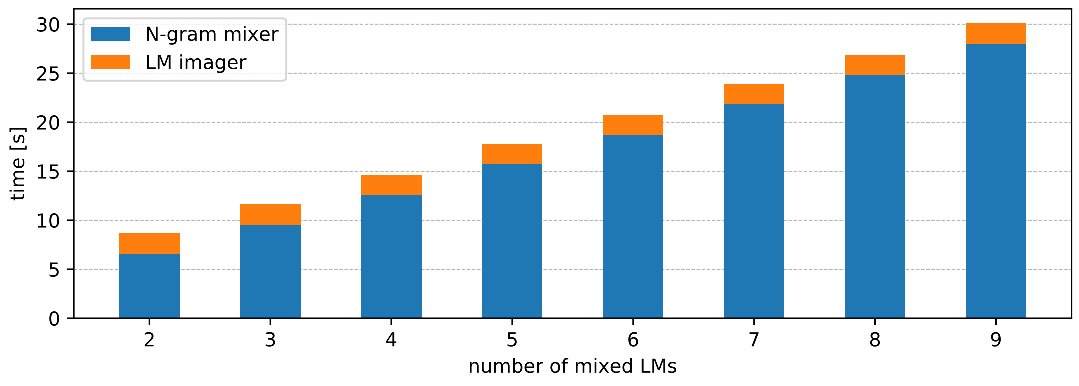

FIGURE 5.9: Processing time of LM mixer with increasing number of mixed LMs. The total time consists of mixing together N identical n-gram LMs (in this case LMs with vocabulary size of 500 thousand words) and converting the mix into a LM image suitable for the decoder in ASR system.

LM Imager is constant and the time consumed by *N-gram Mixer* scales roughly linearly with increasing N.

5.6.6 Type of Storage

In most cases, LMs are rather large data structures and it is not possible to store all adapted LMs in the RAM. Instead, we must store LMs on a hard drive. Therefore, the reading speed of the storage in use must be reasonable according to the size of LMs. A typical reading speed of current hard disk drives is about 200 MB/s (HDD) or 3 GB/s (SSD).

For example, when the size of a LM is 500 MB and the hosting machine is using HDD to store the data, the time delay caused by loading each adapted LM from the disk would be about 2.5 seconds. If we use larger LM, say 3 GB, the system would have to wait 15 seconds just to read the adapted LM from the disk, which can be unbearable delay in some tasks. However, the same LM would be loaded from the SSD disk in only 1 second. Apparently, when using large LMs, switching to SSD disk may lead to a significant speed up of LM adaptation.

Chapter 6

Experiments on TV News Domain

TV news is an extremely challenging domain for automatic speech recognition for two reasons: (1) each few-minutes-long report can be from a completely different domain and inside each report, speech styles are highly changeable (for example, the report starts with short introduction from the news studio, continues in some noisy exterior introducing different speakers and ends back in the studio making a live phone call with another speaker); (2) reports can contain very specific and rare content-bearing proper nouns crucial for understanding the content of the speech, which are usually missing in the vocabulary (e.g. places where some accident recently happened, names of so-far-publicly-unknown people and organizations etc.).

In this chapter, we applied topic-based LM adaptation algorithms on broadcast TV news and evaluated improvements in terms of WER. We also evaluated transcription of proper nouns, which is very important aspect of this domain.

6.1 Audio Data

To test topic-based LM adaptation in the news domain, we chose TV show *Události*, which is the main daily broadcast TV news show in Czech Republic. Each show was about 48 minutes long and contained about 23 individual reports related to many various topics. At each boundary of two consecutive reports, topics were usually changing, which makes these data suitable to test automatic topic-based LM adaptation.

To cover longer time period and various news themes, we selected one whole week from November 2013 (7 records) and 6 records from consecutive Mondays from March and April 2014. The format of all audio records was a standard PCM (*Pulse-Code Modulation*), 16 bits per sample, 1 channel and sampling rate 16 000 Hz.

All tested records were transcribed by human annotators and report-breaks were marked manually along with underlying topics for individual reports. In sum, our testing data consisted of 13 records, 10.3 hours of audio and 303 individual reports. Details about selected records and corresponding transcripts are summarized in table 6.1.

Date	**Duration**	**Reports**	**Words**	**Proper Nouns**
13.11.2013	0:47:35	17	6 481	501
14.11.2013	0:48:19	23	6 729	429
15.11.2013	0:47:45	26	6 348	338
16.11.2013	0:43:54	22	6 211	375
17.11.2013	0:43:18	19	5 816	389
18.11.2013	0:48:30	23	6 615	352
19.11.2013	0:47:59	21	6 887	363
10.03.2014	0:48:34	24	6 804	367
17.03.2014	0:47:29	21	6 343	386
24.03.2014	0:48:06	26	6 492	361
31.03.2014	0:47:41	27	6 810	368
07.04.2014	0:47:55	29	6 437	318
14.04.2014	0:48:24	25	6 376	352
total	10:15:31	303	84 349	4 899

Table 6.1: Test audio records from TV show "Události" along with duration and numbers of reports, words and proper nouns in transcript.

6.2 Text Data

The target domain was TV news, therefore we had to collect and prepare large amount of suitable text data close enough to this domain to train high-quality and robust LMs. On our department, we had developed a framework solution for mining, processing and storing large amounts of electronic texts for language modeling purposes (Švec et al., 2014). About six years ago, this system started to periodically import and process news articles from many Czech news servers. We had also supplemented the corpus with additional transcripts of selected TV and radio shows and a large amount of newspaper articles bought from a news agency. In sum, we had accumulated corpus with Czech news-related documents amounting to almost 1.3 billion tokens in 3.7 million articles and these numbers are still growing.

In this framework, we had implemented a sequence of tools that processed each document appropriately for language modeling purposes, namely text cleaning, tokenization, normalization, true-casing and vocabulary-based text replacements to unify distinct forms of the same words and multi-word expressions. After applying all these tools, we ended up with 8.6 GB of processed news-related Czech text prepared for language modeling.

To avoid misspelled words and other eccentricities to be recognized in the speech, we checked all tokens in our text database against a list of known and correctly-spelled words, and we marked all out-of-list tokens as unknown words, which reduced the vocabulary size from 4.4 to 1.2 million words.

6.2.1 LMs

Since we trained LMs from a lot of text data and we needed them to fit into the memory, all LMs in this chapter (if not stated otherwise) were trigram models with the minimal count of bigrams in text limited to 3 and the minimal count of trigrams limited to 6.

A general (topic-independent) LM trained from all available texts contained 1.2 million unigrams, 27 million bigrams and 20 million trigrams. The size of this LM (in ARPA format) on the disk was 1.3 GB. The size of adapted LMs was variable depending on particular topics clusters.

To clear the terminology, we are using following terms to address different LMs:

- *general LM* — large topic-independent LM trained from all available text documents; this LM can be prepared offline,
- *single-topic LMs* — LMs trained from documents related to only one topic; these LMs can be also prepared offline,
- *multi-topic LMs* — LMs interpolated from one or more single-topic LMs; these LMs are mixed on demand based on topics detected online in the speech,
- *adapted LMs* — LMs interpolated from multi-topic LM and general LM; these LMs are used to online adapt the LM in the decoder.

6.2.2 Topics

In order to do topic-based LM adaptation, we had to cluster our text data into some topics covering various areas from which news are reported. That means we had to identify the topics each piece of text belonged to. Some topic-related experiments with this dataset have been also published by the author of this thesis in (Lehečka and Švec, 2015) and (Skorkovská et al., 2011).

Some news servers provide a set of labels (or tags) along with each article. Based on presence or absence of such labels in the data, there are two basic approaches to topic identification: supervised text classification in which provided labels are fully exploited, and unsupervised topic models in which the model finds topic clusters itself without the need of any label. I was experimenting with both approaches in sections 6.5 (supervised) and 6.6 (unsupervised).

6.3 Performance Measures

In all experiments in this chapter, a standard word error rate (WER) was evaluated. However, WER perceives all word errors with equal importance. We believe there is a significant difference between minor deviations from the reference text, which the viewer would be able to consume without loss of meaning, and complete misrecognition of a word crucial to understanding the content of the speech.

In the case of news domain, we are not able to list all words which are crucial to understanding the content of the speech. However, we can approximate this list with all proper nouns[1] in the transcript. Indeed, personal names, geographic locations, companies etc. are much more important to be recognized correctly than some common words, which the viewer can often deduce from the context if misrecognized.

Hence, we are introducing additional performance measure more informative about errors in the news domain, called **Proper Noun Error Rate**, $PNER$ for short. Let the reference transcript be preprocessed by some true-casing tool[2] and aligned with a recognized word sequence. Proper noun error rate can then be computed as

$$PNER = \frac{S_{PN} + D_{PN} + I_{PN}}{N_{PN}} \cdot 100\,\%, \tag{6.1}$$

where S_{PN} is number of reference proper-nouns substitutions, D_{PN} is number of reference proper-nouns deletions, I_{PN} is number of reference proper-nouns insertions, and N_{PN} is the total number of proper nouns in the reference word sequence. In other words, $PNER$ is WER evaluated only for proper nouns in the reference transcripts.

[1]For simplicity, we considered all words starting with a capital letter as *proper nouns*.

[2]True-casing must include also decapitalization of sentence beginnings, otherwise all first words of sentences would be considered as proper nouns.

6.4 Baseline Performance

As a baseline system, we used online ASR system described in the section 5.2. We used only the general-LM decoder without any topic-based LM adaptation during the recognition, so the whole test data was transcribed using only one static general LM described in section 6.2.1. The measured performance was $WER = 16.95\%$ and $PNER = 17.41\%$.

6.5 Supervised Text Classification

Supervised approach to identify topics is a standard classification task requiring sufficient amount of annotated data to train a classifier. In our case, annotations were topic labels manually assigned to each article by journalists.

Since each news server uses different label-set and manual labeling suffers from so-called *inter-indexer inconsistency* phenomenon[3], these labels were not consistent across whole multi-source text database.

To avoid topic inconsistency, we decided to choose labeled data from only one server which we believe to be labeled thoroughly and consistently, train a topic identifier on top of these data and automatically assign labels for the rest of (differently labeled or unlabeled) sources. After that, all articles in our database should be labeled consistently. The amounts of our labeled and unlabeled data is summarized in table 6.2. Here, we considered all differently labeled sources as unlabeled data.

	Tokens	**Articles**	**Size**
	(in millions)	(in thousands)	(in MB)
labeled data	84	270	579
unlabeled data	1 203	3 391	8 034

Table 6.2: The amount of labeled and unlabeled text data.

6.5.1 Topic Tree

In our labeled data, there were 23.5 thousand distinct topics. We manually mapped the most frequent topics into a standardized IPTC news topic tree[4]. To avoid mistyped and low-frequent topics with only a little training data, we ignored topics assigned to less than 10 articles.

[3]Two human experts may relatively often disagree on how to label a text document.

[4]`http://cv.iptc.org/newscodes/mediatopic`

Our final topic tree consists of 577 nodes (labels) covering various news areas starting from very general top-level "root" nodes and ending with very specific "leaf" nodes. To get a picture about frequencies of particular topics in our labeled data, we generated a topic cloud into the figure 6.1. As can be seen, the most frequent topics in our labeled data are USA and sport-related topics.

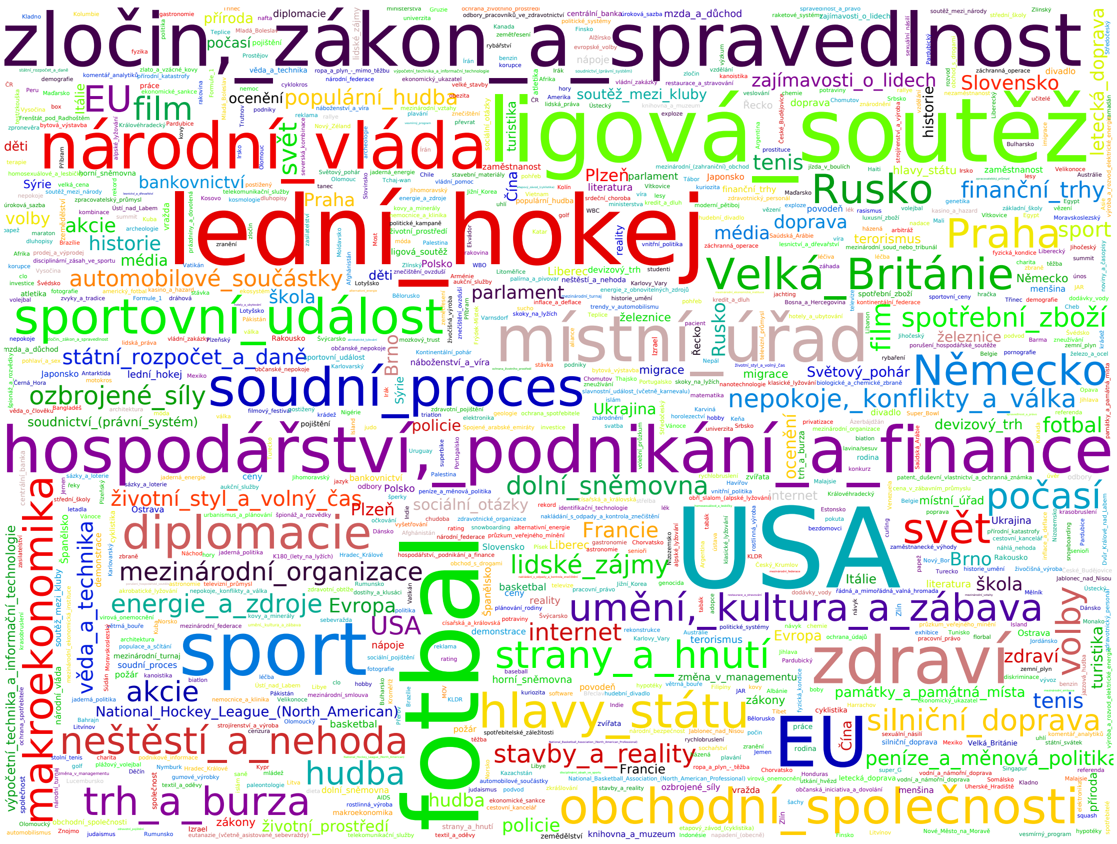

FIGURE 6.1: Topic cloud generated from labeled data (in Czech). The larger font a topic has, the more frequent it is in the data.

To get a picture about the hierarchical structure of our topic tree, an example of one fully expanded top-level topic (economy, business and finance) is depicted in figure 6.2.

Since assigning topics for unlabeled data is a non-trivial problem, especially in the multi-label case, we designed an experiment described in the section 6.5.2, in which we investigated how to correctly assign labels for unlabeled data with respect to a topic-based LM adaptation task.

6.5.2 Topic Clustering of Unlabeled Data

Correct distribution of text data into topic clusters from which single-topic LMs are trained, is very important subproblem in LM adaptation task. With increasing number

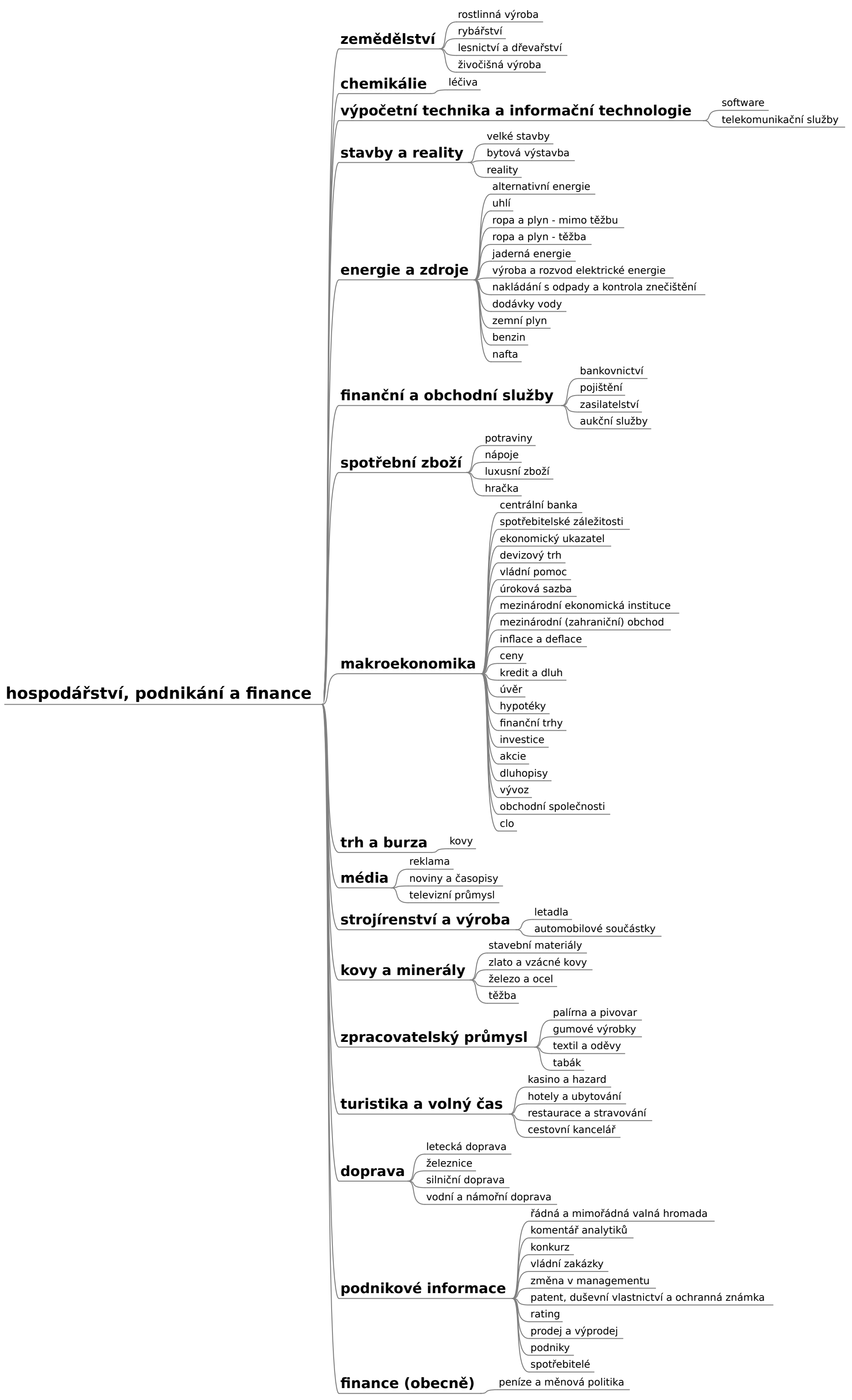

Figure 6.2: Example of fully expanded topic node "economy, business and finance" in our topic tree (in Czech).

of articles misclassified into wrong topics, the difference between topic-specific n-grams becomes less and less noticeable, single-topic LMs become more and more alike and the topic-based LM adaptation gets less and less useful. In this subsection, we were investigating how to correctly assign labels for unlabeled data with respect to a topic-based LM adaptation task.

TIs Preparation

From all labeled data, we trained various soft predictors and in combination with various thresholding strategies, we identified topics for unlabeled data, separated all (labeled and unlabeled) data into topic clusters, trained single-topic LMs, and used them in ASR experiments. We experimented with two topic granularities: (1) whole topic tree consisting of 577 topics and (2) only the top level of the topic tree consisting of 20 "root" topics (see figure 6.3).

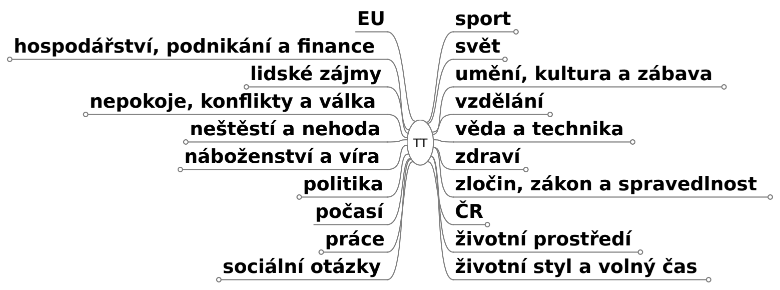

FIGURE 6.3: 20 top-level nodes in our topic tree (in Czech).

LMs Preparation

We followed the scheme depicted in figure 6.4 to process data and create adapted LMs suitable for underlying speech. We started with selecting all labeled data from our database, trained a topic identifier (TI) from them (we were experimenting with various TIs described in section 5.4) and used this TI to assign topics for unlabeled data. After that, we arranged the corpus into topic clusters by putting each article into one or more topic clusters based on the identifications from TI, or based on manually assigned labels in the case of labeled data. For each topic, we trained a *single-topic LM* from all texts belonging to the corresponding topic cluster. Also, we trained a *general LM* from all available text data. All these steps were done offline before the recognition experiments started.

With data prepared in this way, whenever topics in the recognized speech are changed, we can use these LMs and quickly build suitable *multi-topic LM* or, if there is only one

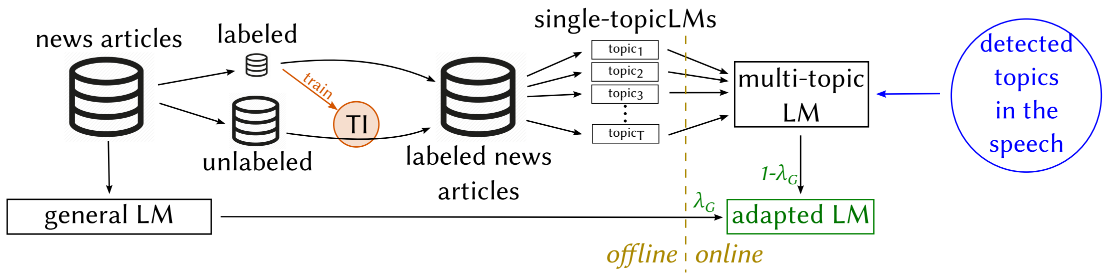

FIGURE 6.4: Scheme of topic-based data clustering in the news domain.

topic in the speech, single-topic LM can be used directly. The final *adapted LM* is then mixed together from general LM and multi-topic LM.

If single-topic LMs are trained with the same n-gram counts limits as general LM, each single-topic LM contains only a subset of general-LM n-grams. Therefore, both adapted and general LMs share the same vocabulary and set of n-grams, but with different probability distributions.

Experiments

To make this experiment simple and the results clear and comparable, we did not identify topics during recognition using the recognized data. Instead, we fixed all topic changes to predefined scenario designed for each record by human annotator. In this way, language models were adapting always in the same audio time to the same topics, therefore all changes in the ASR performance were caused solely by different clustering of unlabeled data into topics. The mixing weight of general LM in adapted LM was fixed to $\lambda_G = 0.5$, i.e. both component LMs (multi-topic and general) were equally important.

Results are summarized in figure 6.5. We evaluated total WER and $PNER$. We used abbreviations, classifiers and thresholding strategies listed in the description of the system in sections 5.4.2 and 5.4.3. Thresholds in strategies *FixedCut* and *RelCut* were optimized on held-out data[5]. As for the *RCut(k)* strategy, we experimented with $k = 1$ (i.e. each article belongs to exactly one topic) and $k = 3$ which is the label cardinality[6] of our labeled dataset. We can see from the graphs that different clustering of unlabeled data into topics and different topic granularity affects the performance of ASR system.

First, let's compare results for two different topic granularities: 577 topics (full topic tree) and only top-level 20 topics, i.e. put the right half of graphs against the left half. Experiments with 577 topics scored slightly better in 9 (out of 15) cases in terms of

[5]Randomly selected 5% of labeled data not used during the training of TI.
[6]Average number of labels of each document in the corpus.

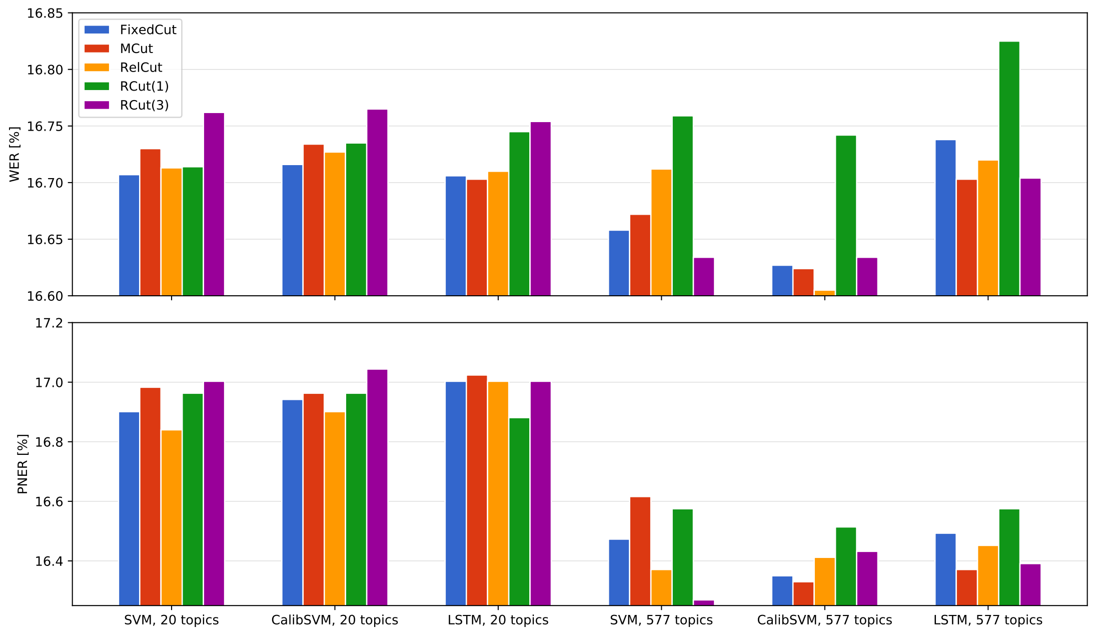

FIGURE 6.5: Results with different topic clustering of unlabeled data in the news domain. For 2 different topic granularities, 3 different classifiers and 5 thresholding strategies, WER and $PNER$ (Proper Noun Error Rate) was evaluated.

WER (0.23% relative WER reduction in average), however significantly better in terms of $PNER$. In all 15 cases, more proper nouns were recognized correctly when using 577 topics, scoring in average 0.52% absolute and 3.1% relative reduction of $PNER$ compared to experiments with 20 topics. We believe it is because smaller and more specific single-topic LMs can put more probability mass specifically to words (proper nouns) highly related to underlying topics.

When comparing topic classifiers *SVM*, *CalibSVM* and *LSTM*, we can see very similar results for experiments with 20 topics. However, in experiments with 577 topics, a small superiority of *CalibSVM* and a significant inferiority of *LSTM* classifier can be observed. We attribute poor results of LSTM classifier to the fact, that this classifier is looking only at the first 500 words of each article in order to identify topics, which is a consequence of re-arranging data into mini-batches to speed-up the training, whereas SVMs can see all words in the bag-of-words representation. Another big advantage of SVMs is the fast training. It took less than half an hour of CPU-time to train *SVM* and 1 day of GPU-time to train *LSTM*.

And finally, let us compare different thresholding strategies. One obvious result is, that strategy *RCut(1)* (which puts each text into exactly one topic cluster), is not suitable for clustering data in the news domain. We attribute it to a multi-topic character of news articles. However, we do not see any superior strategy from the results. Each tested strategy can be suitable for a different classifier.

We can sum these results into following observations:

1. Clustering texts into more branched topic tree is better, especially for proper nouns recognition.
2. Clustering with LSTM classifier is (sometimes surprisingly) worse than with SVMs.
3. It is worth to calibrate SVM's output to form a probability distribution, especially when using higher number of topics.
4. There is not one best thresholding strategy. We must select one with respect to the classifier in use.

The best WER was achieved when clustering data into 577 topics using *CalibSVM* with *RelCut* (16.61%) and the best $PNER$ was achieved using *SVM* with *RCut(3)* (16.27%). For further experiments in this section, we decided to use topic clusters emerged from ***CalibSVM*** **classifier** with ***RelCut*** **thresholding strategy** trained to distinguish between **577 topics**. In the experiments in this section, LM adaptation using these topic clusters performed best in terms of WER and reasonably well in terms of $PNER$. When compared to the baseline (no adaptation), we achieved 1.9% relative reduction of WER and 6.2% relative reduction of $PNER$ using these topic clusters.

However, it must be pointed out that these results are not real-word results, because they were affected by external knowledge, which had been put into the recognition process in order to fix the topic changes and thus focus only on differences caused by various topic clusters. That is why these results are better than those achieved in following sections and therefore they will not be shown in the summary results in section 6.7.

6.5.3 Online Topic Identification

Correct online topic identification of the current speech is a very important part of online topic-based LM adaptation task, because by analyzing the trend in the recent sequence of identified topics, the decision about a topic change (followed by LM adaptation) is made.

In the previous subsection, we were experimenting with several topic identifiers to cluster our (mostly-unlabeled) text corpus in order to train robust adapted LMs for news domain. In this subsection, we were experimenting with the same topic identifiers, but this time, we were using them during online recognition in order to dynamically identify current topics in the speech. This information was then used to adapt the LM on the fly.

6.5.3.1 Classifier and Thresholding Strategy

First, we used the same classifiers and thresholding strategies as in section 6.5.2 to identify topics online from the recently recognized words while keeping all other parameters on default values.

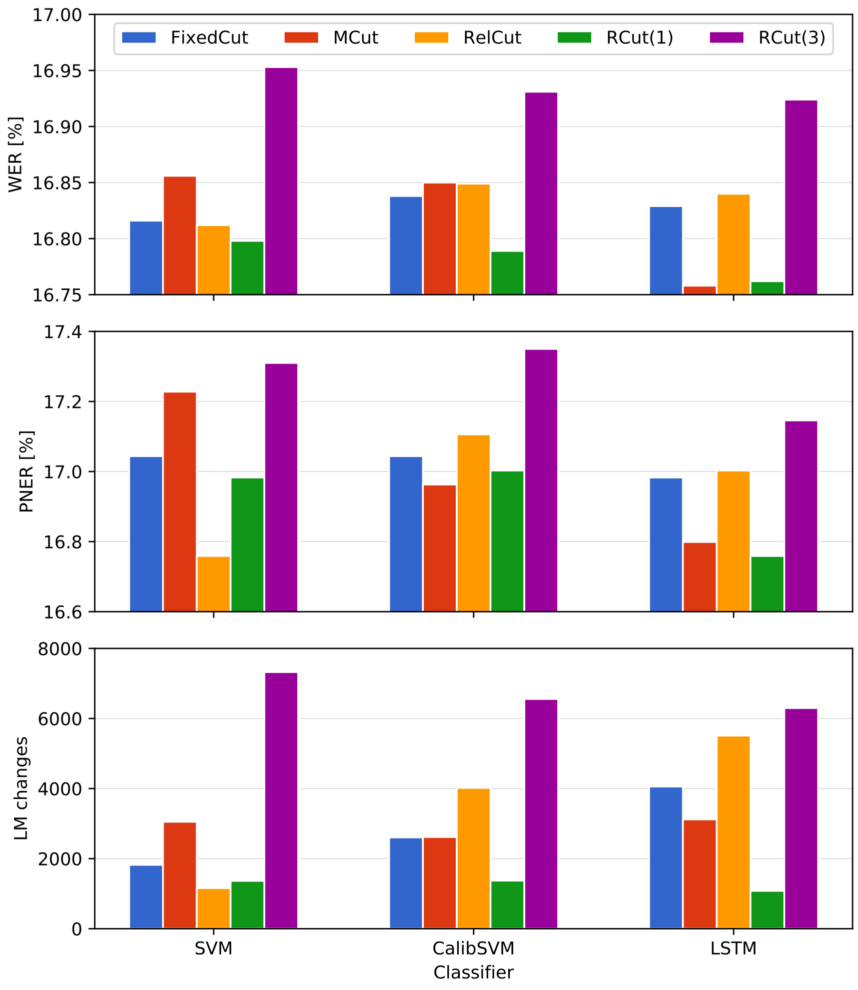

Figure 6.6: Results with different online topic identifiers in the news domain. For 3 different classifiers and 5 thresholding strategies, WER, $PNER$ (Proper Noun Error Rate) and the total number of LM changes applied during the recognition was evaluated.

Results are summarized in figure 6.6. Along with WER and $PNER$, we evaluated also the total number of LM changes, i.e. how many times the LM was adapted during the recognition[7]. From the graphs, we can see that the best WER was achieved using LSTM classifier with MCut and RCut(1) thresholding strategy. We attribute the good performance of LSTMs to the fact, that the system is focusing only on a short recently-recognized text, and since LSTM contains max-pooling layer over time, a strong reaction of one single neuron when seeing a topic-related keyword can result in accurate predictions even if the text is very short. SVMs, on the other hand, were trained to identify the topics from full articles encoded into tf-idf, and thus the prediction may rely on more complex term statistics which are missing in very short text samples.

From the bottom subfigure, we can see that predictions from *RCut(1)* and in case of SVMs also *RelCut*, are rather smooth during the recognition, i.e. they identify the same topics for longer time period and thus, less number of adaptations is needed. Predictions from other strategies are highly changeable during the recognition, therefore more LM changes is needed. One may expect that more LM changes would lead to more accurate recognition, but our results indicate opposite: too many LM adaptations leads to deterioration of the performance. For example, very poor results were achieved using RCut(3), which identifies always 3 topics even if they are not present in the speech, causing too many LM adaptations to off-topic domains.

Number of LM changes is also important from the computational perspective. Each LM change means the adapted LM had to be mixed during the recognition, causing significant delay in the *offline* mode or intensive computations in a parallel background process in the case of *online* mode. If we know we will need only reasonable number of distinct LMs, we can prepare these LMs offline (before the experiment starts) and there would be no need for mixing LMs online during the recognition. However, the number of possible topic combinations grows exponentially with the number of identified topics. For strategies *FixedCut*, *MCut* and *RelCut*, the number of possible adapted LMs is immense, since any number of topics can be identified at any time. To prepare offline all topic mixtures for strategy *RCut(3)*, there would be $\binom{577}{3} \approx 32$ million distinct adapted LMs, which is still hard to store. The only thresholding strategy where preparing LMs offline makes sense, is *RCut(1)*, where only 577 LMs must be prepared.

We can sum these results into following observations:

1. It is not true that the more adaptations of LM is applied, the better performance is achieved.

2. LSTM classifiers are more suitable for identifying topics from shorter texts.

[7]For example, 1000 LM changes means the topics (and the LM) had been changing in average every 37 seconds during the recognition (see data statistics in table 6.1). The total number of topic changes marked by annotators (used for experiments in section 6.5.2) was 303.

3. The very simple thresholding strategy *RCut(1)* evinced good results and moreover, it has one great advantage: the number of possible distinct adapted LMs is equal to the number of topics and thus all LMs can be prepared offline, leading to a fast online recognition without any delay caused by mixing LMs.

For further experiments in this section, we decided to use **LSTM** with ***RCut(1)*** to identify topics online, because it achieved the best $PNER$, second best WER and all LMs can be prepared offline to speed up the recognition.

By using thresholding strategy *RCut(1)*, we assume there is only one topic in the speech at each time, i.e. we are performing *single-topic LM adaptation* while excluding any possibility of adapt the LM to a combination of more topics. This assumption may seem too rigorous, but our results give us strong evidence that it is the best we can do.

6.5.3.2 Cropping of Hypotheses

So far, all parameters influencing the cropping of hypotheses before predicting the topics scores were kept on default values, i.e. at each time step, we took the one-best partially recognized hypothesis (sequence of words), threw last 2 words away as there is usually high uncertainty about them from the recognizer, and sent last 50 recognized words from the rest into the TI to predict the topics scores. In our audio data, there was 2.28 words per second in average, so 50 words means we are analyzing about 22-seconds-long window. In this subsection, we were investigating how changes in this setting influence the online LM adaptation.

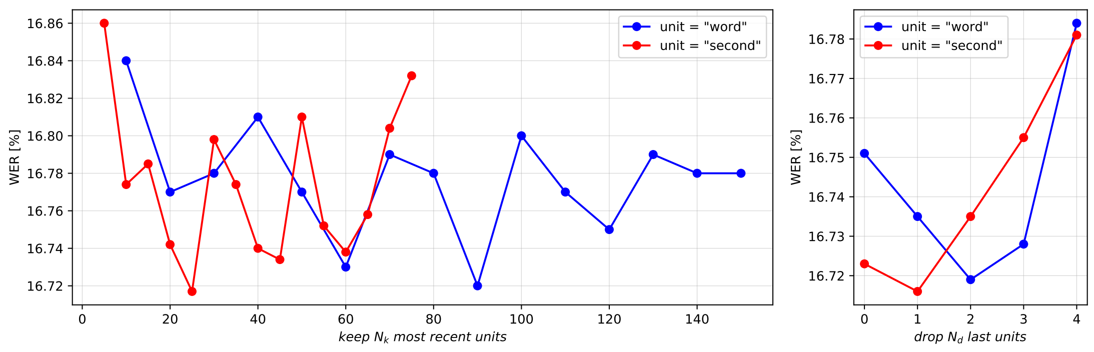

FIGURE 6.7: Results with different cropping of hypotheses in the news domain. For two unit types and variable value of *keep* N_k *most recent units* (left subfigure) and *drop* N_d *last units* (right subfigure), WER was evaluated.

Results are plotted in the figure 6.7. Results for words and seconds are roughly reflecting the previously mentioned ratio in our data: 2.28 words per second in average, so the graphs for seconds are ”squeezed” proportionally.

From the right subfigure, we can see that it is a good practice to throw away the last two words or the last second of recognized hypothesis. We believe it is because of the lack of context at the very end of a partial hypothesis and also because the hypothesis can be reported at the moment, when only a part of some longer word has been processed so far.

In the left subfigure, we do not see any smooth trend. There are more local minima, but we can conclude, that good values for parameter *keep* N_k *most recent units* are between 60 and 120 words or between 25 and 60 seconds alternatively. We observed the best result when identifying topics from hypotheses cropped to the last 25 seconds while throwing the last one second away ($WER = 16.717\%$). We used this setting in the following experiments.

Since the differences in WER in figure 6.7 are rather modest, we can state that fine-tuning of the cropping parameters has only a minor influence on topic-based LM adaptation.

6.5.3.3 Weighting of Recently Recognized History

In the next experiment, we were testing various weighting functions listed in section 5.4.1. From results plotted in the figure 6.8, we can conclude, that giving higher weights to more recent words did not bring any improvement. We attribute these results to the fact, that by weighting recent words, the norm of corresponding word vectors is being scaled, and thus the LSTM classifier must identify topics from modified vectors never seen during the training.

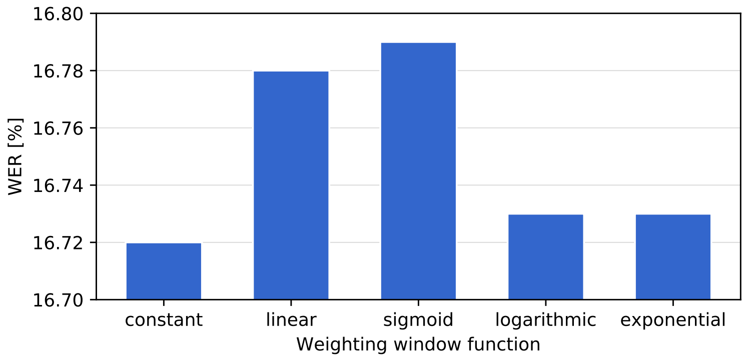

FIGURE 6.8: Results with different weighting functions of recognized words in the news domain.

Based on this experiment, we kept the default value in the system, which is *constant* weighting, i.e. no weighting at all, so all words in the cropped hypothesis are equally important.

6.5.3.4 From Words to Lattices

We tried also to move from 1-best hypothesis to a lattice, specifically to a word confusion network (known as a "sausage"). One may expect that since there are many alternative hypotheses encapsulated in the lattice, the identified topics should be more accurate. However, the best result we were able to achieve using lattices was $WER = 16.761\%$, which is slightly worse than the result with the same setting using 1-best hypothesis ($WER = 16.717\%$).

An explanation for this result can be that the LSTM classifier was trained to process fine-tuned invariable word vectors and when it gets modified word vectors (e.g. averaged or scaled) on the input, the topic-related neurons might not get activated.

6.5.4 Topic-change Detection

In the next experiment, we were testing various values of *steadiness* in the topic-change detector (see section 5.5). Steadiness defines, how many seconds into the history any topic must have been continuously identified to be marked as a steady topic in the speech. Only steady topics are concerned when adapting the LM.

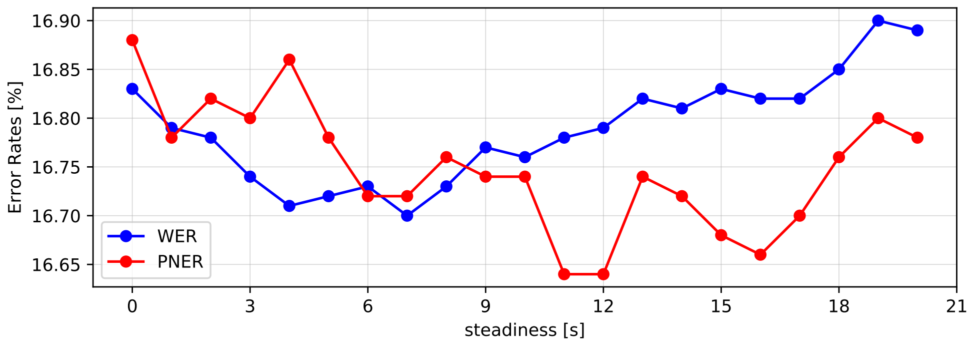

FIGURE 6.9: Results with different steadiness of the topic-change detector in the news domain. For varying values of steadiness, WER and $PNER$ (Proper Noun Error Rate) was evaluated.

Results are shown in the figure 6.9. As for WER, we can observe rough U-shape trend with a minimum at 7 seconds of steadiness ($WER = 16.7\%$). As for recognition of proper nouns ($PNER$), the best values were achieved at 11 and 12 seconds of steadiness.

Higher steadiness has positive influence on $PNER$, because the system is adapting the LM only if it is very certain the topics have changed. However, increasing steadiness delays the LM adaptation behind the real topic change resulting in recognizing the beginning of the speech with LM adapted to the previous topics and thus to a deterioration of overall WER. For the following experiments, we fixed the steadiness to 7.

6.5.5 General LM weights

In the next experiment, we were investigating the influence of the general LM weight on the online topic-based LM adaptation. We tried varying values of the general LM weight, including both extremes: $\lambda_G = 0$ (i.e. the adapted LM consists only from multi-topic LM) and $\lambda_G = 1$ (i.e. the adapted LM consists only from general LM).

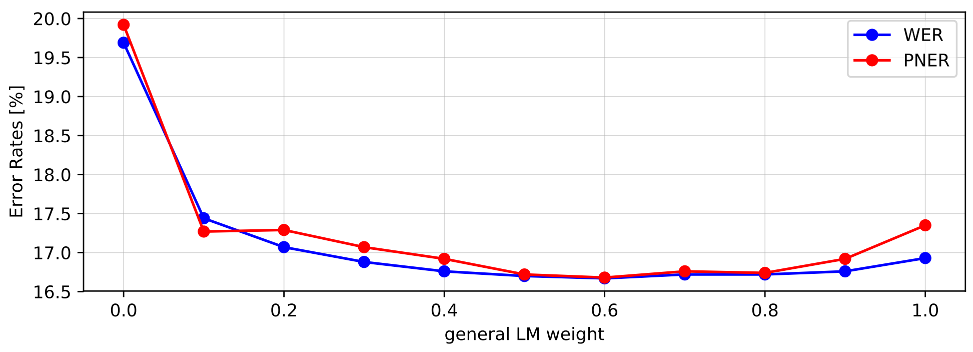

FIGURE 6.10: Results with different weight of general LM in the news domain.

From results plotted in the figure 6.10, we can see, that WER and $PNER$ are in high correlation achieving the best result at $\lambda_G = 0.6$ ($WER = 16.668\%$), which stresses the positive contribution of both multi-topic and general LM. Note that the rightmost point ($\lambda_G = 1$) corresponds to the baseline performance as adaptation with pure general LM is effectively no topic adaptation at all. Another interesting point is, that very poor results were achieved for $\lambda_G = 0$ which is clearly a consequence of higher OOV-rate and the lack of general-language statistics in a pure multi-topic LM.

These results are in strong agreement with our preliminary experiment in (Lehečka, 2016). For the following experiments, we fixed the general LM weight to $\lambda_G = 0.6$.

6.5.6 Adding New Topic-related N-grams

In this last experiment concerning news domain with supervised topic identification, we relaxed the constraints from section 6.2.1 for minimal counts of n-grams in the text corpus when training LMs from a text. Let $N_G = (N_{G1}, N_{G2}, N_{G3})$ be the minimal counts of unigrams, bigrams and trigrams in the text corpus when training a general LM, and similarly, let $N_T = (N_{T1}, N_{T2}, N_{T3})$ be the minimal counts of unigrams, bigrams and trigrams included in a single-topic LM. So far, all LMs were trained with $N_G = N_T = (1, 3, 6)$. In this experiment, we trained the general LM additionally with $N_G = (1, 2, 3)$ and $N_G = (1, 1, 2)$, and single-topic LMs with $N_T = (1, 2, 3)$, $N_T = (1, 1, 2)$ and $N_T = (1, 1, 1)$. Details about LMs with different N_T and N_G are listed in table 6.3.

	1-grams	**2-grams**	**3-grams**	**sum n-grams**	**size [MB]**
average single-topic LMs:					
$N_T = (1, 3, 6)$	87 511	164 357	50 765	302 633	7
$N_T = (1, 2, 3)$	87 511	313 815	162 521	563 847	14
$N_T = (1, 1, 2)$	87 511	1 400 700	410 532	1 908 743	48
$N_T = (1, 1, 1)$	87 511	1 400 700	3 126 964	4 615 175	141
general LMs:					
$N_G = (1, 3, 6)$	1 198 175	26 995 001	20 208 434	48 401 610	1 335
$N_G = (1, 2, 3)$	1 198 175	46 132 589	55 007 125	102 337 889	2 954
$N_G = (1, 1, 2)$	1 198 175	148 974 703	117 139 957	267 312 835	7 831

Table 6.3: Details about LMs with different minimal n-gram counts in the news domain. For different single-topic LMs (4 different limits of N_T) and different general LMs (3 different limits of N_G), we show numbers of 1-grams, 2-grams, 3-grams, all n-grams (sum of 1-, 2- and 3-grams), and the size of the LM on the disk (in ARPA format). Since single-topic LMs are different for each topic, we show the average values over all topics.

Since the adapted LM is a mixture of general LM and single-topic LMs and each single-topic LM was trained from a subset of the whole text corpus (used to train the general LM), the limits of n-gram counts can influence the adaptation in following two ways:

1. If $N_{Tn} < N_{Gn}$, $n \in \{1, 2, 3\}$, the adaptation can fetch also new topic-related n-grams into the recognizer, which are not present in the general LM due to a low overall count in the text corpus.

2. On the other hand, if $N_{Tn} \geq N_{Gn}$, $n \in \{1, 2, 3\}$, the adaptation does not bring any new n-gram into the recognizer, but it assigns more probability mass to the topic-related n-grams and thus accentuate the topics in the adapted LM.

Results of the experiment are shown in the figure 6.11. Clearly, enlarging the general LM (i.e. lowering the limits in N_G) as well as adding more topic-related n-grams during LM adaptation (i.e. lowering the limits in N_T) has a positive impact on WER. However, we do not see the same trend in terms of $PNER$. Comparing with the baseline, there

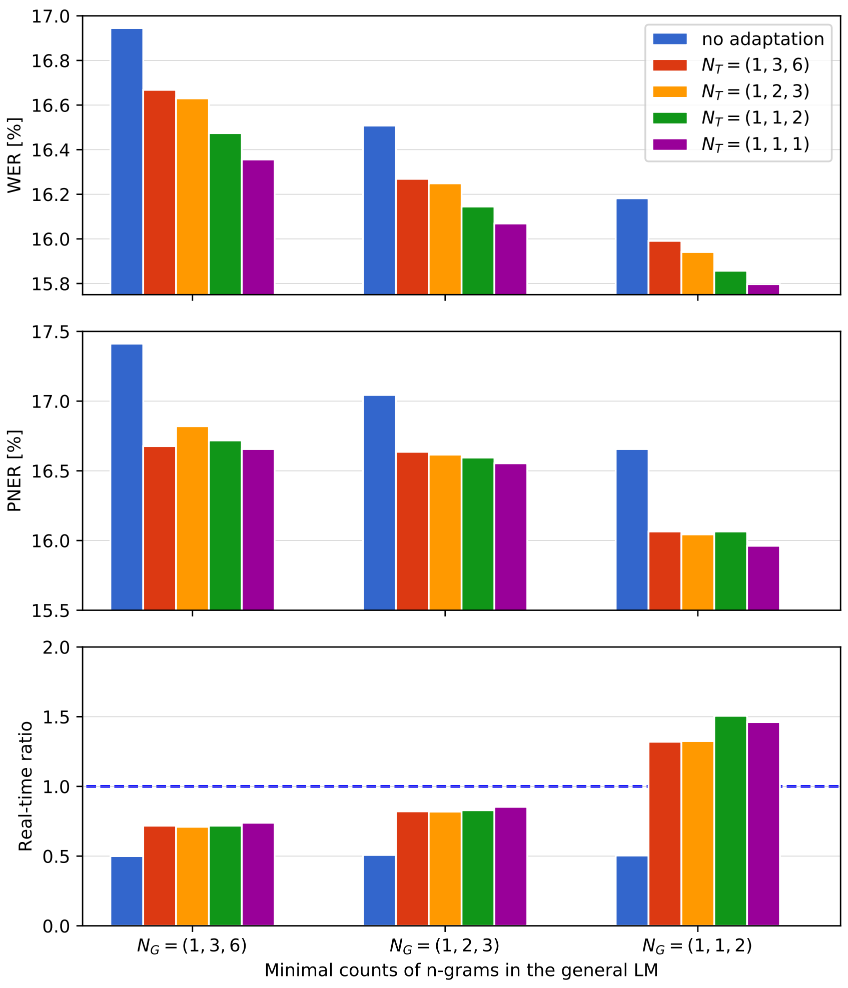

FIGURE 6.11: Results with different n-grams count limitations in the news domain. For 3 different minimal counts of n-grams in the general LM (N_G) and 4 different minimal counts of n-grams in single-topic LMs (N_T), WER, $PNER$ and the real time ratio on Intel Core i7-7800X machine was evaluated. For the purpose of comparing, the values for the baseline system with all three settings of N_G are shown as well (the *no adaptation* bars).

is a significant improvement in $PNER$ when using the smallest tested single-topic LMs ($N_T = (1, 3, 6)$) for topic-based LM adaptation, but adding more topic-related n-grams brings only a minor further improvements in terms of $PNER$. An explanation for this result can be that the dominant portion of n-grams containing topic-related proper nouns has high counts in the text corpus and among less-frequent topic-related n-grams, there is only a modest amount of proper nouns. In other words, the dominant portion of less-frequent topic-related n-grams contains rather common words than proper nouns.

Since enlarging LMs in real-time decoding is limited by computing power and memory of a hosting machine, we investigated also the real-time ratio (RTR) of all tested settings (the bottom graph in the figure 6.11). RTR defines the ratio between processing time needed to recognize the speech and the duration of audio signal. Experiments with $RTR < 1$ were faster than real time on the tested machine (Intel Core i7-7800X, used all 6 cores), which means they could be used for online speech recognition. Experiments with $RTR > 1$ were too slow to be used online.

From the bottom graph in the figure 6.11, we can see that RTR of the baseline system without any LM adaptation (the blue bars) scales very well with the size of the general LM. It is because in our system, the size of the vocabulary has the main influence on the speed of the recognition, and the size of the vocabulary (i.e. the number of 1-grams) stays the same for all settings of N_G. The number of involved bigrams and trigrams does not influence the speed so much. Lower number of bigrams and trigrams causes slightly faster decoding but at the same time, it increases the uncertainty in the decoder, which again slows the computation down.

The RTR of systems with LM adaptation is higher, because there are two parallel decoders running at the same time and because the topic identifier is monitoring the recognized partial results on the fly. However, the large differences in RTR between systems with $N_G = (1, 2, 3)$ and $N_G = (1, 1, 2)$ were caused mainly by slow loading of LMs into the decoder. Adapted LMs mixed from general LM with $N_G = (1, 1, 2)$ were much larger (see the table 6.3). One interesting result here is that the RTR stays almost the same when lowering the limits in N_T, i.e. when adding more topic-related n-grams, which again confirms that the number of involved n-grams does not influence the speed too much.

We can conclude, that including more (ideally all) topic-related n-grams into the single-topic LM is primarily beneficial. In our experiments, it caused a significant improvements in terms of WER and a minor improvements in terms of $PNER$ with only a subtle slow-down of the recognition speed. The best result we achieved was $WER = 15.797\%$, but with $RTR = 1.46$. The best result with $RTR < 1$ was $WER = 16.069\%$ ($RTR = 0.85$). However, to avoid slow data loading and large LM mixing in the real-world applications and also to keep some spare computational power of the computer to withstand peak loads, we would most likely use rather system with $N_G = (1, 3, 6)$, where we achieved $WER = 16.356\%$ (3.5% relative improvement when compared with the non-adaptable system) at the speed of $RTR = 0.74$.

6.6 Unsupervised Topic Models

Apart from supervised text classifiers used in this chapter so far, we experimented also with unsupervised topic models, namely with LDA (see section 3.2.4). In order to use such model, we had to forget all labels (both manually and automatically assigned) in our text database, and let the model find suitable topic clusters itself from the scratch (i.e. from the plain text).

In opposite to supervised learning, where each topic has a clear human-readable label, topics emerged from training the LDA model are represented only by a probability distribution over the vocabulary. Usually, LDA topics are presented to users as lists of N words with the highest probability.

Although automatic evaluation and comparison of LDA models can be arguable (Lau et al., 2014), we can evaluate the whole online ASR system when different LDA models are used inside, and compare WER of recognized texts. In this way, we can efficiently compare different LDA models with respect to topic-based LM adaptation task without explicitly measuring topics coherences or some other metrics.

Since the prediction from the LDA model is a probability distribution (i.e. a soft prediction), we can use the same thresholding strategies as in the case of supervised classifiers in the section 6.5.3.1. Specifically, we used strategies *FixedCut(0.5)*, *MCut*, *RCut(1)* and *RelCut(0.5)* for both text data clustering and topic identification during the recognition. For details about used thresholding strategies, see section 5.4.3.

6.6.1 Adapted LMs Preparation

We followed the scheme depicted in figure 6.12 to process data and create adapted LMs suitable for underlying speech. It is very similar scheme to the one used for supervised classifiers in the section 6.5, but instead of using labeled data to train the classifier, we trained the model in fully unsupervised way in this scheme. We used LDA model trained from all available text data we had with the vocabulary limited to 1.2 million words as described in the section 6.2.

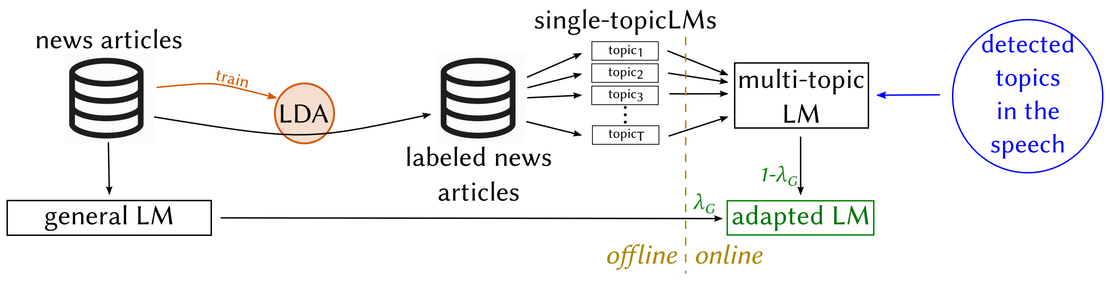

FIGURE 6.12: Scheme of LDA-based data clustering in the news domain.

6.6.2 Number of Topics

The key parameter when designing an LDA model is the number of topics the model can distinguish among. Figure 6.13 shows results of experiments with varying number of LDA topics. In each experiment, we used the same LDA model for both LDA-based text data clustering (and thus to prepare adapted LMs) and LDA-based topics scores prediction during the recognition. All other system parameters were fixed on default values.

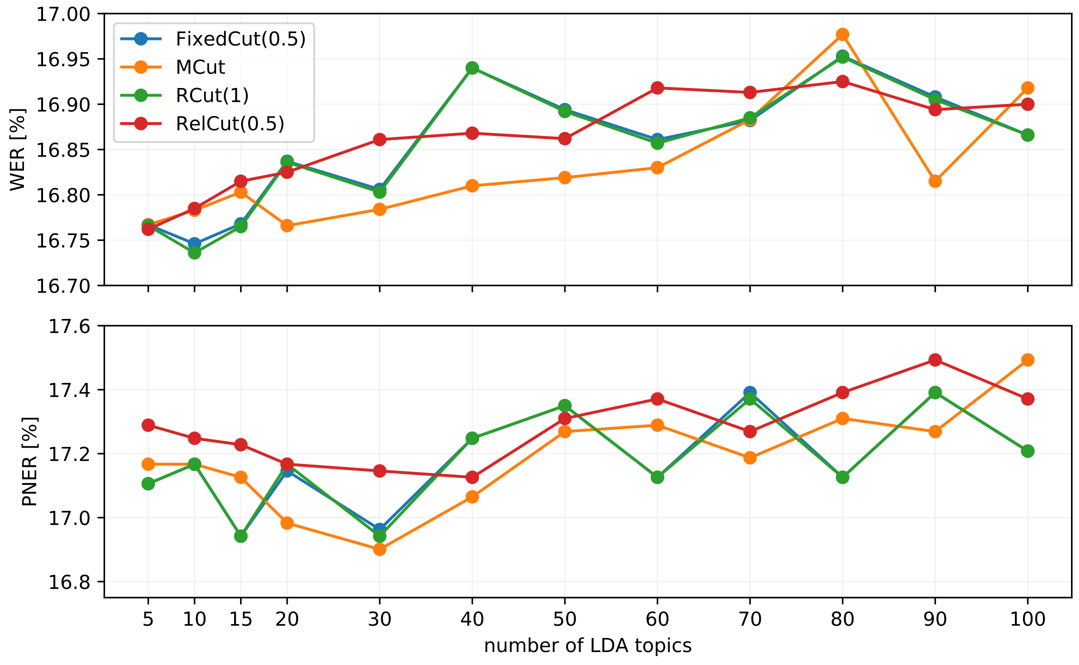

FIGURE 6.13: Results with different number of topics in the LDA model. For 4 different thresholding strategies and varying number of LDA topics, WER and $PNER$ was evaluated.

As can be seen from figure 6.13, results with thresholding strategies *FixedCut(0.5)* and *RCut(1)* are almost the same. It is because in most of soft predictions from the LDA model, there was always one topic with the probability higher than 0.5, therefore considering only topics with probability $p > 0.5$ and considering only 1-best topic resulted in the same hard prediction.

The best WER was achieved with 10 latent topics and *RCut(1)* strategy ($WER =$ 16.736%), while the best $PNER$ with 30 latent topics and *MCut* strategy ($WER =$ 16.901%). Further increasing the number of LDA topics didn't lead to any improvement. Comparing with the baseline, the system with 10 LDA topics scored 1.23% relative reduction in terms of WER and 1.4% relative reduction in terms of $PNER$.

To have a picture of how the LDA model covers important news categories, we draw a word cloud for each topic. All 10 word clouds are shown in the figure 6.14. Clearly, we can see that the LDA model learned how to distinguish among the major categories appearing in the news domain. There are obvious topics about: history and culture (#2), children and schools (#3), money and companies (#4), police and justice (#5), foreign affairs (#6), politics and political parties (#8), traffic and weather (#9) and sports (#10). The topic #1 seems to cover documents containing a lot of numbers (e.g. sport summaries, time harmonograms, various tables etc.). All documents out of previously mentioned topics fall into the topic #7, where we cannot see any dominant topic-related keyword.

6.6.3 Fine-Tuning of Parameters

For the next experiment, we took the best-scoring model from the previous section (LDA with 10 latent topics and *RCut(1)* thresholding strategy) and ran a grid search over system parameters analogous with the fine-tuning of parameters in the system with supervised topic identification (sections 6.5.3.2 - 6.5.5). Since the behavior of parameters was very similar to the experiments with supervised TI, we are only summarizing the results in this short section without plotting detailed graphs.

We achieved the best result when cropping each partial hypothesis to the last 25 seconds excluding the last one second, no weighting of the recognized words, steadiness set to 8 seconds and the weight of general LM set to $\lambda_G = 0.4$. Using this setting, the system achieved $WER = 16.683\%$ (1.55% relative reduction over the baseline system).

We also experimented with enlarging the amount of topic-related n-grams used to adapt the LM analogous to the section 6.5.6. When keeping the general LM on the default size ($N_G = (1, 3, 6)$) and adding all topic-related n-grams into the single-topic LMs (i.e. $N_T = (1, 1, 1)$), we achieved $WER = 16.368\%$ (3.41% relative reduction over the baseline system). The speed of the system was $RTR = 0.7$.

6.7 Results Summary

The most important results achieved in this chapter concerning topic-based LM adaptation in the news domain are summarized in table 6.4. We are showing the baseline results, in which no LM adaptation was incorporated, and two versions of topic-based LM adaptation:

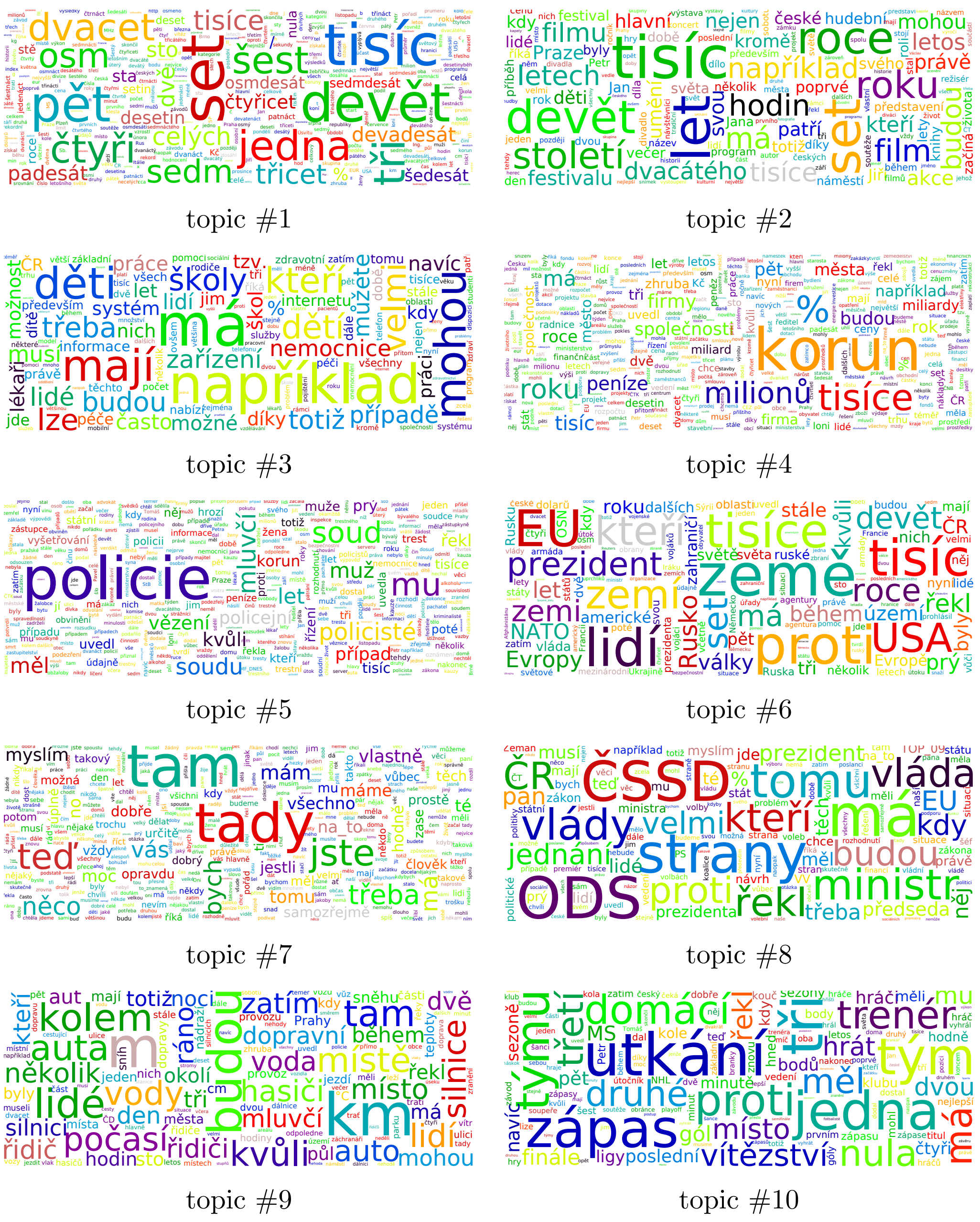

FIGURE 6.14: Word clouds for individual topics in trained LDA model (in Czech). The larger font a word has, the higher probability it has in the particular latent topic.

- LM adaptation with $N_T = (1, 3, 6)$ uses fixed set of the most frequent n-grams during whole recognition of the speech. The principle of this LM adaptation is based on dynamic increasing the probabilities of topic-related n-grams at the expense of out-of-topic n-grams.

- LM adaptation with $N_T = (1, 1, 1)$ uses all topic-related n-grams we have in data, therefore the size of LM is changeable and depends on the amount of topic-related data. The principle of this LM adaptation is based on dynamic increasing the probabilities of topic-related n-grams existing in the general LM and moreover, adding also less-frequent topic-related n-grams into the adapted LM.

For each version of LM adaptation, we are showing the best results achieved with both unsupervised and supervised topic identification methods. Also, we are showing relative reduction of error rates over the baseline system and real-time ratios.

	TI	**WER** [%]	**PNER** [%]	**RTR**
general LM (baseline)	-	16.95	17.41	0.50
adapted LM, $N_T = (1, 3, 6)$	U	16.68 (-1.55%)	17.02 (-2.23%)	0.66
	S	16.67 (-1.63%)	16.68 (-4.22%)	0.72
adapted LM, $N_T = (1, 1, 1)$	U	16.37 (-3.41%)	16.88 (-3.05%)	0.70
	S	16.36 (-3.48%)	16.66 (-4.34%)	0.74

Table 6.4: Results summary for the news domain. For two versions of LM adaptation and two topic identification (TI) methods: unsupervised (U) and supervised (S), we are showing the best WER and $PNER$ we achieved and corresponding real-time ratios (RTR) on the tested machine (Intel Core i7-7800X, used all 6 cores).

From the table 6.4, we can see that using supervised TI leads to better results than when using unsupervised TI, especially for the recognition of topic-related proper nouns ($PNER$). We can also see that using all available topic-related n-grams during LM adaptation (i.e. $N_T = (1, 1, 1)$) leads to significantly better WER with only a minor slow down of the system.

Chapter 7

Experiments on TV Sports Domain

TV sports is another example of an extremely challenging domain for automatic speech recognition. If the TV show is transmitted from a single-sport event (e.g. live broadcast of a football match), the key problem for ASR is to recognize names of featuring sportsmen correctly. Usually, the major portion of the actual roster are OOV words missing in the training data, therefore they must be added into the language model from some external source, and in the case of inflected languages, all inflected forms must be added too. Also, correct phonetic transcription of all inflected forms of names must be defined in the lexicon (in the case of regular pronunciation, it can be generated automatically). Single-sport TV shows can be transcribed using one static language model trained specifically for the target sport and extended with actual roster. This approach has been used e.g. for real-time closed captioning of TV ice-hockey commentaries (Hoidekr et al., 2006).

Much more complicated situation occurs when the TV show is a summary of a more complex multi-sport event like Olympic Games. In such shows, previously mentioned problems are especially significant, because the roster is very large (thousands of names) and there are sportsmen[1] from all countries in the world with sometimes very exotic pronunciation of their names.

Moreover, the sport and thus the style of the speech and used vocabulary are highly changeable within each episode. For example, the episode starts live from the Olympic studio summing up results from an archery tournament, then it continues with a recorded

[1]To clear the terminology, we will use the term *sportsmen* to address all men and women participating in any sport at the event, and the term *athletes* to address all men and women participating only in the athletics disciplines.

footage of the winner's shots followed by simultaneously translated interview, medal ceremony and then back to the studio, where the speech switches to another report about completely different sport, e.g. judo. Usually, sportsmen are highly related to only one sport and there is only a little chance that their names will occur when speaking about different sport (e.g. the probability that some judoka's name will occur when talking about archery is very low).

All issues mentioned above make live TV sports summaries suitable domain to employ online topic-based language model adaptation.

7.1 Audio Data

To test topic-based LM adaptation in the sports domain, we chose TV show *Velký přehled*, which is a daily-based summary from Olympic Games broadcast by Czech Television during Olympic Games. The language of the show is Czech. As for the venue, we chose Summer Olympic Games held from 8 to 24 August 2008 in Beijing, China.

Date	Duration	Reports	Words	Names
11.8.2008	0:44:58	14	5 442	308
13.8.2008	0:57:10	14	6 886	350
14.8.2008	0:53:33	24	6 685	387
15.8.2008	0:59:56	20	7 138	482
16.8.2008	0:42:53	7	5 618	406
17.8.2008	1:18:08	21	9 231	499
18.8.2008	1:03:11	21	7 849	373
19.8.2008	1:32:18	25	11 630	733
20.8.2008	0:51:41	10	6 517	296
21.8.2008	1:11:05	15	8 817	526
22.8.2008	0:57:47	16	7 477	426
23.8.2008	0:58:30	16	7 189	464
24.8.2008	0:23:50	10	2 119	74
total	12:34:59	213	92 598	5 324

Table 7.1: Test audio records from sports summary TV shows along with duration and numbers of reports, words and sportsmen names in transcript.

We obtained audio records from almost every day the show had been broadcast. All records were transcribed and manually annotated by human annotators. Details about our test audio data are summarized in the table 7.1 along with numbers of words, names of sportsmen and individual reports. In sum, our audio data consists of more than 12.5 hours and in the reference transcripts, there are 92.6 thousand words, from which 5.3 thousand words are names of sportsmen featuring at the Olympics. Into this names-count, we included only sportsmen present in the official roster from Summer Olympic Games in Beijing 2008, all other personal names were considered as common words.

We denote a continuous speech about exactly one sport as one *report* in the table 7.1. Whenever the sport was changed in the speech, the annotator labeled the position as a report-break (i.e. the change of a topic). An ASR system with an optimal topic-based LM adaptation should change language models exactly at these positions.

Since the background noise in the sports domain is especially interfering, we let a shadow speaker[2] to re-speak all tested audio records, which is a standard technique to suppress all negative acoustic events during the real-time closed captioning. For comparison, we had also evaluated the system when recognizing the original audio (i.e. the signal broadcast to TV viewers) in the section 7.10. Of course, these results are much worse due to a varying style of speech and a lot of background noise in the signal.

The format of all audio records is a standard PCM (*Pulse-Code Modulation*), 16 bits per sample, 1 channel and sampling rate 16 000 Hz in the case of original audio and 22 050 Hz in the case of the shadow speaker.

For acoustic modeling we used three-state HMMs with output probabilities modeled by a Deep Neural Network (DNN). We used a general-purpose acoustic model to recognize original audio and for the shadow speaker, we used a special AM trained specifically for particular speaker.

Although we are experimenting with audio signals recorded in the past, we will simulate the conditions of a real-time recognition in all our experiments, therefore the results will be the same as if the audio was processed online just in time it was broadcast.

7.2 Topics

At the 2008 Summer Olympic Games, there were sportsmen competing in 28 sports, 41 disciplines and 302 events. Based on a similar vocabulary usage in some disciplines and sport clusters used in our own data, we defined our topics by grouping one or more Olympic disciplines as depicted in figure 7.1.

[2]A skilled and specifically trained employee who listens to the audio stream and re-speaks it as intelligibly as possible in the real time.

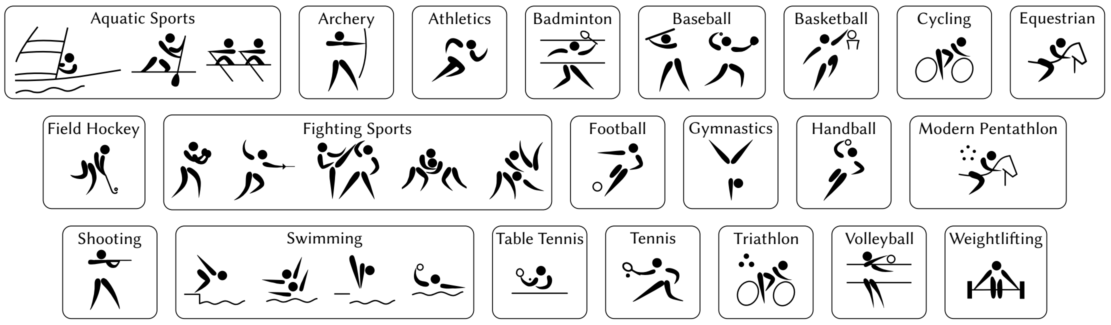

FIGURE 7.1: Mapping from summer Olympic disciplines to topic clusters. The set of disciplines and pictograms were adopted from `https://en.wikipedia.org/wiki/2008_Summer_Olympics`.

In sum, we have 21 topics covering all Olympic disciplines. As can be seen, we let most of the disciplines as one topic with following exceptions:

- all aquatic sports taking place outside of swimming pools (i.e. sailing, canoeing and rowing) were grouped into a topic *Aquatic sports*,
- softball was included into the *Baseball* topic,
- all fighting sports (i.e. boxing, fencing, taekwondo, wrestling and judo) were grouped into one topic *Fighting sports*,
- all aquatic sports taking place inside swimming pools (i.e. swimming, synchronized swimming, diving and water polo) were grouped into a topic *Swimming*.

7.3 Roster

It has been already mentioned that usually the main portion of featuring sportsmen are missing in the language model training data. Therefore, an actual roster must be supplemented into the language model from some external source before the ASR starts.

We have downloaded the full Olympic roster[3] from a web page providing an extensive collection of sports data[4]. In sum, we downloaded roster with 10.9 thousand sportsmen from 204 countries. Each sportsman in the roster has following metadata assigned:

- personal (full) name,
- country, for which the sportsman participate,

[3]Here, by the term *roster*, we mean a full list of all sportsmen from all countries participating at the Olympic Games.

[4]`https://www.sports-reference.com/olympics/summer/2008/`

- sport, in which the sportsman participate,
- age (of that time),
- gender (male/female),
- gold, silver and bronze medals (to simulate the conditions before the Olympics, we will not use this information).

For our work, the mapping from names to sports and vice versa is the most important information. We will use it to teach topic identifier a strong relationship between names and identified topics (sports) and also to group all sport-related names for the LM adaptation purposes.

Very important information can be also the gender, especially for languages that are somehow modifying male or female names. For example in Czech, female surnames are usually derived from male surnames by adding a suffix ”-ová”, and it is a custom to modify also foreign female surnames similarly (e.g. the name of pole vaulter *Yelena Isinbayeva* should be renamed to *Yelena Isinbayevová*). Using several grammatical rules, we modified all female surnames by adding an appropriate suffix. We expect their names to be pronounced in these forms during the sports summary shows.

7.3.1 Derived Forms of Names

Since the Czech language is highly inflected, we had to derive all related words in order to have a complete list of all possible forms of names which can occur in the speech. Based on Czech grammatical rules, from each name, several possessive forms were derived. For example, for the name *Bolt*, English has one possessive form *Bolt's*, however Czech has five forms depending on the gender and grammatical number of the subject: *Boltův*, *Boltova*, *Boltovy*, *Boltovi* and *Boltovo.*

Each base form and possessive form was then declined into seven cases of singular. We expect names to appear only in singular, although theoretically, also plural forms can occur when, for example, two brothers of the same surname would participate together.

Moreover, we store each form of each name in two variants: one as a *multiword*[5] containing all parts of the personal name and one consisting only from the surname. Both forms are very frequent in transcripts.

Theoretically, each sportsman can have almost a hundred derived forms (and thus vocabulary entries) of his name that can be used in the Czech language. However, in

[5]More words joined in the text by underscore symbol. Multiwords are considered as one token during the ASR.

practice, many derived forms are the same. After deriving all forms from all 10.9 thousand names, we ended up with a list of 118.5 thousand name-forms. All forms were derived automatically based on Czech grammatical rules.

7.3.2 Phonetic Transcriptions of Names

To get a phonetic transcription for each name and all derived forms, we used WFST-based G2P model trained from our existing lexicons. We used Phonetisaurus software[6] to train the model and generate 1-best phonetic transcription.

Our training lexicons contained both Czech names and foreign names with manually assigned transcriptions. We trained one model only for given names[7] and one model only for surnames. The trained models accept only Latin letters, so we mapped all non-Latin letters in names to the closest graphemes from the Latin alphabet.

We are aware that the transcription obtained from G2P model is not always absolutely correct, especially for exotic names. However, the manual transcription would be expensive and it would take long time. Since there is usually not much time between the announcement of the roster and the start of the sports event, and in order to keep the whole process fully automatic, we decided to use automatic G2P transcription regardless some minor errors. When annotating the audio records, we noticed that since sports commentators are not experts on pronunciation of foreign names, they often make similar errors in pronunciation as the trained model does, which slightly reduces the negative impact of G2P errors.

7.4 Text Data

For experiments in this chapter, we used three main sources of text data:

- Czech news articles — the same data as described in section 6.2,
- Czech sports transcripts — a collection of transcripts from various sports events (described later in this section),
- the roster — a list of all sportsmen as described in section 7.3.

In each adapted LM, we'd like to have some combination of all these text sources, because each source brings different vocabulary and n-grams into the adapted LM. The roster

[6]`https://github.com/AdolfVonKleist/Phonetisaurus`

[7]Here, by the term *given names*, we mean all parts of the personal name except for the surname.

contains all relevant names, sports transcripts cover all relevant n-grams for each sport and news articles supports the LM with a common language used by sports journalists.

The **news articles** corpus is described in the section 6.2. We used the whole corpus to train a general topic-independent LM in order to cover common journalist language and also for sporadic cases when the speech steps out from the sports event (e.g. to visit some winner-related place like a native village, favorite pub etc.). To simulate the worst case, in which we do not have any participating sportsman in our data, we had replaced all names from the roster by *unk* token[8], i.e. the same conditions as if all names from the roster were OOV words missing in the corpus. The corpus was collected automatically from web, so we did not have any annotations and no sport-labels in the data.

The **sports transcripts** corpus was collected at our department for live closed captioning of single-sport events broadcast by TV. The corpus contains (taking into account only summer Olympic disciplines) 21 sports and 7.5 million tokens. Transcripts are grouped into text files based on the underlying sports. To unify distinct forms of the same words, we used the same text preprocessing as in the news articles corpus.

All texts in the sports transcripts corpus were manually transcribed and annotated. The name of each player or competitor was manually labeled and supplemented by a number representing one of six grammatical cases[9]. The names that did not relate to the transcribed match or competition were not labeled, because the commentators may use them freely (e.g. legendary sportsmen such as "Michael Jordan", "Carl Lewis", or "Muhammad Ali"). Considering the above-mentioned labels instead of the individual words, these data are suitable to train a *class-based LMs* discussed later in section 7.6. For details about this corpus, see (Psutka et al., 2014).

7.5 Performance Measures

Analogously to additional performance measure defined in section 6.3, we are defining additional measure which will be used together with standard WER to evaluate experiments in this chapter. In the sports domain, the essential words, which should be recognized correctly, are mainly personal names of sportsmen. For example, when comparing two made-up hypotheses "Phelps win" and "helps wins" with a correct transcription "Phelps wins", both hypotheses would score the same WER (one word correct, one substitution, so $WER = 50\%$), however the former hypothesis is in our opinion much more informative for the viewer.

[8]A special token representing unknown words in the text.

[9]Actually, Czech has 7 grammatical cases, but the 5th case, vocative, used to directly addressing a person during conversation, is very rare in sports transcripts.

Hence, we are introducing additional performance measure more informative about errors in the sports domain, called **Name Error Rate**, NER for short. By names, we mean all personal names listed in the official roster including all derived name-forms. All other personal names in the reference transcript were not taken into account. Let the reference transcript be aligned with a recognized word sequence. Name error rate can then be computed as

$$NER = \frac{S_N + D_N + I_N}{N_N} \cdot 100\,\%, \tag{7.1}$$

where S_N is number of reference names substitutions, D_N is number of reference names deletions, I_N is number of reference names insertions, and N_N is the total number of names in the reference word sequence. In other words, NER is WER evaluated only for names of sportsmen in the reference transcripts.

7.6 Adapted LMs Preparation

We employed so called *class-based LMs* to join annotated corpus (sports transcripts) and the roster into one LM. The idea behind class-based LMs is that similar words appear in similar contexts. If we group similar words into one class, we can find all observed contexts of that class and assign them for all class-members (so-called *class expansion*). We can do it even if the member have not appeared in the corpus before. In this way, we can "inject" new class members into the LM and assign them relevant contexts as if we have observed them in our data.

Let us demonstrate the principle of class-based LMs on a very simplified example. Imagine, we have in our corpus bigram "Bolt wins" and we know "Bolt" is a name of a runner (someone has annotated our corpus). Then, we replace all occurrences of word "Bolt" by a class token "CLASS_NAME". Now let's assume we have got a roster before a new run and there are many names never observed before (let's call them runner1, runner2, ...). So we train a class-based LM from our corpus (we should have a bigram "CLASS_NAME wins" there), assign all names from the roster into the class *CLASS_NAME* and do the expansion of LM. After that, we should see bigrams "runner1 wins", "runner2 wins" and all other combinations of class names and contexts in the language model with appropriate probabilities. This simplified example could work in a non-inflected language. In Czech, however, we must define one special class per each grammatical category.

Since there are only 21 sport clusters in our data and the chance that the recognized speech will be about more than one sport in the same time is very low, we decided to prepare all LMs offline, save them and use the ASR system in the *prepared* mode (see section 5.6.1).

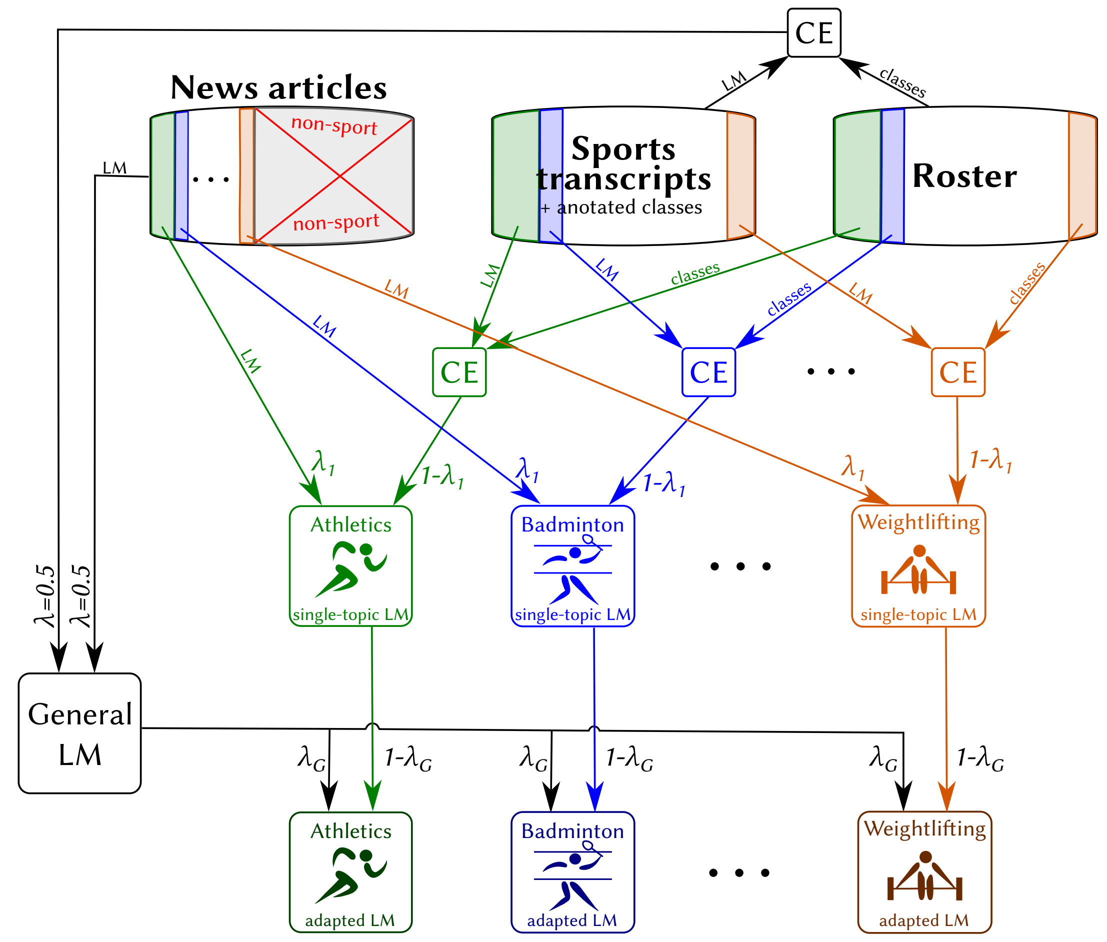

FIGURE 7.2: Scheme of adapted LMs preparation in the sports domain. The three upper rounded rectangles are text corpora, blocks with pictograms are language models, block "CE" stands for class expansion operation and colors are coding individual sports (topics).

The scheme of offline LM preparation is depicted in figure 7.2. We started with 3 text corpora: news articles, sports transcripts and the roster. The later two corpora can be easily separated into 21 single-sport subcorpora (differentiated by different colors in the figure). The news articles corpus was divided into single-sport subcorpora based on a topic identifier trained from sports transcripts (see section 7.7 for details about trained TI). The small amount of originally labeled data in the news articles corpus (see section 6.5) was used to find an optimal TI threshold to separate "non-sport" subcorpus containing documents irrelevant to any sport.

To get a single-topic language model, we trained a class-based LM from sports transcripts subcorpus, expanded classes with names from the roster and mixed it with news articles subcorpus. To get an adapted LM, we mixed single-topic and general LM. We will demonstrate the whole process (i.e. from text sources to the adapted LM for each sport) on the following badminton example (coded by blue color in the scheme). LMs for other sports were created analogously.

1. All badminton-related texts were selected from *sports transcripts* corpus. Let's denote this subcorpus T_B (the blue section of the *sports transcripts* block).
2. A trigram class-based LM, $LM_C(T_B)$, was trained from T_B. All unigrams, bigrams and trigrams from the data were included into $LM_C(T_B)$.
3. All badminton-related names R_B were selected from the *roster* (the blue section of the *roster* block) to define all class members. Each grammatical category represented one class. Within each class, all members were distributed uniformly.
4. Defined classes members were used to expand $LM_C(T_B)$ (blue block *CE*, short for "Class Expansion"). Since the number of class members was sometimes very high and fully expanded LMs would be extremely large, we did the full expansion only for 20 most frequent class n-grams for each grammatical category. Let's denote the expanded LM as $LM(T_B + R_B)$.
5. All badminton-related texts N_B from *news articles* corpus were selected (the blue section of the *news articles* block) and a standard word-based trigram LM, $LM(N_B)$, was trained. All unigrams, all bigrams and trigrams occurring at least twice in the subcorpus were included into $LM(N_B)$.
6. $LM(N_B)$ was mixed together with $LM(T_B + R_B)$ using linear interpolation. The weight of the former LM (λ_1) is a variable to be experimented with. The resulted mixed LM, LM_B^{ST}, is a single-topic LM trained from all badminton-related data where all names and name-forms of all badmintonists from the roster are included in expanded n-grams with appropriate probabilities.
7. A general LM, LM^G, was created similarly to single-topic LMs, but this time with all available data at once. First, LM trained from whole *sports transcripts* corpus was expanded with the full roster. Since classes containing all sportsmen were extremely large, we limited the class expansion only to unigrams, resulting in a LM, where all names and name-forms are included, but without any context, only as unigrams[10]. Then, expanded LM was mixed with a trigram LM trained from whole *news articles* corpus (same LM as described in the section 6.2.1). We fixed the interpolation weight to 0.5 as it yielded the best performance on the test data.

[10]Expanding higher order of n-grams actually does not make much sense in the case of full roster, because it would populate the expanded LM with a huge amount of cross-sport n-grams very unlikely to occur, e.g. "Bolt swims" or "Phelps shoots".

8. LM_B^{ST} was mixed together with LM^G using linear interpolation. The weight of the later LM (λ_G) is a variable to be experimented with. The resulted mixed LM, LM_B, is the final model adapted to the speech about badminton. It contains all names and name-forms of all badmintonists, badminton-specific words and n-grams and, through LM^G, also common-speech n-grams and all other-sports-specific names and n-grams (with lower probabilities).

For each experiment, after preparing all LMs according to the scheme 7.2, the only thing that needed to be done online during the experiment, was to switch between prepared LMs based on actual sport detected in the speech. This very simple switching scheme is depicted in figure 7.3. Aside from LMs prepared for single sports, we allowed the system to switch also back to the general LM whenever the underlying topic was unknown or ambiguous.

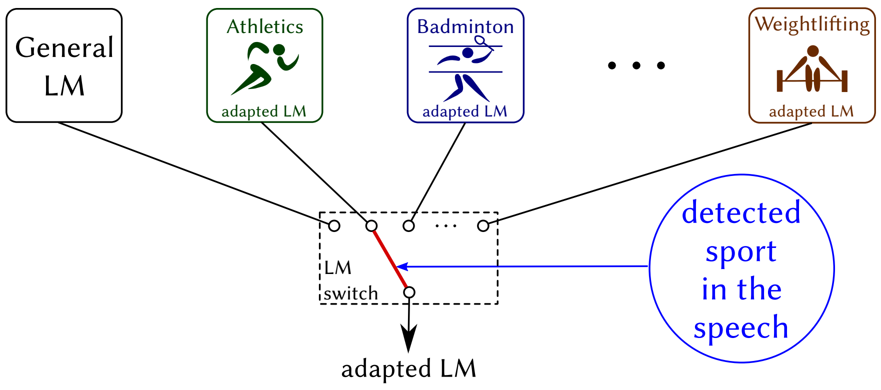

FIGURE 7.3: Scheme of online LM adaptation in the sports domain.

7.7 Online Sport Identification

Since the *sports transcripts* corpus is annotated and each transcript is assigned to some sport, we can consider this corpus as a labeled training dataset and train a sport identifier. For this purpose, all class occurrences in the transcripts were ignored, i.e. replaced with an *unk* token.

As some transcripts in the corpus were very long, and we used the topic identifier to identify sports from a very short context (usually tens of words), we split the corpus into sentences and considered each sentence as one short "document" with a corresponding sport label. To allow the model to learn also relations between sports and sportsmen, we supported each single-sport subcorpus with a list of all names and name-forms relevant to the sport (each name-form as one short "document") based on the roster.

We used trained topic identifier to cluster documents from the *news articles* corpus into individual sports. Since a small part of news articles was labeled from journalists, we used these data to find an optimal probability threshold to separate non-sport portion of the corpus. After that, we mixed all adapted LMs as described in the previous section 7.6.

To have a picture of which topic identifier to use, we tried different models, recognized all test records (with default system parameters and $\lambda_1 = \lambda_G = 0.5$) and compared the results. As a soft predictor, we trained all three supervised models listed in section 5.4.2 (SVM, CalibSVM, LSTM). As for a hard prediction, we are expecting exactly one sport at a time in the speech, so we fixed the thresholding strategy to *RCut(1)* (i.e. only the one most probable sport is identified at each time). For details about hard predictors, see section 5.4.3.

News clustering	**Online predictions**	**WER** [%]	**NER** [%]
SVM	SVM	4.333	17.524
	CalibSVM	4.346	17.863
	LSTM	4.345	17.787
CalibSVM	SVM	**4.322**	**16.886**
	CalibSVM	4.354	17.431
	LSTM	4.346	17.111
LSTM	SVM	4.323	17.036
	CalibSVM	4.359	17.524
	LSTM	4.333	17.280

Table 7.2: Comparison of different sport identifiers. We experimented with three models (SVM, CalibSVM, LSTM) trained from transcripts and roster. Each model was used in two different subtasks: clustering of news articles (documents from *News articles* corpus) into individual sports while throwing away all non-sport articles (news clustering column) and the second subtask was online sport identification during the recognition. For each combination, we evaluated WER and NER on the test dataset.

Results are shown in table 7.2. We achieved the best results ($WER = 4.32\%$, $NER = 16.89\%$) when using CalibSVM model to cluster unlabeled news data and SVM model to identify sports from partial hypotheses during the recognition. We will use this setting in following experiments.

To explain obtained results, we assume that when using CalibSVM, we were able to find more accurate threshold to separate non-sport articles in the news corpus and thus, the resulted topic clusters were more compact. On the other hand, CalibSVM was assembled from more sub-models using cross-validation, and thus each sub-model had observed only a part of the training dataset. Since each name of sportsman was present in the training dataset exactly once, the assembly of sub-models could get confused during online prediction when seeing a sport-related name in the recognized hypothesis. Similarly, we attribute worse results of LSTM classifier to the fact, that it was impossible to train reliable word vectors for names observed just once without any context, and so LSTM classifier was failing to identify a correct sport when seeing a sport-related name.

In this section, we were not experimenting with unsupervised models like LDA, because the set of sports (topics) was clear, and we had explicit linkage from transcripts and names to topics in the datasets (except for news articles corpus). If we trained an LDA model, we would have to infer again which sportsman and which transcript belongs to which latent topic, and thus we would throw away a lot of useful information we have in the dataset.

7.8 Weights of LMs

In the section 7.6, we have defined two interpolation weights as variables to be experimented with (λ_1 and λ_G). Both variables are also apparent in the figure 7.2. In this section, we ran experiments to find out optimal values of λ_1 and λ_G with respect to online topic-based LM adaptation.

Parameter λ_1 represents a proportion of *news articles* corpus in the single-topic LM. In other words, it defines the balance between sport-specific news LM and sport-specific transcripts LM (including the roster) when mixing together. Figure 7.4 shows the impact of λ_1 on WER and NER with λ_G fixed to 0.5.

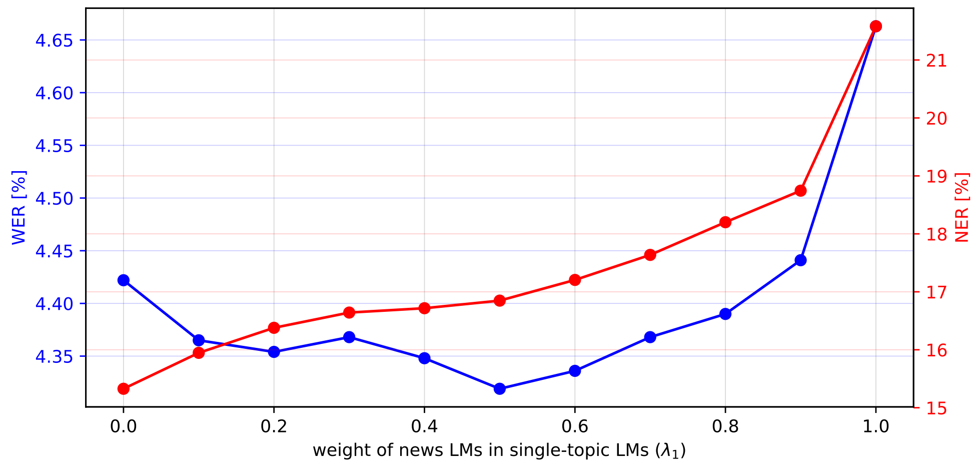

FIGURE 7.4: Results with different weight of news LM in single-topic LM in the sports domain. For varying values of λ_1 and all other parameters fixed on default values, WER and NER was evaluated.

The leftmost point in the figure 7.4 ($\lambda_1 = 0$) corresponds to a situation, when the single-topic LM was trained using only transcripts corpus with expanded names from the roster. The rightmost point ($\lambda_1 = 1$), on the other hand, was an experiment, when only data from the news domain corpus was used to train single-topic LM (i.e. no transcripts and no names). As we expected, the higher λ_1 was, the lower number of names was recognized correctly. However, the trajectory of WER indicates, that a balanced mix of both LMs is needed to reach an optimal word error rate. We scored the best WER with $\lambda_1 = 0.5$, i.e. equally balanced mix of both LMs.

Parameter λ_G represents a proportion of general LM in the adapted LM. Figure 7.5 shows the impact of λ_G on WER and NER with λ_1 fixed to 0.5. As can be seen, there is a significant fall in performance for $\lambda_G = 0$. It clearly demonstrates that the general LM should not be completely omitted in the adapted LM, but it should be involved to support adapted LM with general statistics about common-language n-grams and names from other sports. One may expect that when using only topic-specific data in the adapted LM, NER should not be deteriorated, however sometimes the sport was not identified correctly, and in these situations, without general LM, all relevant names were completely missing in the adapted LM, and hence the NER was significantly deteriorated.

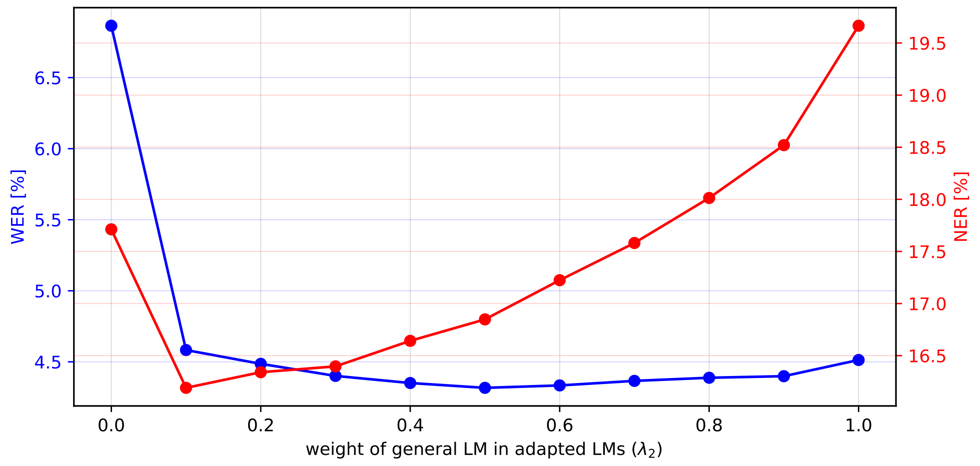

FIGURE 7.5: Results with different weight of general LM in the sports domain. For varying values of λ_G and all other parameters fixed on default values, WER and NER was evaluated.

The trend of the rest of the figure is similar to λ_1 (fig. 7.4): the higher λ_G was, the lower number of names was recognized correctly, as expected. The best WER was achieved when mixing general LM with single-topic LMs in 50:50 ratio.

In sum, we scored the best result with $\lambda_1 = \lambda_G = 0.5$ ($WER = 4.32\%$, $NER = 16.89\%$). By manipulating lambdas to lower values, we achieved better recognition of names, but only at the expense of a deterioration in WER. By manipulating lambdas to higher values, both WER and NER were deteriorated.

The result obtained with $\lambda_G = 1$ ($WER = 4.51\%$, $NER = 19.68\%$) would be the same for all setting of λ_1, because adapted LMs do not depend on single-topic LMs in this case. Actually, adapting to a LM consisting fully of a general LM is effectively no LM adaptation at all. Therefore, this result equals to a baseline performance of the system.

7.9 Fine-Tuning of Parameters

To find optimal system settings, we ran a grid search over system parameters analogous with the fine-tuning of parameters in the news domain (sections 6.5.3.2 - 6.5.5). Since the behavior of many parameters was very similar, we are summarizing the results in this short section while discussing only few interesting points.

We achieved the best results when cropping each partial hypothesis to the last 50 words excluding the last three words. As for the weighting function of the cropped word sequence, we observed slightly better result using linear function than default constant weight, however the differences were very modest (see figure 7.6).

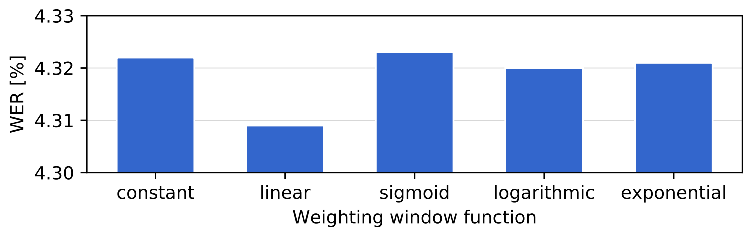

FIGURE 7.6: Results with different weighting functions for recognized words in the sports domain. For varying weighting functions and all other parameters fixed on default values, WER was evaluated.

As for the hypotheses merger, we obtained the best results when setting parameter merge-trim to 1 second and merge-offset to 3 seconds.

Surprisingly, we observed interesting results when trying different values of steadiness. We plotted these results in the figure 7.7. It can be seen that WER and NER were highly correlated reaching the best results with steadiness set to zero seconds. It means that the best result was achieved when the system considered all identified topics always as steady and immediately adapted the LM to whatever sport the TI had identified. That implies the performance of the sport identifier was very accurate without much noise in predictions. Further increasing the value of steadiness led only to deterioration of both WER and NER.

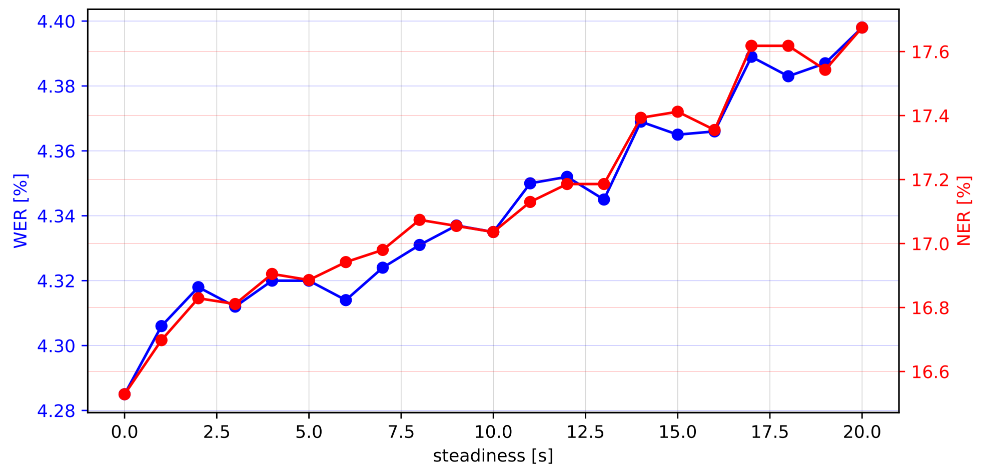

Figure 7.7: Results with different steadiness of the topic-change detector in the sports domain. For varying values of steadiness and all other parameters fixed on default values, WER and NER was evaluated.

The best result, we achieved with steadiness set to zero seconds and other parameters set as mentioned above, was $WER = 4.255\%$ and $NER = 16.566\%$. Compared to the baseline system (with no LM adaptation), we achieved 5.74% relative reduction of WER and 15.84% relative reduction of NER.

The main results achieved so far along with results for the baseline system are summarized in table 7.3.

	WER [%]	**NER** [%]
general LM (baseline)	4.51	19.68
adapted LM, default params	4.32	16.89
adapted LM, fine-tuned params	4.26	16.57

Table 7.3: Results summary for experiments with shadow speaker in the sports domain. For the baseline system and two different LM adaptations, we are showing the best WER and NER we achieved.

7.10 Using Original Audio

To have an idea of how experiments in the sports domain would end without the support of the shadow speaker, we ran similar experiments, but this time we used original audio records as inputs to our system, i.e. the exact same signal that was broadcast to TV viewers. Note, that different acoustic model had to be used when recognizing original audio. Instead of fine-tuned AM trained specifically for particular shadow speaker, a general-purpose AM was used in this case. The main results are summarized in table 7.4.

	WER [%]	**NER** [%]
general LM (baseline)	25.30	41.46
adapted LM, default params	24.56	37.85
adapted LM, fine-tuned params	24.43	36.72

Table 7.4: Results summary for experiments without shadow speaker in the sports domain. For the baseline system and two different LM adaptations, we are showing the best WER and NER we achieved.

As can be seen, without the re-speaking of the commentary by the shadow speaker, WER of the baseline system was more than 25%, which means there was an error in every fourth word in average. NER of the system was more than 40% meaning only six out of ten names were recognized correctly. Topic-based LM adaptation slightly improved the baseline system, especially in terms of NER, however, the error rates of recognized hypotheses are still too high to be used to generate understandable captions.

Original audio in the sports domain suffers a lot from a loud background noise and frequent changes in the style of the speech, that is why results for this audio signal were so poor. The re-spoken audio, on the other hand, contained only one skilled and specifically trained speaker with background noise reduced to minimum. In the case of original audio signal, we had to use general speaker-independent acoustic model, but in the case of re-spoken audio, we were using fine-tuned speaker-specific AM achieving more accurate results.

Clearly, the original audio stream was too noisy to be used for accurate real-time recognition with a state-of-the-art speech recognition system. Based on these results, we are stating, that the usage of shadow speaker is essential for closed captioning of TV sports summaries.

7.11 Results Summary

The most important results achieved in this chapter concerning topic-based LM adaptation in the sports domain are summarized in table 7.5. We are showing results for two different audio signals:

- original audio — the exact same signal that was streamed to TV viewers during the Olympic summary shows,
- re-spoken audio — audio produced by a shadow speaker who was listening to the original audio and re-speaking it in the real time.

Audio signal	**ASR system**	**WER** [%]	**NER** [%]
original	*general LM (baseline)*	25.30	41.46
	adapted LM	24.43 (−3.45%)	36.72 (−11.42%)
re-spoken	*general LM (baseline)*	4.51	19.68
	adapted LM	4.26 (−5.74%)	16.57 (−15.84%)

Table 7.5: Results summary for the sports domain. For two audio signals, we are showing the best WER and NER (Name Error Rate) we achieved and corresponding relative improvements over non-adaptable baseline systems.

Since we had only restricted amount of relevant texts for each sport in our text sources, both general LM and adapted LMs were about the same size, thus the real-time ratio was not an issue in this chapter. In all experiments, baseline systems were recognized with the real-time ratio about 0.3 on the tested machine (Intel Core i7-7800X, used all 6 cores) and systems with adapted LM about 0.4. That is why we are not evaluating differences in real-time ratio in table 7.5.

Chapter 8

Experimental Results Summary

In this chapter, we are summing up the most important results we achieved in this thesis. We have extended state-of-the-art speech recognition system with an online topic-based LM adaptation as described in the chapter 5. We used this system to automatically transcribe a speech in two multi-topic domains: live TV news (for details, see chapter 6) and live TV sports summaries (for details, see chapter 7). The most significant results from both domains along with the relative improvements over non-adaptable baseline systems (i.e. systems using a general topic-independent language model) are shown in table 8.1. In the case of TV news domain, we are showing for comparison results obtained with a topic model trained in fully unsupervised way from a collection of plain text documents (unsup.) and with a supervised topic identifier trained from manually labeled dataset (sup.).

Domain	**Audio**	**ASR system**	**WER** [%]	**PNER/NER** [%]
TV news	original	*general LM (baseline)*	16.95	17.41
		adapted LM (unsup.)	16.37 (−3.41%)	16.88 (−3.05%)
		adapted LM (sup.)	16.36 (−3.48%)	16.66 (−4.34%)
TV sports	original	*general LM (baseline)*	25.30	41.46
		adapted LM	24.43 (−3.45%)	36.72 (−11.42%)
	re-spoken	*general LM (baseline)*	4.51	19.68
		adapted LM	4.26 (−5.74%)	16.57 (−15.84%)

TABLE 8.1: The main results of the thesis. For both TV news and TV sports domain, we are showing the best results we achieved in this thesis and corresponding relative improvements over non-adaptable baseline systems. Experiments were evaluated in terms of WER and $PNER$ (Proper Noun Error Rate) in the case of news domain or NER (Name Error Rate) in the case of sports domain.

8.1 Test of Significance

To verify that improvements in the table 8.1 are statistically significant, we ran a significance tests. We chose *Matched Pairs Sentence-Segment Word Error* (MAPSSWE) test, which is a very powerful test often used for the analysis of benchmark speech recognition systems (Gillick and Cox, 1989). We used *Speech Recognition Scoring Toolkit* (SCTK) implemented in National Institute of Standards and Technology (NIST) (Pallett et al., 1990). For each system with adapted LM in table 8.1, we tested following null hypothesis: the number of unique utterance errors are equal for both baseline and proposed systems.

All tests confirmed a significant difference between recognized utterances of baseline and proposed systems at the level of $p = 0.001$. Therefore, we can reject the null hypothesis at the 99% confidence level, and state that all improvements listed in the table 8.1 **are statistically significant**.

Chapter 9

Discussion

Experiments in this thesis showed that proposed ASR system with online topic-based LM adaptation performs significantly better than a system without LM adaptation in real-time multi-topic tasks.

In the news domain (chapter 6), the proposed adaptable system scored 3.48 % relative reduction of WER and 4.34 % relative reduction of proper noun error rate over the baseline system. One interesting result is that unsupervised topic models (specifically LDA) performed almost as well as supervised text classifiers, which were trained from a large collection of manually labeled data. This result emphasizes, that proposed topic-based LM adaptation can be successfully employed even in problems, where the text corpus is not annotated and manual topic-labeling of documents would be unfeasible.

Experiments in section 6.5.3.1 showed another interesting result: even though news reports tend to be multi-topic by their nature (in our data, there are three topics per news article in average), a similar, and sometimes even lower, error rates can be achieved when performing only a single-topic LM adaptation, i.e. when the system is adapting only to one most probable topic at each time. For example, when a news report is about heavy snow and traffic jams, a system, which switches the adapted LM between these two topics, performs similarly (sometimes even better) than a system, which adapts to a mixture of both topics. This result is interesting, because using single-topic LM adaptation is very beneficial in proposed system, since the number of possible adapted LMs equals to the number of topics, and thus, all adapted LMs can be prepared in advance and stored in cache leading to LM adaptation with very low latency.

Real-time transcription of TV sports summaries is another multi-topic problem, where topic-based LM adaptation can help a lot, because individual sports have markedly different word sequence statistics and sportsmen are always highly related to only one sport. In experiments in chapter 7, the system was adapting to one of 21 sports (topics) at each time. The improvements scored by the proposed system over the baseline system are very significant, especially in the terms of name error rates: 11.42 % relative reduction of NER when recognizing original audio and 15.84 % when recognizing audio re-spoken by a shadow speaker.

Results in section 7.10 clearly demonstrate the necessity of re-speaking the original audio by a shadow speaker in closed captioning of TV sports summaries task. Re-speaking reduces the effect of a loud background noise and frequent changes in the style of the speech. For example, the WER scored by a baseline system with and without a shadow speaker was 4.51 % and 25.30 %, respectively.

On the other hand, there are some limitations of proposed solution. It is not suitable for offline problems (e.g. to transcribe a multi-topic audio record), because it is designed to work online only with limited partial hypotheses. For offline problems, where the system can look at the whole record at once, it is usually better to use a standard two-pass approach to segment the record into topics. Also, too large LMs cannot be used for online LM adaptation in proposed system, because there would be too long delays caused by loading huge files into the decoder (see for example bottom graph in the figure 6.11). The size of LMs must be reasonable with respect to the reading speed of a storage in use. Another limitation is that because of the architecture of existing ASR system, all LMs must be static n-gram models in ARPA format, excluding all dynamic features inside LMs.

Chapter 10

Conclusion

The research area of this thesis was online topic-based language model (LM) adaptation. In present days, it is a very important problem, especially in multi-topic tasks, where a real-time speech recognition is required, but a general LM cannot model varying word sequence statistics in particular topics appropriately.

This thesis surveys the state of the art of the problem including also detailed theoretical background of used methods and models. Published works which are most relevant to this thesis are described in more details and the main differences from solution proposed in this thesis are pointed out.

Chapter 5 describes the core of this thesis: an innovative solution to extend existing real-time ASR system by online topic-based LM adaptation. The originality of proposed solution lies in minimizing latency of the topic-based adaptation by using two parallel decoders (instead of commonly used two-pass approaches) and online merging their outcomes. Another contribution of this work is the way how the change of the speech topics is online detected from a stream of partial hypotheses.

The proposed adaptable system was implemented and tested in two multi-topic real-time ASR problems: live transcription of TV news and live transcription of TV sports summaries. For both problems, experiments in this thesis showed that proposed system performs significantly better than a system without LM adaptation, and that topic-based LM adaptation can reduce error rates of live closed captions, especially by better recognizing topic-specific content-bearing proper nouns crucial for deaf and hard-of-hearing audiences to understand the content of the speech. It can also ease the work of shadow speakers by dynamically and automatically enriching the ASR system with topic-specific words and n-grams whenever the topic occurs in the speech.

As for the future work, it would be interesting to add a support of dynamic LMs directly into the decoding process. It would avoid mixing of static LMs in the background processes and high data traffic when saving and loading large LMs. However, with current architecture of our ASR system, we found such support to be very hard to implement while keeping the system optimized for low-latency real-time recognition. Also, it will be interesting to monitor the research of end-to-end ASR systems based on deep recurrent neural networks, where we can observe a lot of effort in present days, because such models could learn various topic-related features from training data and implicitly perform online LM adaptation by their nature.

My future work in this field consists mainly in extending the usage of the system by preparing more topic-specific LMs covering additional important domains with specific word sequence statistics, such as geographical regions, various professions etc. We plan to include these topic-specific LMs into our ASR system in order to produce more accurate closed captions for various live TV shows and also to enhance our existing dictating system.

By writing this thesis, I believe I have presented an innovative and working solution to improve automatic speech recognition in online multi-topic tasks while fulfilling all objectives defined in chapter 2.

Appendix A

Default Configuration File of Proposed System

```
[onlineASR]
ASR_set               = "ASR.SET" # path to compiled LVCSR set (AM+LM+NET)
lib_name              = "libLVCSR_mkl.so" # path to LVCSR lib
licence               = ""         # licence ID
activation            = ""         # activation key
lProcessorNumber      = 1          # number of CPUs
lProgressPeriod       = 100        # how often a progress is reported, in number
                                   # of frames
lResultPeriod         = 100        # how often a partial result is generated,
                                   # in number of frames
lResultMinFrameNumber = 0
lResultMinWordNumber  = 0
uintInputDeviceID     = 0
uintOutputDeviceID    = 0
bGPUUsing             = True       # use GPU if available
bRealtimeProcessing   = True

[LMAdapt]
adapt_LMW             = 13         # language model weight of adapted LMs
adapt_WIP             = -5         # word insertion penalty of adapted LMs
retro_rec             = 5          # retro-recognition, in seconds

[RecognitionParam]
fn_type               = 0          # 0..wav, 1..htk
lattice_slf           = None       # prefix of periodically saved SLF lattices
lattice_json          = None       # prefix of periodically saved JSON lattices
return_json           = False      # results are JSON lattices
```

```
return_mesh           = False     # results are word confusion networks
mesh_threshold        = 0.01      # pruning threshold for word confusion networks
parallel_recognition  = True      # enable two parallel decoders
vol_norm              = False     # volume normalization

[Audio]
convert_input         = True      # convert the audio/video before ASR
audio_tool            = "ffmpeg"  # ffmpeg, avconv or similar
target_n_channels     = 1
target_frequency      = 16000

[HypothesesMerger]
merge_online          = True      # generate merged hypotheses online
merge_trim            = 2         # how many seconds to trim (drop) from both
                                  # ends of the adapted-LM hypothesis
merge_offset          = 3         # the width of the interval at both ends of
                                  # remaining adapted-LM hypothesis, where the
                                  # best cut is searched for; in seconds

[TopicIdentifier]
OTI                   = True      # do online topic identification
unit_type             = "word"    # "word" | "sec"
drop_N_last_units     = 2         # how many last units are thrown away
keep_N_most_recent_units = 50     # how many last units are selected from
                                  # remaining result
trained_encoder       = "TFIDF.pkl" # pickled data encoder
units_weighting       = "constant" # "constant" | "linear" | "sigmoid" |
                                  # "logarithmic" | "exponential"
trained_topic_predictor = "SVM.pkl" # pickled topic predictor
thresholding_strategy = "RCut"
thresholding_strategy_kwargs = {"k": 1}

[TopicChangeDetector]
steadiness            = 5         # how many seconds into the history any topic
                                  # must have been continuously identified to be
                                  # marked as a steady topic in the speech

[LMMixer]
OTA                   = True      # do online topic adaptation of LM
mode                  = "prepared" # "prepared" | "offline" | "online"
topic2LM              = "map.txt@utf-8" # path to a mapping topic -> LM
cache_size            = 25        # how many recent adapted LMs to keep in cache
general_LM            = "LM.arpa" # path to general LM
general_LM_w          = 0.5       # weight of general LM
```

```
imgLM_bin             = "ImageLM.exe" # path to LM Imager
ASR_net               = "ASR.NET" # path to compiled recognition network
srilm_param           = "-unk"    # additional params for SRILM's ngram program
no_topic_LM           = ""        # LM used when no steady topic is identified,
                                  # empty string for keeping the last LM
no_topic_LM_patience  = 3         # how many seconds to wait before change LM to
                                  # no_topic_LM
```

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

James Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media, 2012.

Alexandre Allauzen and Jean-Luc Gauvain. Diachronic vocabulary adaptation for broadcast news transcription. In *INTERSPEECH*, pages 1305–1308, 2005.

Sankaranarayanan Ananthakrishnan, Stavros Tsakalidis, Rohit Prasad, and Premkumar Natarajan. On-line language model biasing for multi-pass automatic speech recognition. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

Cedric GP Auzanne, John S Garofolo, William M Fisher, and Jonathan G Fiscus. Automatic language model adaptation for spoken document retrieval. In *RIAO*, pages 132–141. Citeseer, 2000.

Madeleine Bates, Robert Bobrow, Pascale Fung, Robert Ingria, Francis Kubala, John Makhoul, Long Nguyen, Richard Schwartz, and David Stallard. The BBN/HARC spoken language understanding system. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 111–114. IEEE, 1993.

Jerome R Bellegarda. Statistical language model adaptation: review and perspectives. *Speech communication*, 42(1):93–108, 2004.

Adam Berger and Robert Miller. Just-in-time language modelling. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 705–708. IEEE, 1998.

Stefan Besling and Hans-Günter Meier. Language model speaker adaptation. In *EUROSPEECH*. ISCA, 1995.

Brigitte Bigi, Yan Huang, and Renato De Mori. Vocabulary and language model adaptation using information retrieval. In *INTERSPEECH*, 2004.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Gilles Boulianne, Jean-Francois Beaumont, Maryse Boisvert, Julie Brousseau, Patrick Cardinal, Claude Chapdelaine, Michel Comeau, Pierre Ouellet, and Frédéric Osterrath. Computer-assisted closed-captioning of live TV broadcasts in French. In *INTERSPEECH*. Citeseer, 2006.

Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, page 4. ACM, 2010.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: a neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016.

Langzhou Chen, Jean-Luc Gauvain, Lori Lamel, Gilles Adda, and Martine Adda. Using information retrieval methods for language model adaptation. In *Seventh European Conference on Speech Communication and Technology*, 2001.

Langzhou Chen, Lori Lamel, Jean-Luc Gauvain, and Gilles Adda. Dynamic language modeling for broadcast news. In *INTERSPEECH*, 2004.

Lungzhou Chen, Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. Unsupervised language model adaptation for broadcast news. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–220. IEEE, 2003.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

X Chen, T Tan, Xunying Liu, Pierre Lanchantin, M Wan, Mark JF Gales, and Philip C Woodland. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Katya Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint arXiv:1712.01769*, 2017.

Philip R Clarkson and Anthony J Robinson. Language model adaptation using mixtures and an exponentially decaying cache. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 799–802. IEEE, 1997.

Alexis Conneau, Holger Schwenk, Loıc Barrault, and Yann Lecun. Very deep convolutional networks for natural language processing. *arXiv preprint*, 2016.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3061–3069, 2015.

John N Darroch and Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480, 1972.

Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

Salil Deena, Madina Hasan, Mortaza Doulaty, Oscar Saz, and Thomas Hain. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition and alignment. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 27(3):572–582, 2019.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

Karol Draszawka and Julian Szymański. Thresholding strategies for large scale multi-label text classifier. In *Human System Interaction (HSI), 2013 The 6th International Conference on*, pages 350–355. IEEE, 2013.

JD Echeverry-Correa, J Ferreiros-López, A Coucheiro-Limeres, R Córdoba, and JM Montero. Topic identification techniques applied to dynamic language model adaptation for automatic speech recognition. *Expert Systems with Applications*, 42 (1):101–112, 2015.

Marcello Federico and Nicola Bertoldi. Broadcast news LM adaptation over time. *Computer Speech & Language*, 18(4):417–435, 2004.

Sadaoki Furui. Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 1991–1994. IEEE, 1986.

Laurence Gillick and Stephen J Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 532–535. IEEE, 1989.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772, 2014.

Roberto Gretter and Giuseppe Riccardi. On-line learning of language models with word error probability distributions. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 557–560. IEEE, 2001.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. DeepSpeech: scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Ali Haznedaroglu and Levent M Arslan. Language model adaptation for automatic call transcription. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4102–4106. IEEE, 2014.

Marti A Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 3–10. Association for Computational Linguistics, 1999.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Geoffrey E Hinton, James L McClelland, and David E Rumelhart. Distributed representations. In *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, pages 77–109. MIT Press, 1986.

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for Latent Dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

Jan Hoidekr, JV Psutka, Aleš Pražák, and Josef Psutka. Benefit of a class-based language model for real-time closed-captioning of TV ice-hockey commentaries. In *Proceedings of LREC*, pages 2064–2067, 2006.

Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62. Citeseer, 2005.

Bo-June Paul Hsu and James Glass. Style & topic language model adaptation using HMM-LDA. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 373–381. Association for Computational Linguistics, 2006.

Akinori Ito, Yasutomo Kajiura, Shozo Makino, and Motoyuki Suzuki. An unsupervised language model adaptation based on keyword clustering and query availability estimation. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 1412–1418. IEEE, 2008.

Rukmini M Iyer and Mari Ostendorf. Modeling long distance dependence in language: topic mixtures versus dynamic cache models. *Speech and Audio Processing, IEEE Transactions on*, 7(1):30–39, 1999.

David Janiszek, Renato De Mori, and Frederic Bechet. Data augmentation and language model adaptation. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 549–552. IEEE, 2001.

Fred Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam, 1980.

Frederick Jelinek, Bernard Merialdo, Salim Roukos, and Martin Strauss. A dynamic language model for speech recognition. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

Thorsten Joachims. *Text categorization with support vector machines: learning with many relevant features.* Springer, 1998.

Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems*, pages 919–927, 2015.

Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using one-hot LSTM for region embeddings. *arXiv preprint arXiv:1602.02373*, 2016.

Slava M Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401, 1987.

Dietrich Klakow. Language model adaptation for tiny adaptation corpora. In *INTERSPEECH.* Citeseer, 2006.

Filipp Korkmazsky, Oliver Jojic, and Bageshree Shevade. Boosting of speech recognition performance by language model adaptation. In *Aerospace Conference, 2007 IEEE*, pages 1–10. IEEE, 2007.

Roland Kuhn and Renato De Mori. A cache-based natural language model for speech recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12 (6):570–583, 1990.

Christine Largeron, Christophe Moulin, and Mathias Géry. MCut: a thresholding strategy for multi-label classification. In *Advances in Intelligent Data Analysis XI*, pages 172–183. Springer, 2012.

Jey Han Lau, David Newman, and Timothy Baldwin. Machine reading tea leaves: automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539, 2014.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: a maximum entropy approach. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 45–48. IEEE, 1993.

Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

Gwénolé Lecorvé, Guillaume Gravier, and Pascale Sébillot. Constraint selection for topic-based MDI adaptation of language models. In *International Conference on Speech and Language Technology, Interspeech'09*, pages 368–371, 2009.

Gwénolé Lecorvé, John Dines, Thomas Hain, and Petr Motlicek. Supervised and unsupervised Web-based language model domain adaptation. Technical report, Idiap, 2012.

Jan Lehečka and Aleš Pražák. Online LDA-based language model adaptation. In *International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, pages 334–341. Springer, 2018.

Jan Lehečka and Jan Švec. Improving speech recognition by detecting foreign inclusions and generating pronunciations. In *International Conference on Text, Speech and Dialogue*, pages 295–302. Springer, 2013.

Jan Lehečka and Jan Švec. Improving multi-label document classification of Czech news articles. In *International Conference on Text, Speech, and Dialogue*, pages 307–315. Springer, 2015.

Jan Lehečka. Online topic-based language model adaptation. Report for state doctoral examination, University of West Bohemia, Faculty of Applied Sciences, 2016.

David D Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50. ACM, 1992.

Juan Manuel Lucas-Cuesta, Javier Ferreiros, F Fernández-Martı, Julian David Echeverry, S Lutfi, et al. On the dynamic adaptation of language models based on dialogue information. *Expert Systems with Applications*, 40(4):1069–1085, 2013.

Paul Maergner, Alex Waibel, and Ian Lane. Unsupervised vocabulary selection for real-time speech recognition of lectures. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4417–4420. IEEE, 2012.

Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding consensus among words: lattice-based word error minimization. In *Eurospeech*, 1999.

James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 2000.

Ciro Martins, António Teixeira, and João Neto. Dynamic language modeling for a daily broadcast news transcription system. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 165–170. IEEE, 2007.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

Gary Monroe, James C French, Allison L Powell, et al. Obtaining language models of web collections using query-based sampling techniques. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 1241–1247. IEEE, 2002.

Arturo Montejo-Ráez and Luis Alfonso Ureña-López. Selection strategies for multi-label text categorization. In *Advances in Natural Language Processing*, pages 585–592. Springer, 2006.

Hiroaki Nanjo and Tatsuya Kawahara. Unsupervised language model adaptation for lecture speech recognition. In *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. WFST-based grapheme-to-phoneme conversion: open source tools for alignment, model-building and decoding. In *10th International Workshop on Finite State Methods and Natural Language Processing*, page 45, 2012.

David Pallett, W Fisher, and J Fiscus. Tools for the analysis of benchmark speech recognition tests. In *proc. ICASSP*, volume 1, pages 97–100, 1990.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

Ignazio Pillai, Giorgio Fumera, and Fabio Roli. Threshold optimisation for multi-label classifiers. *Pattern Recognition*, 46(7):2055–2065, 2013.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

A. Pražák, L. Müller, and Psutka Josef V. LVCSR system for automatic online subtitling. pages 325–328, Moscow, 2005. Moscow State Linguistics University. ISBN 5-7452-0110-X. URL `http://www.kky.zcu.cz/en/publications/PrazakA_2005_LVCSRsystemfor`.

Aleš Pražák, Zdeněk Loose, Jan Trmal, Josef V. Psutka, and Josef Psutka. Novel approach to live captioning through re-speaking: tailoring speech recognition to re-speaker's needs. In *INTERSPEECH*, 2012.

Josef V Psutka, Aleš Pražák, Josef Psutka, and Vlasta Radová. Captioning of live TV commentaries from the Olympic games in Sochi: some interesting insights. In *International Conference on Text, Speech, and Dialogue*, pages 515–522. Springer, 2014.

J Ross Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014.

Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Kanishka Rao, Fuchun Peng, Hasim Sak, and Françoise Beaufays. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE, 2015.

Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

Frederick Richardson, Mari Ostendorf, and Jan Robin Rohlicek. Lattice-based search strategies for large vocabulary speech recognition. In *Acoustics, Speech, and Signal*

Processing, 1995. ICASSP-95., 1995 International Conference on, volume 1, pages 576–579. IEEE, 1995.

Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language*, 10(3):187–228, 1996.

Sarah E Schwarm, Ivan Bulyko, and Mari Ostendorf. Adaptive language modeling with varied sources to cover new vocabulary items. *Speech and Audio Processing, IEEE Transactions on*, 12(3):334–342, 2004.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

Abhinav Sethy, Panayiotis G Georgiou, and Shrikanth Narayanan. Building topic specific language models from webdata using competitive models. In *INTERSPEECH*, pages 1293–1296, 2005.

Yangyang Shi, Pascal Wiggers, and Catholijn M Jonker. Combining topic specific language models. In *Text, Speech and Dialogue*, pages 99–106. Springer, 2011.

Yangyang Shi, Pascal Wiggers, and Catholijn M Jonker. Adaptive language modeling with a set of domain dependent models. In *Text, Speech and Dialogue*, pages 472–479. Springer, 2012.

Lucie Skorkovská, Pavel Ircing, Aleš Pražák, and Jan Lehečka. Automatic topic identification for large scale language modeling data filtering. In *Text, Speech and Dialogue*, pages 64–71. Springer, 2011.

Bernd Souvignier, Andreas Kellner, Bernhard Rueber, Hauke Schramm, and Frank Seide. The thoughtful elephant: strategies for spoken dialog systems. *Speech and Audio Processing, IEEE Transactions on*, 8(1):51–62, 2000.

Andreas Stolcke et al. SRILM - an extensible language modeling toolkit. In *INTERSPEECH*, 2002.

Jan Švec, Jan Lehečka, Pavel Ircing, Lucie Skorkovská, Aleš Pražák, Jan Vavruška, Petr Stanislav, and Jan Hoidekr. General framework for mining, processing and storing large amounts of electronic texts for language modeling purposes. *Language resources and evaluation*, 48(2):227–248, 2014.

Yik-Cheung Tam and Tanja Schultz. Dynamic language model adaptation using variational Bayes inference. In *INTERSPEECH*, pages 5–8, 2005.

Edmondo Trentin and Marco Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37(1):91–126, 2001.

Grigorios Tsoumakas, Apostolos Papadopoulos, Weining Qian, Stavros Vologiannidis, Alexander D'yakonov, Antti Puurula, Jesse Read, Jan Švec, and Stanislav Semenov. WISE 2014 challenge: multi-label classification of print media articles to topics. In *Web Information Systems Engineering–WISE 2014*, pages 541–548. Springer, 2014.

Gokhan Tur and Andreas Stolcke. Unsupervised language model adaptation for meeting recognition. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–173. IEEE, 2007.

Anand Venkataraman and Wen Wang. Techniques for effective vocabulary selection. *arXiv preprint cs/0306022*, 2003.

Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009.

Wen Wang and Andreas Stolcke. Integrating MAP, marginals, and unsupervised language model adaptation. In *INTERSPEECH*, pages 618–621. Citeseer, 2007.

Shinji Watanabe, Tomoharu Iwata, Takaaki Hori, Atsushi Sako, and Yasuo Ariki. Topic tracking language model for speech recognition. *Computer Speech & Language*, 25(2): 440–461, 2011.

Pascal Wiggers and Leon JM Rothkrantz. Topic-based language modeling with dynamic Bayesian networks. In *INTERSPEECH*, 2006.

Jun Wu and Sanjeev Khudanpur. *Maximum entropy language modeling with non-local dependencies.* PhD thesis, Johns Hopkins University, 2002.

Yiming Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1-2):69–90, 1999.

Steve J Young and Sj Young. *The HTK hidden Markov model toolkit: design and philosophy.* Citeseer, 1993.

Harry Zhang. The optimality of naive Bayes. *AA*, 1(2):3, 2004.

Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Xiaojin Zhu and Ronald Rosenfeld. Improving trigram language modeling with the world wide web. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 533–536. IEEE, 2001.

List of Author's Publications

Lucie Skorkovská, Pavel Ircing, Aleš Pražák, and Jan Lehečka. Automatic topic identification for large scale language modeling data filtering. In *Text, Speech and Dialogue*, pages 64–71. Springer, 2011.

Jan Lehečka and Jan Švec. Improving speech recognition by detecting foreign inclusions and generating pronunciations. In *International Conference on Text, Speech and Dialogue*, pages 295–302. Springer, 2013.

Jan Švec, Jan Lehečka, Pavel Ircing, Lucie Skorkovská, Aleš Pražák, Jan Vavruška, Petr Stanislav, and Jan Hoidekr. General framework for mining, processing and storing large amounts of electronic texts for language modeling purposes. *Language resources and evaluation*, 48(2):227–248, 2014.

Jan Lehečka and Jan Švec. Improving multi-label document classification of Czech news articles. In *International Conference on Text, Speech, and Dialogue*, pages 307–315. Springer, 2015.

Jan Lehečka and Aleš Pražák. Online LDA-based language model adaptation. In *International Conference on Text, Speech, and Dialogue*, pages 334–341. Springer, 2018.

List of Author's Other Works

Jan Lehečka. JMZW: detekce významných slov mimo slovník. In *SVK 2011 - magisterské a doktorské studijní programy, sborník rozšířených abstraktů*, pages 75–76, 2011. `http://svk.fav.zcu.cz/download/sbornik_svk_2011.pdf`.

Jan Švec, Aleš Pražák, Jan Hoidekr, Lucie Skorkovská, Jan Vavruška, Jan Lehečka, Dan Pressl, Petr Stanislav, and Pavel Ircing. Výzkumná zpráva projektu Jazykové modelování z webu 2011. Technical report, FAV, ZČU, Plzeň, 2011a.

Jan Švec, Aleš Pražák, Jan Hoidekr, Lucie Skorkovská, Jan Vavruška, Jan Lehečka, Dan Pressl, Petr Stanislav, Daniel Soutner, Pavel Ircing, and Jakub Kanis. Technologie pro multimediální archiv a jazykové modelování, 2011b.

Jan Lehečka. Detekce slov s nepravidelnou výslovností v českém textu. Master thesis, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Plzeň, 2012a. URL `https://theses.cz/id/dg9tyu`.

Jan Lehečka. Detekce slov s nepravidelnou výslovností v českém textu. In *SVK 2012 - magisterské a doktorské studijní programy, sborník rozšířených abstraktů*, pages 77–78, 2012b. `http://svk.fav.zcu.cz/download/sbornik_svk_2012.pdf`.

Jan Lehečka. Příprava dat pro online adaptaci LM. In *SVK 2014 - magisterské a doktorské studijní programy, sborník rozšířených abstraktů*, pages 77–78, 2014. `http://svk.fav.zcu.cz/download/sbornik_svk_2014.pdf`.

Jan Lehečka. The influence of thresholding strategy on multi-label topic identification. In *SVK 2015 - magisterské a doktorské studijní programy, sborník rozšířených abstraktů*, pages 91–92, 2015. `http://svk.fav.zcu.cz/download/sbornik_svk_2015.pdf`.

Jan Lehečka. Neuronové sítě v úloze identifikace tématu z textu. In *SVK 2016 - magisterské a doktorské studijní programy, sborník rozšířených abstraktů*, pages 95–96, 2016. `http://svk.fav.zcu.cz/download/sbornik_svk_2016.pdf`.

Jan Lehečka. Online topic-based language model adaptation. Report for state doctoral examination, University of West Bohemia, Faculty of Applied Sciences, 2016.

Jan Lehečka. Adaptace jazykového modelu v úloze automatického titulkování živě vysílaných sportovních přehledů. In *SVK 2018 - magisterské a doktorské studijní programy, sborník rozšířených abstraktů*, pages 60–61, 2018. `http://svk.fav.zcu.cz/download/sbornik_svk_2018.pdf`.

Zdeněk Loose, Aleš Pražák, Jan Lehečka, and Vlasta Radová. Využití skriptů pořadů pro zvýšení kvality živých titulků, 2018. `http://www.kky.zcu.cz/cs/sw/ScriptSystem`.