Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

FACULTY OF ELECTRICAL ENGINEERING, AUTOMATICS, COMPUTER SCIENCE AND BIOMEDICAL ENGINEERING

DEPARTMENT OF APPLIED COMPUTER SCIENCE

Master of Science Thesis

Zastosowanie morfosyntaktycznych i semantycznych modeli językowych w automatycznym rozpoznawaniu mowy

Application of Morphosyntactic and Semantic Language Models in Automatic Speech Recognition

Glossary

F_0 Fundamental frequency

$f(z)$ The sigmoid activation function

$g(z)$ The softmax activation function

$\hat{H}$ Cross-entropy

H Entropy

h_i Word history for w_i

φ_i Equivalence class of word history h_i

$\pi(w_i)$ Class mapping function

$P(W|Y)$ Probability of a word sequence W being produced from acoustic evidence Y

q^w Pronunciation of word w

$\hat{W}$ the word sequence most likely to have generated the sequence of acoustic features

Y the sequence of observed acoustic features

Acronyms

AM Acoustic Model

ASR Automatic Speech Recognition

BPTT Backpropagation Through Time

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DNN Deep Neural Network

FIR Finite Impulse Response

GMM Gaussian Mixture Model

HMM Hidden Markov Model

IDFT Inverse Discrete Fourier Transform

LM Language Model

LVCSR Large Vocabulary Continuous Speech Recognition

MFCC Mel Frequency Cepstral Coefficient

MLE Maximum Likelihood Estimation

NKJP The National Corpus of Polish

OOV Out of Vocabulary

POS Part of Speech

RNN Recurrent Neural Network

WER Word Error Rate

WERR Word Error Rate Reduction

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Automatic speech recognition

Automatic Speech Recognition (ASR) can be defined as "*independent, computer-driven transcription of spoken language into readable text in real time*" [56, 24]. Although this process is performed almost effortlessly by the human brain, it is extremely difficult to reverse engineer[1] Large Vocabulary Continuous Speech Recognition (LVCSR) falls into two categories: speech transcription and speech understanding. The former aims to find the exact orthographic transcription of analysed utterance, while the latter aims to find its meaning. The scope of this thesis is limited to speech transcription, as it allows to measure the performance of language models using well-established metrics.

1.1.1. Bayesian framework

In general, the recogniser tries to determine the word sequence $\hat{W} = w_1, \ldots, w_L$ out of all possible hypotheses W which is most likely to have generated the sequence of observed acoustic features $Y = y_1, \ldots, y_T$:

$$\hat{W} = \arg\max_{W} P(W|Y). \tag{1.1}$$

The conditional probability can be rearranged using the Bayes rule:

$$P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)} \propto P(Y|W)P(W). \tag{1.2}$$

$P(W)$ is estimated by the Language Model (LM), while $P(Y|W)$ is evaluated using the Acoustic Model (AM). Typical ASR systems use Hidden Markov Models (HMMs) to represent the sequential structure of speech signal and Gaussian Mixture Models (GMMs) to model the emission distribution of HMMs [2, 7].

[1] The observation that low-level sensorimotor skills require far more computational resources than high-level reasoning is known as the Moravec's paradox and has been formulated independently by several artificial intelligence researchers in the 1980s [38].

1.1.2. Feature extraction

Speech is a non-stationary process, so to represent it as a succession of discrete states, it is assumed that its properties are constant over a short period of time. Under this assumption, it is possible to extract statistically meaningful acoustic parameters from a sampled speech waveform. One of the most popular types of features are Mel Frequency Cepstral Coefficients (MFCCs). The cepstrum is formally defined as the Inverse Discrete Fourier Transform (IDFT) of the log magnitude of the Discrete Fourier Transform (DFT) of the signal:

$$c[n] = \sum_{n=0}^{N-1} \log \left(\left| \sum_{n=0}^{N-1} x[n] e^{\frac{-2\pi i k n}{N}} \right| \right) e^{\frac{2\pi i k n}{N}}. \tag{1.3}$$

In ASR applications, Discrete Cosine Transform (DCT) is commonly used in place of IDFT. A useful property of the cepstral analysis is that it enables to deconvolve the glottal source waveform of particular fundamental frequency F_0 from the vocal tract filter and extract the vocal tract properties, which carry most information about the phone being produced. The algorithm for calculating MFCCs consists of the following steps:

1. Preemphasis and windowing
2. DFT
3. Conversion to log mel scale
4. DCT
5. Calculating the delta, double delta, and energy coefficients

First, the speech waveform is subjected to high-frequency preemphasis in order to compensate for lip radiation and the attenuation of high frequencies caused by the sampling process [52]. Typically, the signal is passed through a high-pass Finite Impulse Response (FIR) filter:

$$H(z) = 1 - \frac{a}{z}, \tag{1.4}$$

where $0.9 \leq a \leq 1.0$. The signal is then segmented into a sequence of frames using 20-30 millisecond windows with about 50% overlap. The Hamming window is commonly used, as it allows to avoid discontinuities at the boundaries:

$$w[n] = \begin{cases} 0.54 - 0.46 \cos(\frac{2\pi n}{L}) & 0 \leq n \leq L-1 \\ 0 & \text{otherwise.} \end{cases} \tag{1.5}$$

In the next step, the DFT is applied. The resulting spectrum carries the information the amounts of energy in N discrete frequency bands:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-2\pi i k n}{N}}. \tag{1.6}$$

The DFT bands are evenly-spaced, but many studies confirm that human hearing is not equally sensitive at all frequencies – above about 500 Hz increasingly large intervals are judged by listeners to produce equal pitch increments [54, 15]. It was shown that simulating this property of the human brain during the feature extraction improves the ASR performance. The signal is therefore passed through a bank of 26 triangular filters spaced linearly below 1000 Hz and logarithmically above. This corresponds to a following mapping between the raw acoustic frequency f and the mel frequency m [39]:

$$m = 1127 \ln(1 + \frac{f}{700}). \tag{1.7}$$

Another property of the human hearing is that it is less sensitive to variations in signal level at high amplitudes. To model that response, the logarithm of each of the mel spectrum values is taken.

Finally, the DCT of the log filterbank energies is calculated. Higher values on the cepstrum x-axis represent the glottal pulse, while lower values correspond to the vocal tract characteristic. Generally, the MFCCs are formed from the first 12 cepstral values.

In addition to the 12 cepstral coefficients for each frame, the energy within a frame and the variability between frames are also taken into account, because they provide useful cues for phone identity. For each of the 13 features (12 cepstral features and energy), the delta (velocity) and double delta (acceleration) features are calculated, resulting in a total of 39 MFCCs [26].

1.1.3. Acoustic model

The posterior probability $P(Y|W)$ in Equation 1.2 is estimated using an acoustic model, which represents the relationship between feature vector sequences and some linguistic units. Most acoustic models are based on HMMs, although other techniques, such as neural networks, segmental models, and conditional random fields, have also been applied successfully [61, 60, 37]. Considering the number of words in a typical language, it is impractical to train a separate HMM for each word. For this reason, sub-word units are used, generally phoneme-sized. Each spoken word w from sequence W is decomposed into a sequence of K_w basic sounds, called pronunciation:

$$q^w = q_1, \ldots, q_K. \tag{1.8}$$

Since a word can have multiple pronunciations, the posterior probability has to be calculated as a sum over all possible pronunciations:

$$P(Y|W) = \sum_Q P(Y|Q)P(Q|W), \tag{1.9}$$

where Q is the particular sequence of pronunciations:

$$P(Q|W) = \prod_{i=1}^{L} P(q^{w_i}|w_i). \tag{1.10}$$

In practice, the number of alternative pronunciations for each word w_i is small, which makes the summation in Equation 1.9 feasible.

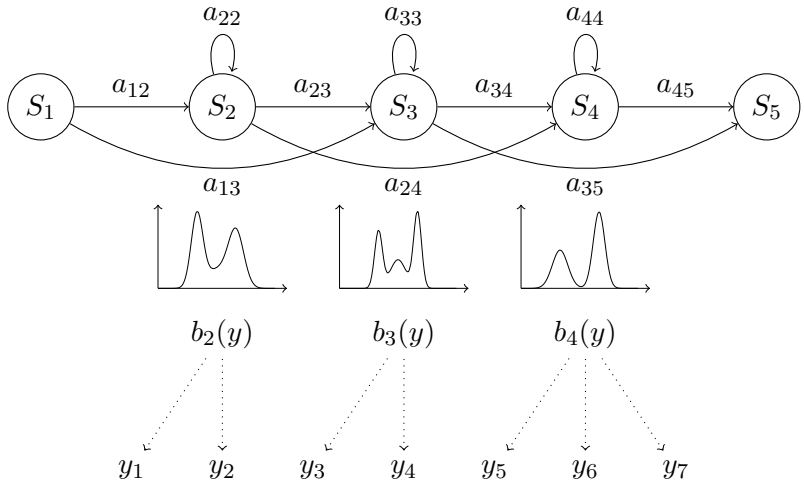

Figure 1.1: A Hidden Markov Model with five states.

S_1 and S_5 are non-emitting entry and exit states, a_{ij} denotes the probability of transition from state S_i to S_j, b_i is the probability density function associated with state S_i, and $Y = [y_1, \ldots, y_7]$ is the observation sequence. The depicted HMM is an example of a Bakis model, which allows self-transitions, transitions to adjacent states, and skipping of individual states within a sequence. It is able to capture temporal extensions and reductions of the signal, while keeping the number of parameters low in comparison with more flexible topologies, such as left-to-right and ergodic [14].

To calculate $P(Y|W)$, we represent each base phone q by an HMM of the form shown in Figure 1.1, described by the following elements:

1. N – number of states,

2. $S = \{S_1, \ldots, S_N\}$ – the set of states,

3. θ_t – state at time t,

4. $\pi = \{\pi_i\}$ – initial state distribution:

$$\pi_i = P(\theta_1 = S_i) \qquad (1 \leqslant i \leqslant N),$$

5. $A = \{a_{ij}\}$ – transition matrix describing the probability of transition from state S_i to state S_j:

$$a_{ij} = P(\theta_{t+1} = S_j | \theta_t = S_i) \qquad (1 \leqslant i, j \leqslant N),$$

6. $B = \{b_i(y)\}$ – observation distribution associated with state S_i, usually modeled by an affine combination of M Gaussians, called a Gaussian Mixture Model (GMM):

$$b_i(y) = \sum_{k=1}^{M} w_{ik}\mathcal{N}(y, \mu_{ik}, \Sigma_{ik}) \tag{1.11}$$

$$(1 \leqslant i \leqslant N, \quad w_{ik} \geqslant 0, \quad \sum_{k=1}^{M} w_{ik} = 1)$$

where μ_{ik} and Σ_{ik} are the mean vector and covariance matrix associated with state S_i and Gaussian k, while w_{ik} is the linear combination coefficient [25].

Every time step, the HMM makes a transition from its current state to one of its connected states, generating a feature vector according to the probability density function associated with the state being entered. Two important assumptions follow:

1. *The first-order Markov assumption* – the probability of a state θ_t at time t is only dependent on the previous state θ_{t-1}:

$$P(\theta_t|\theta_{t-1}, \ldots, \theta_1) = P(\theta_t|\theta_{t-1}). \tag{1.12}$$

2. *The output independence assumption* – every observation is conditionally independent of all other observations:

$$P(y_t|y_{t-1}, \ldots, y_1) = P(y_t|y_{t-1}). \tag{1.13}$$

These assumptions significantly simplify the calculation of the posterior probability $P(Y|W)$ from Equation 1.9. We can now define it in terms of the acoustic model parameters [30]:

$$P(Y|Q) = \sum_{\Theta} P(\Theta, Y|Q), \tag{1.14}$$

where $\Theta = \theta_0, \ldots, \theta_{T+1}$ is a state sequence and

$$P(\Theta, Y|Q) = a_{\theta_0\theta_1} \prod_{t=1}^{T} b_{\theta_t}(y_t) a_{\theta_t\theta t+1}. \tag{1.15}$$

In the above equation, θ_0 and θ_{T+1} are non-emitting states (entry and exit). The model parameters $\lambda = \{a_{ij}, b_i(y)\}$ can be estimated from a corpus of training data using the forward-backward algorithm [3]. For a detailed explanation of the algorithm, see [48].

The described technique has an important shortcoming – decomposing each vocabulary word into a sequence of independent monophones ignores the context-dependent variation that exists in real speech. A common solution is to use triphones, i.e. phonemes in their left and right context. However, this approach results in a data sparsity problem – for N base phones, there are N^3 potential triphones. This can be avoided by mapping the complete set of logical triphones L to a reduced set of physical models P by clustering and tying together the parameters in each cluster. The clustering is usually done at state-level using phonetically driven decision trees [18].

HMMs remained a *de facto* standard for acoustic modelling until the advent of Deep Neural Networks (DNNs), first successfully used for acoustic modelling in 2012, as a result of cooperation between Google, Microsoft, IBM, and the University of Toronto. According to Richard Rashid, former head of Microsoft Research, it was "*the most dramatic change in accuracy since 1979*" [32]. In [22], DNNs with many hidden layers have been shown to outperform GMMs on a variety of challenging speech recognition tasks, sometimes by a large margin. The unprecedented gains in error reduction were further confirmed in [44], where it is reported that DNNs can consistently achieved about 25-30% relative error reduction over the best discriminatively trained GMMs.

1.1.4. Language models

Language modelling lies at the core of many natural language processing tasks, such as speech recognition and synthesis, document classification, grammar and spelling checking, parsing, machine translation, and information retrieval. In general, the task of LMs is to estimate the likelihood of a sequence of words. This information can then be used to reject invalid sentences or resolve ambiguities. In case of automatic speech recognition the language model guides and constrains the search among alternative word hypotheses [19].

In a Bayesian framework presented in equation 1.2, the language model estimates the a priori likelihood by assigning probability $P(W)$ to each word sequence $W = w_1, \ldots, w_n$ such that $\sum_W P(W) = 1$. Since the search is usually performed unidirectionally, $P(W)$ can be formulated as a chain rule:

$$P(W) = \prod_{i=1}^{n} P(w_i|h_i), \tag{1.16}$$

where $h_i = w_1, \ldots, w_{i-1}$ is the word history for w_i, often reduced to an equivalence class φ_i:

$$P(w_i|h_i) \approx P(w_i|\Phi(h_i)) = P(w_i|\varphi_i). \tag{1.17}$$

Good equivalence classes maximise information about the next word w_i given its history, but usually require a vast quantity of example sequences. The development of effective statistical language models is therefore limited by the availability of representative and machine readable text corpora [49].

1.2. Polish Language

1.2.1. Introduction

Although ASR has been an active field of research for several decades, for a long time most of the effort has been focused on English. Many important techniques were developed as a result of research programmes financed by the American Defense Advanced Research Projects Agency and

involving companies and institutions like IBM, AT&T Bell Labs, Institute for Defense Analysis, Princeton University, and CMU. In the nineties, numerous ASR systems originally developed for English had been successfully ported to other languages, such as French, German, Japanese, and Mandarin Chinese [5]. The recently introduced DNN models were also first trained and benchmarked on English corpora, and later used for other languages [22]. Although this suggest that similar modelling assumptions can hold across languages, Polish and other inflected languages still pose a formidable challenge for speech technology researchers, due to their complex grammar, rich inflection, and a large set of phonetically similar prefixes and suffixes. This section describes the features of Polish in the context of ASR and explains where the main problems arise and how can they can be tackled.

1.2.2. Slavic Languages

Polish is an Indo-European language used by about 40 million speakers in Poland and around the world. It belongs to the Slavic group, containing about 20 languages and dialects spoken by over 400 million people in Central, Southern, and Eastern Europe, as well as in the Asian part of Russia [27]. This family of languages is traditionally divided into three main branches:

- West Slavic – Polish, Czech, and Slovak,
- South Slavic – Serbian, Croatian, Bulgarian, Slovene, and Macedonian,
- East Slavic – Russian, Ukrainian, and Bellarusian.

In the field of speech technology, research and development efforts has been focused primarily on Czech [41, 43], Polish [62, 64], Russian [27], and Slovak [29].

1.2.3. Inflection

Polish, like other Slavic languages, exhibits a large degree of inflection. This means that a lexical unit (lexeme) modifies its basic form (lemma) depending on grammatical, morphological, and contextual relations. Grammatical classes, e.g. nouns, adjectives, adverbs, have an associated set of grammatical categories, such as number, case, or aspect. Major grammatical categories and their values are:

- number – singular, plural,
- case – nominative, genitive, dative, accusative, instrumental, locative, vocative,
- gender – human masculine, animate masculine, inanimate masculine, feminine, neuter,
- person – first, second, third,
- degree – positive, comparative, superlative,

- aspect – imperfective, perfective,
- negation – affirmative, negative.

For a given class, a grammatical category can be morphological (all lexemes from that class are subject to inflection with respect to that category), or lexical (for each lexeme belonging to that class, all forms of that lexeme have the same value of that category, although that value differs across lexemes). For example, a noun changes its orthographic and phonetic form with respect to number and case, but its gender does not change, as shown in Table 1.1. Similarly, Table 1.2 shows that a verb in past tense is subject to conjugation with respect to person, gender, and number, but its aspect is a fixed property.

Table 1.1: Inflection of the feminine noun *mowa* (speech) with respect to case and number.

	singular	plural
nominative	mowa	mowy
genitive	mowy	mów
dative	mowie	mowom
accusative	mowę	mowy
instrumental	mową	mowami
locative	mowie	mowach
vocative	mowo	mowy

Table 1.2: Conjugation of the verb *rozpoznawać* (recognise) in past tense and perfective aspect, with respect to number (singular, plural), person (first, second, third), and gender (masculine, feminine, neuter)

(a) singular

	first	second	third
masculine	rozpoznałem	rozpoznałeś	rozpoznał
feminine	rozpoznałam	rozpoznałaś	rozpoznała
neuter	-	-	rozpoznało

(b) plural

	first	second	third
masculine	rozpoznaliśmy	rozpoznaliście	rozpoznali
feminine	rozpoznałyśmy	rozpoznałyście	rozpoznały
neuter	rozpoznałyśmy	rozpoznałyście	rozpoznały

Examples presented in Tables 1.1 and 1.2 already hint at the most important problem in the ASR of inflected languages: the size of the lexicon. While an English noun usually has just two forms (singular and plural), its Polish equivalent can have fourteen distinct forms. In case of verbs or adjectives, the difference is even more pronounced. New words can be created not only by changing the endings, but also by adding multiple prefixes and suffixes, as well as modifying the stem itself. Even negative, comparative, or superlative forms of an adjective are usually distinct lexical items. This is a serious challenge in building an ASR system, as the number of distinct word-forms can easily exceed one million. In case of English, 135 vocabulary items are needed to account for half of the Brown Corpus [13], and an inventory of 50000 words is enough to achieve 99% coverage [33]. In case of inflected language, the word building mechanisms allow for producing virtually unlimited number of words, while even a vocabulary of one million items is too large to be processed in real time [42]. The most straightforward strategy of dealing with that problem is to limit the lexicon to words that occurred at least N times in the training corpus. There is obviously a trade-off between the size of the model and recognition accuracy.

1.2.4. Word order

From the ASR perspective, the most problematic grammatical feature of Polish and other Slavic language, is the relatively free word order. Thanks to rich morphology, the subject, verb, and adjectives may appear at almost arbitrary positions. The role of a word in the sentence can be inferred from its inflection, as there is a strong grammatical agreement between the parts of a sentence. The subject of a sentence must agree in gender, person, number, and case with all its modifiers. Moreover, verbs constrain the case of corresponding nouns, forming a predicate expression called a verbonominal collocation [57]. Therefore, the agreement relation often does not fit the left-to-right direction of language processing and frequently binds words which are not immediate neighbours. Because of this, n-gram models for Slavic languages will never be as efficient as in case of English, which imposes strict constraints on the relative order of words in a sentence. The mechanisms of grammatical agreement are too complex to be formalised into a set of machine-processable rules. A more common approach is to use class-based n-gram models instead of word-based models. The idea is to map each word from the corpus to a corresponding class, based on grammatical, morphological, or semantic features. The number of classes is significantly lower than the size of the lexicon, so such models are less prone to data sparsity problems and usually do not require smoothing (see Section 2.1.2).

1.2.5. Phonology

The Polish vowel system consists of six oral monophthongs and two nasal diphthongs, presented in Table A.1. There are several inconsistencies that make the orthographic to phonemic transcription challenging. Note for example that the nasal vowels, denoted by letters <ą> and

<ę>, are pronounced either as a mid-vowel followed by a nasalised labio-velar or palatal glide, or as a combination of an oral vowel and a nasal consonant. Moreover, letter <i> can denote a vowel /i/, as in *igła* /igwa/ (needle), have a purely orthographic function of marking the palatalisation of preceding consonant when followed by a vowel, as in *cieplo* /t͡ɕɛpwɔ/ (heat), or do both at the same time, for example in *cichy* /t͡ɕixɨ/ (silent). It can also denote a consonant /j/, as in *hiena* /xjɛna/ (hyena), or a combination of /j/ and /i/, for example in *naiwny* /najivnɨ/ (naïve). In contrast, the vowel /u/ can be denoted by two different letters: <ó> and <u>. Although there is no difference in articulation, they cannot be used interchangeably. It is a relic of the vowel length system, which is still present in Czech and Slovak, but has disappeared from Polish.

The Polish consonant system is far more complex, at least from the ASR perspective, mostly due to the distinction between the "rustling" laminal retroflex sounds (sz, ż, cz, dz), the corresponding "humming" alveolo-palatals (ś, ź, ć, dź), and "hissing" alveolars (s, z, c, dz). Table A.2 presents Polish consonants – their orthographic transcription, corresponding symbols of the International Phonetic Alphabet, examples, and English approximations [20].

1.3. Motivation and outline

This thesis presents a comparative analysis of morphosyntactic and semantic language models in the context of automatic speech recognition of Polish. It focuses on word n-grams, class n-grams, and neural probabilistic models. The goal of the experimental part is to find a model that is optimal in terms of the chosen metrics. The analysis of existing literature on the subject shows that language modelling of Polish is still in its incipient stage when compared to English. Most of the available works is limited to different variations of the n-gram model [31, 63]. Until recently, there were no studies on using neural networks for language modelling of Polish [16, 8]. A comparative review of existing approaches would allow to select the most promising directions of research and in the long-term perspective contribute to the development of efficient LVCSR systems for Polish. Moreover, the results could be directly translatable to other inflected languages.

The introduction has outlined the process of automatic speech recognition and the role of acoustic and language models in the Bayesian framework. Section 1.2 discusses some features of Polish that hinder the performance of standard n-gram models, in particular the rich morphology resulting in large lexicons and the relatively free word order. Chapter 2 is a comprehensive overview of language models used in the experimental part: word n-grams, class n-grams, and neural network models. Section 2.3 presents popular evaluation metrics for language models: perplexity, Word Error Rate (WER) and Word Error Rate Reduction (WERR). The first section of Chapter 3 describes the tools and resources used in the experiments with a special emphasis on the National Corpus of Polish (*Narodowy Korpus Języka Polskiego*, NKJP), the most comprehensive annotated collection of Polish texts, used as a source of training data in

the experimental part. The second section is a detailed description of the process of building the models – from obtaining and processing the data to the actual training. Chapter 4 is perhaps the most important part of the thesis, as it presents the results of the experiments – a comparison of all the models with respect to performance metrics described in Section 2.3. Chapter 5 contains a short summary of the results and discusses the implications of the conducted experiments, while at the same time pointing at some limitations of the methodology. It also proposes new directions for studies on language modelling of Polish and other similar languages.

2. Statistical language models

2.1. N-gram models

In Section 1.1.4 we defined the chain rule of probability, which lets us decompose the joint probability of a sequence of N words into a product of conditional probabilities. Let w_1^n denote $w_1, \ldots, w_n$. The chain rule can be formulated as

$$P(w_1^n) = \prod_{i=1}^{n} P(w_i|w_1^{i-1}) = \prod_{i=1}^{n} P(w_i|h_i). \tag{2.1}$$

However, there is no way of computing the probability of a word given a long history of preceding words. Estimating it from relative frequency counts is infeasible, if only for data sparsity reason. We can deal with this problem by using an n-gram model. An n-gram is simply a sequence of n words (or other modelling units) – an n-gram of size 1 is called a unigram, size 2 is a bigram, and size 3 is called a trigram. The n-gram model approximates the probability of a word given all the previous words with the probability of the word given $n-1$ preceding words:

$$P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1}). \tag{2.2}$$

The n-gram can be interpreted as a left-to-right Markov chain of order $n-1$, because the probability of the current state depends only on $n-1$ previous states. To put it differently, the n-gram models satisfies the $n-1$ order Markov assumption (see Section 1.1.3). In contrast to HMMs, the states of Markov chains are directly visible.

Another way to look at the n-gram model is through the notion of history equivalence class, introduced in Section 1.1.4. As indicated previously, it is impossible to obtain a probability estimate for every imaginable word history, but the number of parameters can be reduced by classifying word histories into equivalence classes. Indeed, in [24], language modelling is defined as the task of finding the best history classification function:

$$\Phi : h_i \mapsto \varphi_i = \Phi(h_i) = \Phi(w_1^{i-1}) \tag{2.3}$$

Finding an optimal history classification function is the matter of striking a fine balance between its predictive power and complexity. To be a useful predictor, an equivalence class has to maximise the information about next word, while minimising the dimensionality of the resulting

model. In case of n-gram models, the history classification function Φ simply limits the word history to the last $n-1$ elements:

$$\Phi(w_1^{i-1}) = w_{i-n+1}^{i-1}. \tag{2.4}$$

Two word histories are therefore equivalent if their last $n-1$ elements are identical. The trade-off between efficiency and complexity quickly becomes apparent when the value of n is increased – higher order n-grams typically exhibit lower perplexity (see Section 2.3.1) and better performance, but are more expensive in terms of time and memory.

2.1.1. Maximum Likelihood Estimation

The most common technique of estimating the n-gram probabilities from a text corpus is Maximum Likelihood Estimation (MLE). The procedure is very simple – the n-gram probability of a word w_i given the truncated history $\varphi_i = w_{i-n+1}^{i-1}$ is the observed frequency of the n-gram w_{i-n+1}^{i} normalised by the sum of observed frequencies of all n-grams sharing the same history:

$$P(w_i|\varphi) = \frac{C(\varphi, w_i)}{\sum_{w_j} C(\varphi, w_j)}. \tag{2.5}$$

Note that the sum in the denominator is equal simply to $C(\varphi_i)$, so the formula in Equation 2.5 can be more intuitively understood as the proportion of cases in which a particular equivalent word history φ_i is followed by the word w_i:

$$P(w_i|\varphi_i) = \frac{C(\varphi_i, w_i)}{C(\varphi_i)}. \tag{2.6}$$

This ratio is called the relative frequency. In the special case of unigram probability it can be calculated as the count of the particular word w_i normalised by the size of the corpus:

$$P(w_i) = \frac{C(w_i)}{\sum_{w} C(w)}. \tag{2.7}$$

Table 2.1 shows an example of calculating n-gram probabilities using a corpus of four phrases from Monty Python's notorious spam sketch. Inspecting it reveals a major problem with the MLE approach. Note that some probability estimates are equal to one, so using the model to generate random sentences would in many cases result in phrases taken verbatim from the corpus. Even worse, there are also probability estimates equal to zero. This means that because of the chain rule, the model will assign a zero probability to any sequence containing a known word in a new context. This includes one of the previous lines of the sketch:

$$P(\texttt{bacon}|\texttt{and}) = 0 \Rightarrow P(\texttt{<s> egg and bacon </s>}) = 0 \tag{2.8}$$

Both these artifacts are a consequence of data sparsity – the corpus is simply too small for the model to be an accurate representation of the language. The probability of observed items is

Table 2.1: Calculating selected bigram and trigram probabilities using maximum likelihood estimation [26]. The <s> and </s> tags denote the beginning and the end of the sentence.

<s> egg bacon and spam </s>
<s> egg bacon sausage and spam </s>
<s> spam bacon sausage and spam </s>
<s> spam egg spam spam bacon and spam </s>

$P(\text{bacon}\|\text{spam}) = \frac{2}{8}$	$P(\text{spam}\|\text{spam}) = \frac{1}{8}$	$P(\text{egg}\|\text{<s> <s>}) = \frac{2}{4}$
$P(\text{bacon}\|\text{egg}) = \frac{2}{3}$	$P(\text{spam}\|\text{<s>}) = \frac{2}{4}$	$P(\text{egg}\|\text{bacon and}) = 0$
$P(\text{bacon}\|\text{and}) = 0$	$P(\text{spam}\|\text{and}) = 1$	$P(\text{</s>}\|\text{and spam}) = 1$

overestimated, while the probability of unobserved items is underestimated. Although the example is obviously exaggerated, the problem persists in case of full-sized corpora. There are many techniques of correcting this bias by shifting the probability mass from frequent to previously unseen items. This process, called smoothing or discounting, is described in Section 2.1.2.

Zero probability is one problem, but there is also an issue of words that do not appear in the corpus at all. In speech recognition, Out of Vocabulary (OOV) tokens are inevitable and they need to be identified, because they contribute to recognition errors in surrounding words. More importantly, they are often information-rich nouns, such as proper names, domain-specific notions, or foreign words. Language models can be used to facilitate the process of OOV token detection by incorporating the information about unknown words into the training process. All words that appear in the LM training data, but do not appear in the ASR vocabulary, are simply substituted by the unknown word token <ign>. These pseudo-words are then treated like any other regular word in the corpus, similarly to <s> and </s>.

Another conclusion that can be drawn from the example in Table 2.1 is that the model is only as representative as the corpus it is trained on. This is especially important in LVCSR, where the model not only has to represent non-domain-specific vocabulary, but also focus on spoken language, which is often very different from writing. ASR systems are often trained on written texts, because this kind of data is usually readily available, but using speech transcripts can lead to better results with less training data [12]. This idea is further explored in the experimental part of the thesis.

2.1.2. Smoothing

Smoothing is a way of dealing with the zero probability problem. In principle, it is the process of adjusting the MLE estimates to produce more accurate probabilities [11]. High probabilities are adjusted downwards and low probabilities are adjusted upwards, so the resulting distribution is more uniform. The simplest method of smoothing is additive smoothing. The idea is to add a

constant δ to every n-gram count:

$$P_{\text{ADD}}(w_i|\varphi i) = \frac{\delta + C(\varphi_i w_i)}{\delta|V| + C(\varphi_i)}, \tag{2.9}$$

where typically $0 < \delta \leq 1$. This method is easy to implement, but has been shown in [17] to generally perform poorly.

Katz back-off

Katz back-off builds on the Good-Turing estimate, based on the *symmetry requirement* which states that two events which occur the same number of times in the sample must have equal probabilities [59]. The idea is to adjust the count of n-grams that occur c times using the counts of n-grams that occur $c+1$ times – in particular, to estimate the probability of unseen n-grams using the singleton counts. Let n_c denote the number of n-grams that occur c times in the sample. For each count c, an adjusted count $\hat{c}$ is computed:

$$\hat{c} = (c+1)\frac{n_{c+1}}{n_c}. \tag{2.10}$$

From Equation 2.10 it follows that the total number of counts that will be assigned to n-grams with zero counts is equal to the number of singletons. The updated probability estimate can be calculated by normalising the updated count by the total number of tokens. For an n-gram α with c_α counts:

$$P_{\text{GOOD}}(\alpha : C(\alpha) = c_\alpha) = \frac{\hat{c}_\alpha}{\sum_c c n_c}. \tag{2.11}$$

The problem with using plain Good-Turing estimates for discounting is that n_{c+1} quite often equals zero for high c. Katz back-off extends the idea behind Good-Turing estimation by using lower-order distributions to better reallocate the count mass subtracted from nonzero counts. For example, while adjusting the counts of bigrams, the unigram distribution is used:

$$C_{\text{KATZ}}(w_{i-1}, w_i) = \begin{cases} d_c c & \text{if } c > 0 \\ \alpha(w_{i-1}) P_{\text{MLE}}(w_i) & \text{if } c = 0. \end{cases} \tag{2.12}$$

The discount coefficient d_c depends on the the n-gram count c. Counts larger than some arbitrary threshold k are not discounted ($d_c = 1$). The value of d_c when $c \leq k$ is chosen so that the resulting discount is proportional to the Good-Turing discount:

$$1 - d_c = \mu(1 - \frac{\hat{c}}{c}). \tag{2.13}$$

Furthermore, the total number of counts discounted in the n-gram distribution should be equal to the total number of counts assigned to zero-count n-grams according to the Good Turing estimate (see Equation 2.10):

$$\sum_{c=1}^{k} n_c(1 - d_c)c = n_1. \tag{2.14}$$

The only solution that satisfies the constraints formulated in Equation 2.13 and 2.14 is given by:

$$d_c = \frac{\frac{\hat{c}}{c} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}. \tag{2.15}$$

Katz back-off for higher-order n-gram is defined recursively, where the unigram model is taken to be the MLE unigram model to end the recursion [59]. The updated probability P_{KATZ} of word w_i given truncated history w_{i-n+1}^{i-1} is given by:

$$P_{\text{KATZ}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{C(w_{i-n+1}^{i})}{C(w_{i-n+1}^{i-1})} & \text{if } C(w_{i-n+1}^{i}) > k \\ d_{C(w_{i-n+1}^{i})} \cdot \frac{C(w_{i-n+1}^{i})}{C(w_{i-n+1}^{i-1})} & \text{if } 1 \leq C(w_{i-n+1}^{i}) \leq k \\ \alpha(w_{i-n+1}^{i-1}) \cdot P_{\text{KATZ}}(w_i|w_{i-n+2}^{i-1}) & \text{if } C(w_{i-n+1}^{i}) = 0, \end{cases} \tag{2.16}$$

where:

$$\alpha(w_{i-n+1}^{i-1}) = \frac{1 - \sum_{w_i:C(w_{i-n+1}^{i})>0} P_{\text{KATZ}}(w|w_{i-n+1}^{i-1})}{\sum_{w_i:C(w_{i-n+1}^{i})=0} P_{\text{KATZ}}(w|w_{i-n+2}^{i})}. \tag{2.17}$$

Note that the value of α is chosen so that the total number of counts in the distribution is unchanged. For details on the derivation of the back-off weight α and the discount coefficient d_c, see [11].

Kneser-Ney smoothing

The Kneser-Ney smoothing is an extension of absolute discounting, which involves subtracting a fixed discount $\delta \in (0, 1)$ from each nonzero count:

$$P_{\text{ABS}}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{C(w_{i-n+1}^{i}) - \delta, 0\}}{\sum_{w_i} C(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i}})P_{\text{ABS}}(w_i|w_{i-n+2}^{i-1}). \tag{2.18}$$

The parameter λ is taken so that the resulting distribution sums to one:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{\delta}{\sum_{w_i} C(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\ \bullet). \tag{2.19}$$

The expression $N_{1+}(\varphi\ \bullet)$, denotes the number of unique words that follow the history φ:

$$N_{1+}(w_{i-n+1}^{i-1}\ \bullet) = |\{w_i : C(w_{i-n+1}^{i}) > 0\}. \tag{2.20}$$

The parameter δ is chosen using held-out estimation. In [40], it is estimated as

$$\delta = \frac{n_1}{n_1 + 2n_2}. \tag{2.21}$$

The idea behind Kneser-Ney smoothing is that since the lower-order model is only necessary when the count is small or zero in the higher-order model, it should be optimised for that purpose. In most other algorithms, the lower-order distribution is just a smoothed version of the

MLE distribution. In Kneser-Ney smoothing, the unigram probability is not proportional to the number of occurrences of the word, but the number of words it follows:

$$P_{\text{KN}}(w_i) = \frac{N_{1+}(\bullet\ w_i)}{N_{1+}(\bullet\ \bullet)}, \tag{2.22}$$

where $N_{1+}(\bullet\ w_i)$ is the number of different words that precede w_i in the training data and $N_{1+}(\bullet\ \bullet)$ is the number of unique bigrams with nonzero count:

$$N_{1+}(\bullet\ w_i) = |w_{i-1} : C(w_{i-1}, w_i) > 0|, \tag{2.23}$$

$$N_{1+}(\bullet\ \bullet) = |(w_{i-1}, w_i) : C(w_{i-1}, w_i) > 0|. \tag{2.24}$$

The general formula for the unmodified Kneser-Ney smoothing is

$$P_{\text{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{C(w_{i-n+1}^{i}) - \delta, 0\}}{\sum_{w_i} C(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i}})\frac{N_{1+}(\bullet\ w_{i-n+2}^{i})}{N_{1+}(\bullet\ w_{i-n+2}^{i-1}\ \bullet)}, \tag{2.25}$$

where:

$$N_{1+}(\bullet\ w_{i-n+2}^{i}) = |w_{i-n+1} : C(w_{i-n+1}^{i}) > 0|, \tag{2.26}$$

$$N_{1+}(\bullet\ w_{i-n+2}^{i-1}\ \bullet) = |w_{i-n+1,w_i} : C(w_{i-n+1}^{i}) > 0|. \tag{2.27}$$

The Chen and Goodman's modification of the original Kneser-Ney algorithm uses three different discount values, δ_1, δ_2, δ_3, that are applied to n-grams with one, two, and three or more counts, respectively [11]. While estimating $P_{\text{KN}}(w_i|w_{i-n+1}^{i-1})$, the parameter δ simply becomes a function of $C(w_{i-n+1}^{i-1})$:

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ \delta_1 & \text{if } c = 1 \\ \delta_2 & \text{if } c = 2 \\ \delta_{3+} & \text{if } c \geq 3. \end{cases} \tag{2.28}$$

The distribution must still sum to one, so the new value of α is:

$$\alpha(w_{i-n+1}^{i-1}) = \frac{\delta_1 N_1(w_{i-n+1}^{i-1}\ \bullet) + \delta_2 N_2(w_{i-n+1}^{i-1}\ \bullet) + \delta_{3+} N_{3+}(w_{i-n+1}^{i-1}\ \bullet)}{\sum_{w_i} C(w_{i-n+1}^{i})}. \tag{2.29}$$

The modified version has been shown to significantly outperform the original Kneser-Ney algorithm, because the optimal average discount for n-grams with one or two counts is different from the optimal average discount for n-grams with higher counts. Chen and Goodman provide estimates of the optimal values for δ_1, δ_2, and δ_3:

$$\begin{aligned} Y &= \frac{n_1}{n_1 + 2n_2} \\ \delta_1 &= 1 - 2Y\frac{n_2}{n_1} \\ \delta_2 &= 2 - 3Y\frac{n_3}{n_2} \\ \delta_{3+} &= 3 - 4Y\frac{n_4}{n_3}. \end{aligned} \tag{2.30}$$

2.1.3. Word n-grams

So far, while describing the n-gram models, we used the terms "word" and "modeling unit" interchangeably. However, depending on the application, n-grams can use sub-lexical units, such as phonemes, letters, and syllables, or supra-lexical units, such as Part of Speech (POS) tags. In case of ASR, words are the most popular choice for the modeling units, because word-based models can be easily incorporated into the search process described in section 1.1.4. For example, consider this famous quote:

`open the pod bay doors.`

We would like to estimate the probability of next word being `hal`. Using a bigram model, we get:

$$P(\texttt{hal}|\texttt{open the pod bay doors}) \approx P(\texttt{hal}|\texttt{doors}). \tag{2.31}$$

Truncating the history to $n-1$ units drastically reduces the number of free parameters. Although this number is still enormous – for a bigram model and a vocabulary of 50000 words, there are potentially two and a half billion parameters – it is at least feasible to estimate the probability of the entire sequence using the chain rule from Equation 2.1:

$$\begin{aligned} &P(\texttt{<s> open the pod bay door hal </s>}) \approx \\ &\approx P(\texttt{open}|\texttt{<s>})P(\texttt{the}|\texttt{open})P(\texttt{pod}|\texttt{the})P(\texttt{bay}|\texttt{pod})P(\texttt{door}|\texttt{bay})P(\texttt{hal}|\texttt{door})P(\texttt{</s>}|\texttt{hal}). \end{aligned} \tag{2.32}$$

There are several disadvantages of word-based n-gram models. The first one is the obviously unrealistic Markov's assumption. Language dependencies often span across far more than just three or four words, especially in inflected languages. Consequently, low-order word-based n-grams may do a poor job at disambiguating grammatically invalid sentences, because an incorrect sentence can be constructed from a sequence of valid short n-grams. Take this quote from Yoda as an example:

`<s> if no mistake you have made losing you are </s>`

A bigram model would likely assign a relatively high probability to this phrase, because each pair of consecutive words is entirely plausible. In the above example, using a trigram model would probably yield better results, but at a considerable cost – even for a lexicon of 50 000 words (far too few for modelling any inflected language), switching from bigram to trigram introduces more than one hundred trillions potential parameters! In short, there is always a trade-off between the quality of predictions and the amount of data required to calculate the probability estimates. Training and storing higher-order word n-grams can be simply infeasible due to data sparsity and memory constraints. Another issue with word-based n-grams is that the truncation of word history may lead to unintuitive or inconsistent behaviour. This problem is especially pronounced in case of languages with weak constraints on word order – permutations of the same sequence

may be assigned different probability estimates despite being grammatically and semantically equivalent. Furthermore, very similar word histories may lead to different predictions. Consider these two sequences:

`she graduated from`

`she graduated very recently from`.

From the point of view of predicting the next word, these two histories are very similar. However, they would be considered completely different by a trigram word model [59]. In the second case, we would need at least a 5-gram model to capture the intuition that the next word is probably a name of a university.

Although word-based n-gram language models have been outperformed by deep neural networks in terms of perplexity and word error rate, they are still commonly used in ASR systems, as they strike a fine balance between effectiveness and simplicity. They are straightforward to train and can be easily used to guide the search through the lattice of word hypotheses. However, in case of inflected languages, the problem of data sparsity is significantly amplified and therefore using words as the modeling unit is not necessarily the best choice.

2.1.4. Class-based n-gram models

Class-based models address the problem of data sparsity by reducing the number of parameters. The idea is to cluster words with similar statistical distributions or linguistic properties into groups. A class n-gram model with a vocabulary of size V and C classes has $V - C$ word emission parameters and $C^n - 1$ independent n-gram parameters [10]. Class-based models always have fewer parameters than analogous word-based models, and therefore their parameters can be more accurately estimated, as most classes will be well represented in the corpus. Class n-gram models are constructed in a similar fashion as word-based models, except that words are mapped to equivalence classes:

$$\pi : w_i \mapsto \pi(w_i). \tag{2.33}$$

The class mapping π can be deterministic or probabilistic. An example of a deterministic mapping is the `<ign>` token described in Section 2.1.1 – a word either appears in the vocabulary or not, so the mapping is unambiguous. In contrast, clustering based on linguistic knowledge is often ambiguous, because the same word can belong to different grammatical categories, depending on the context. In case of a deterministic mapping, the probability of a word given its history can be calculated as the product of the probability of a particular word given its class (word emission probability) and the probability of a certain class given a history of $n-1$ classes:

$$P_{\text{CLASS}}(w_i|w_{i-n+1}^{i-1}) = P(w_i|\pi(w_i))P(\pi(w_i)|\pi(w_{i-n+1}), \ldots, \pi(w_{i-1})). \tag{2.34}$$

The class n-gram probabilities can be calculated from the corpus using MLE, similarly to word n-gram probabilities. The word emission probability is often dropped, but can be otherwise

estimated as the relative frequency of the form:

$$P(w_i|c_i) = \frac{C(w_i)}{C(\pi(w_i))}. \tag{2.35}$$

In case of a probabilistic mapping, there are multiple realisations of the same word history, as each word can belong to several classes. Therefore, the prediction of the current word requires a summation over the probabilities of all the possible realisations [40]. The methods for finding the class mapping function fall roughly into two categories – knowledge-based and data-driven. The former approach takes advantage of prior linguistic knowledge. Examples include clustering based on POS tags or semantic functions. This method usually requires additional resources such as tagged corpora, wordnets, or automatic morphological taggers. The data-driven clustering generally uses a greedy algorithm to automatically cluster words in such a way that the perplexity of the corpus is minimised. The class-based models used in this work are deterministic, knowledge-based morphosyntactic models.

2.2. Neural network models

For decades, n-grams have been the standard approach to statistical language modelling. Although many alternative techniques were proposed, the improvements in performance usually came at the cost of computational complexity [36]. The neural network based language models were introduced in [4] and motivated by the observation that the word-based n-gram models have two major setbacks – they are unable to capture longer contexts and they ignore the similarity between words.

2.2.1. The feedforward architecture

The neural models use a distributed representation to deal with the high dimensionality of the training data. The idea is to represent each word as a real-valued vector in $\mathbb{R}^m$ and then express the probability function of word sequences in terms of these vectors. The probability function is a smooth function of the word representations, so these kind of models generalize much better to unknown sequences. Because the word embedding W: $words \mapsto \mathbb{R}^m$ has a useful property of clustering together synonyms and words from the same class, it allows to deal with the data sparsity by generalizing every training sentence to a class of similar sentences. The embedding and the parameters of the probability function can be learned simultaneously. In [4], a feedforward multi-layer neural network implementing this framework was shown to yield a much better perplexity than the Kneser-Ney back-off n-gram models, but at the cost of high computational complexity and nontrivial implementation.

2.2.2. The recurrent architecture

In [35], a recurrent network architecture was shown to outperform the feedforward one. The Recurrent Neural Networks (RNNs) have a simple implementation and a very useful ability to store information in the hidden layer. The architecture of neural networks used in the experiments is described in [34] and presented in Figure 2.1. The input layer is formed by concatenating $w(t)$, a vector representing the current word with $s(t-1)$, the output of the hidden layer from the previous time step:

$$x(t) = [w(t)^T s(t-1)^T]^T. \tag{2.36}$$

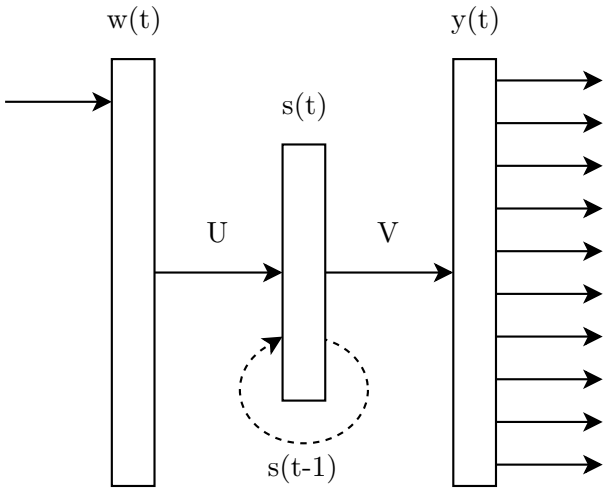

Figure 2.1: A simple RNN called an Elman network, based on [35]

.

The word vector $w(t)$ has the same size as the vocabulary. It uses 1-of-N coding, meaning that the i-th word of the vocabulary is coded by setting the i-th element to one and all the other elements to zero [51]. The output layer $y(t)$ has the same dimensionality and represents the probability distribution of the next word. The network is trained using the standard backpropagation algorithm, described in [50]. The output of the hidden layer is computed by applying the sigmoid activation function $f(z)$ on the product of the input and the weight matrix U:

$$s_j(t) = f(\sum_i x_i(t) u_{ji}), \tag{2.37}$$

where:

$$f(z) = \frac{1}{1+e^{-z}}. \tag{2.38}$$

The values of the output layer are calculated by multiplying the output of the hidden layer by the weight matrix V and applying the softmax function:

$$y_k(t) = g(\sum_j s_j(t) v_{kj}), \tag{2.39}$$

The softmax function $g(z)$ transforms a real-valued vector z into a vector σ of real values in the range (0, 1) adding up to one, so that it forms a valid probability distribution:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}}. \tag{2.40}$$

2.2.3. Extensions to the RNN model

The RNNs used in the experimental part take advantage of two important techniques – Backpropagation Through Time (BPTT) and output layer factorisation. The former aims to improve the model performance, while the latter allows to speed up the training process. BPTT is an extension of the backpropagation algorithm on the recurrent networks. The network is unfolded by duplicating the recurrent weight for an arbitrary number of time steps τ, and the error is propagated back through an unfolded network, as presented in Figure 2.2.

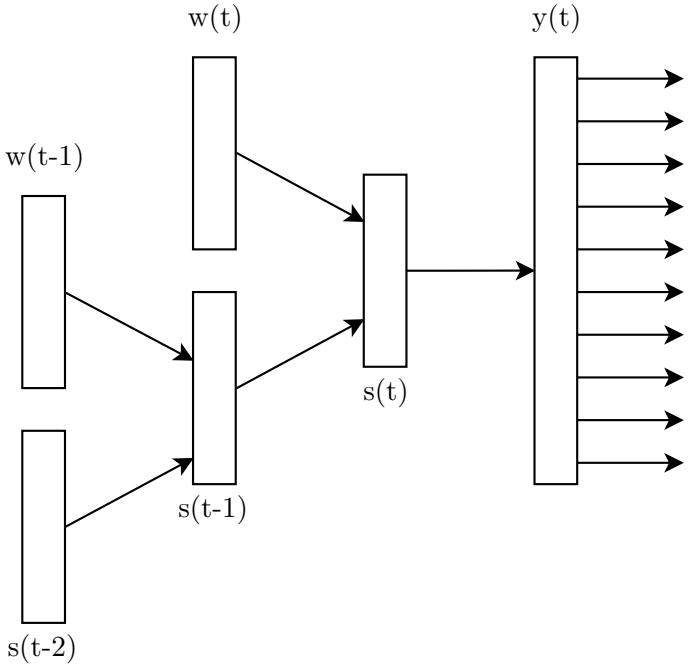

Figure 2.2: An unfolded network for BPTT with τ=2, based on [6]

.

To speed up the training, one could use a technique analogous to the one described in Section 2.1.4. Assuming that each word belongs to exactly one class, the probability of the word given its history can be calculated as in equation 2.34. This reduces the computational complexity from $(1 + H)H\tau + HV$ to $(1 + H)H\tau + HC$, where H is the size of the hidden layer, V is the size of the vocabulary, τ is the number of BPTT steps, and C is the number of classes. That can be a substantial improvement, as the HV term is usually the computational bottleneck and of course C is chosen so that $C \ll V$. However, the neural network architecture allows to extend this idea and assume that the emission probability depends on the hidden layer $s(t)$. Equation

2.34 becomes:

$$P_{\text{CLASS}}(w_i|w_{i-n+1}^{i-1}) = P_0(w_i|\pi(w_i), s(t))P_1(\pi(w_i)|s(t)). \tag{2.41}$$

Words are assigned to classes using the frequency binning with the number of classes as a parameter. The modified network architecture is presented in Figure 2.3.

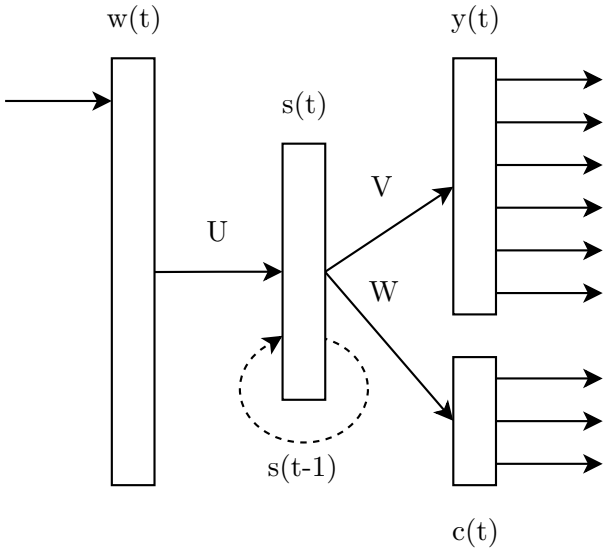

Figure 2.3: RNN with a factorised output layer, based on [35]

.

The probability distribution is first estimated over the classes and then over the words from the class containing the predicted word:

$$c_l = g(\sum_j s_j(t)w_{lj}) \tag{2.42}$$

$$y_c(t) = g(\sum_j s_j(t)v_{cj}) \tag{2.43}$$

2.3. Evaluation of language models

The best way to evaluate a language model in the context of automatic speech recognition is to simply incorporate it in an existing ASR system and measure the performance. This approach, known as extrinsic evaluation, is the only way to know whether a particular model will actually improve the recognition rates, but it can be time-consuming, as it requires running the entire system multiple times [26]. Moreover, the recognition task is a very complex one, so there are a lot of factors that can affect the performance of the model. It is therefore more practical to use an intrinsic evaluation metric, such as perplexity, which enables to quickly and objectively measure the quality of a model in an application-agnostic manner. This section describes the evaluation methods used in the experimental part.

2.3.1. Entropy and perplexity

Perplexity is the most common intrinsic metric of language model quality. It can be derived from the notion of entropy. In information theory, an information source is defined as a device which emits symbols from a finite set V. Entropy of an information source can be intuitively understood as the measure of its unpredictability or information content, expressed in bits. For an information source independently emitting symbols x with probability $P(x)$ it is defined as

$$H = -\sum_{x} P(x) \log_2 P(x). \tag{2.44}$$

Entropy is maximised when all emission probabilities are equal, that is when $P(x) = \frac{1}{|V|}$. This means that a source with entropy H contains the same amount of information as a source emitting symbols from a set of size 2^H with probability 2^{-H}.

Language can be treated as an information source producing sequences of modelling units $w_1, \ldots, w_n$ with probability $P(w_1, \ldots, w_n)$, where each token w_i is taken from a vocabulary V. Because the symbols are not emitted independently, the notion of per-word entropy is used:

$$H = -\lim_{n\to\infty} \frac{1}{n} \sum_{w_1,\ldots,w_n} P(w_1, \ldots, w_n) \log_2 P(w_1, \ldots, w_n). \tag{2.45}$$

The Shannon-McMillan-Breiman theorem states that in case of a stationary and ergodic process, it is possible to use one long sequence in the equation 2.45 instead of the summation over all possible sequences [1]. The assumption here is that a very long sequence will contain most of the shorter sequences and their frequencies will correspond to their probabilities [26]. The per-word entropy can therefore be approximated by:

$$H \approx \lim_{n\to\infty} -\frac{1}{n} \log_2 P(w_1, \ldots, w_n). \tag{2.46}$$

However, the exact probability of a word sequence is unknown, so it is more practical to use the notion of cross-entropy, which enables to replace the actual probability $P(w_1, \ldots, w_n)$ with the language model probability estimate $\hat{P}(w_1, \ldots, w_n)$. The sequences are generated according to the actual probability distribution, but the LM probability estimates are used for the log summation:

$$\hat{H} = -\lim_{n\to\infty} \frac{1}{n} \sum_{w_1,\ldots,w_n} P(w_1, \ldots, w_n) \log_2 \hat{P}(w_1, \ldots, w_n). \tag{2.47}$$

Cross-entropy can be approximated similarly to per-word entropy, using the Shannon-McMillan-Breiman theorem:

$$\hat{H} = -\frac{1}{n} \log_2 \hat{P}(w_1, \ldots, w_n). \tag{2.48}$$

The quantity in equation 2.48 can be thought of as the average amount of bits that are required to specify a word. The difference between the entropy H and cross-entropy $\hat{H}$ is a measure of the accuracy of a model. Since the true entropy is always less than or equal to the cross-entropy, a model with a lower cross-entropy is a better representation of the language. Perplexity of

the model $\hat{P}$ on a test text $W = w_1, \ldots, w_n$ is formally defined as the exponentiation of the cross-entropy:

$$PP = \sqrt[n]{\frac{1}{P(w_1, \ldots, w_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i|w_1, \ldots, w_{i-1})}} \tag{2.49}$$

Perplexity is equivalent to the weighted average branching factor of a language. A model with a perplexity of 2^H is as confused while predicting the next word as if there were on average 2^H equally probable words.

Despite its popularity as a measure of language model quality, perplexity is a rather poor method of estimating the actual performance of a language model in an ASR task. A model with lower perplexity is, in some sense, a better representation of the language, but there is no guarantee it will translate to high recognition accuracy. The main reason behind this is that perplexity does not take into account the acoustic similarities between words, which are crucial in the speech recognition process. A model able to accurately discriminate between acoustically similar words often performs better, even if its perplexity is relatively high. Some alternatives to perplexity have been proposed, such as low-level language model estimates, adversarial evaluation, artificial lattices, or classification of pseudo-negative sentences, but the word error rate is the only reliable method of measuring model performance in an ASR system [45][53].

2.3.2. WER

Word Error Rate (WER) is a common performance metric for ASR or machine translation systems. It can be defined as the length normalised word-level Levenshtein distance. More intuitively, it is the minimal quantity of insertions, deletions, and substitutions of words required to convert a hypothesis phrase into the reference phrase, divided by the length of the reference phrase:

$$WER = \frac{S + D + I}{N},$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions and N is the number of words in the reference phrase. All of these numbers are non-negative, so the minimal value of WER is zero precisely when the hypothesis and the reference are identical. However, there is no upper bound, as the number of insertions can theoretically be arbitrary. WER is often expressed as a percentage. It is common to assign different weights to insertions, deletions, and substitutions, but for all WER values in this paper they were weighted equally. Table 2.2 illustrates the WER calculation.

Word error rate is commonly used as a speech recognition accuracy metric, although it has been shown that in some applications it is not necessarily the best choice. For example, optimising the parameters of the ASR component of a speech translation system by translation metrics has been shown to lead to greater translation quality, despite a decrease in WER [21]. For the purpose of this thesis, the absolute WER reduction (WERR) is used as an evaluation

Table 2.2: An example of WER calculation for a reference phrase *let's recognize speech*

hypothesis	S	D	I	WER
lets reckon a nice beach	3	2	0	$\frac{5}{3}$
lets wreck a nice speech	2	2	0	$\frac{4}{3}$
let us wreck a nice beach	3	3	0	$\frac{6}{3}$
let's recognize	0	0	1	$\frac{1}{3}$

score for language models. WERR is defined as a reduction of WER before and after applying the model.

2.3.3. N-best list rescoring

One way of calculating the WERR of a language model is the n-best list rescoring framework. An n-best list is simply a list of n hypotheses that are most probable according to the acoustic model, along with their probabilities. The hypotheses are re-ranked by incorporating the language model information [9]. For the purpose of this work, the acoustic model probability (P_{AM}) was combined with the language model probability (P_{LM}) using the log-linear interpolation:

$$P_{\text{RE}}(h) \propto P_{\text{AM}}^{(1-\alpha)} P_{\text{LM}}^{\alpha}. \tag{2.50}$$

Table 2.3 shows an example of n-best list rescoring. The WERR of the language model can be calculated as the difference between the WER before and after rescoring. The WER can be calculated as the WER of the best hypothesis, as shown in equation **??**, or a weighted average over a set of hypothesis, with AM or LM probabilities as weights, but in the context of an ASR system, the first method is a more natural evaluation metric.

Table 2.3: N-best list rescoring with $\alpha = 0.6$.

hypothesis	P_{AM}	P_{LM}	P_{RE}	WER
lets reckon a nice beach	0.5	0.2	0.32	$\frac{5}{3}$
lets wreck a nice speech	0.4	0.1	0.22	$\frac{4}{3}$
let us wreck a nice beach	0.4	0.3	0.34	$\frac{6}{3}$
let's recognize	0.3	0.6	0.48	$\frac{1}{3}$

$$\begin{aligned} WER_1 &= WER(\arg\max P_{\text{AM}}) = \frac{5}{3} \\ WER_2 &= WER(\arg\max P_{\text{RE}}) = \frac{1}{3} \\ WERR &= WER_1 - WER_2 \approx 133.33\% \end{aligned}$$

3. Building the models

3.1. Tools and resources

This section describes the resources that served as a source of training data, and the tools that were used for gathering, filtering, and transforming the data, including XML-parsing, POS-tagging, lemmatisation, and morphological disambiguation. It also presents two language modelling frameworks that enabled building large-scale n-gram and neural language models on consumer hardware.

3.1.1. NKJP

Collecting high quality language data is a difficult task, as large and representative collections of modern texts are generally hard to obtain, if only for copyright reasons. One potential source of modern texts is obviously the Internet. However, web data can be extremely noisy and scraping, cleaning, and normalising it would be a cumbersome process. It should also be considered that the language of the Internet is different from that used in speech. Indeed, transcripts of spoken language appear to be a more valuable source of training data in ASR applications, than written texts [12]. Moreover, using raw text data to build POS-based models would require a very accurate and robust morphological tagger. For these reasons, linguistic corpora have become essential in advanced language technology. Fortunately, there exists an extensive, publicly available reference corpus of Polish language.

The National Corpus of Polish (NKJP) is a shared initiative of four institutions: the Institute of Computer Science at the Polish Academy of Sciences, the Institute of Polish Language at the Polish Academy of Sciences, the Polish Scientific Publisher PWN, and the Department of Computational and Corpus Linguistics at the University of Łódź. It has been carried out as a research-development project of the Ministry of Science and Higher Education. The full corpus contains over one and a half billion words. The list of sources for the corpora consists of literature, scientific journals, magazines, daily newspapers, and a variety of Internet texts. Most importantly, it contains transcripts of parliamentary proceedings and conversations. Moreover, the creators of the corpus claim that the conversations were chosen so that they represent both male and female speakers, in various age groups, coming from various regions in Poland [28].

Another important feature of the NKJP is that every text is accompanied by several layers of annotation. The metadata contain information about the text, such as title, author, and source, or – in case of speech transcripts – the topic of the conversation, the level of formality, background information about the speakers, and so on [47]. More importantly, every lexical unit is described by several tags carrying information about its grammatical class and category. This enabled to easily filter the data by rejecting foreign words, incomplete or corrupted segments, punctuation, and non-alphanumeric characters.

For the purpose of the experimental part of this thesis, the redistributable subcorpus was used as the main source of training data. It consists of all text of the full NKJP that are free from intellectual property constraints. The sources include mostly texts of legal documents and transcripts of parliamentary and investigation commission proceedings. For detailed information about the contents of the final training set, see Section 3.2.1.

Table 3.1: Contents of the redistributable subcorpus of the NKJP.

source	word count
books	67 000
proceedings of investigation commissions	4 623 000
legal texts	6 970 000
parliamentary proceedings	87 621 000

3.1.2. Concraft-pl

Concraft-pl is a morphosyntactic tagger for Polish. It combines Maca – a morphosyntactic segmentation and analysis tool and Concraft – a morphosyntactic disambiguation library based on constrained conditional random fields [58]. It was trained on the manually annotated subcorpus of the NKJP, so that the tagsets are compatible. The tagger was used to tag plain text data in cases where the manual or semi-automatic annotation was unavailable. Although the output of the tagger is a list of most probable morphosyntactic tags with corresponding probabilities, only the ones chosen by the disambiguation library were taken into consideration. The class mapping is therefore deterministic, which simplifies the language model and enables to use existing NKJP annotation when possible. As for lemmatisation, Concraft-pl does not disambiguate between lemmas when their morphosyntactic tag is the same, so the NKJP annotation was used in almost every case and a random lemma from the Concraft-pl output was chosen otherwise.

3.1.3. SRILM

The Stanford Research Institute Language Modelling Toolkit is a set of utilities for building and applying statistical language models, primarily for use in speech recognition, statistical

tagging, and machine translation. It has been under development in the SRI Speech Technology and Research Laboratory since 1995. It consists of a set of C++ libraries implementing language models, data structures and miscellaneous utility functions, a set of executable programs built on top of these libraries to perform standard tasks such as training and testing language models, and a collection of miscellaneous scripts facilitating minor related tasks [55]. It mainly supports n-gram modelling, so it was used in the experimental part to train and evaluate word-based and class-based n-gram models. To automate certain tasks, the toolkit was accessed via a basic Python wrapper. SRILM was used under the SRI's Research Community License and the Python binding by Nathaniel Smith was used under the MIT Licence.

3.1.4. RNNLM

The Recurrent Neural Network Language Modelling Toolkit is a simple language modelling toolkit built by Mikolov et al. [35]. It consists of a single C++ library for training language models based on recurrent neural networks. It also facilitates model evaluation by providing functionalities for perplexity calculation and n-best list rescoring. The training process is controlled by a set of parameters, such as the size of the hidden layer, the number of clustering classes, and the number of time steps to use in BPTT (see 2.2 for details). The learning rate is adjusted automatically based on an entropy score calculated on a validation file.

3.2. Method

Building language models can be roughly divided into two stages – first, the training data is gathered and pre-processed, and then the models are trained and evaluated. This section contains a description of the process conducted in the experimental part. The first subsection presents in detail the subsequent stages of the data preprocessing pipeline, while the second is devoted to the training process.

3.2.1. Building the training sets

The training data was extracted from the NKJP, more precisely from the redistributable subcorpus and the one million manually annotated subcorpus. First, the files were divided into two categories – written texts and transcripts of speech. This process was automated using the metadata. The speech training corpus contains mostly the transcripted proceedings of the Polish parliament and investigation commissions, while the text corpus consists mainly of legal documents. The testing set was built solely from parliamentary speeches and contains 20 000 sentences, each at least five words long. Testing the models on transcripts of spoken language seems reasonable if one wants to simulate the use of language models in an ASR system. However, the strong bias towards formal speech and writing is a serious issue. The training and

testing corpora come from a specific domain and therefore cannot be treated a comprehensive representation of Polish (see Figures 3.3-3.5 presenting the most common n-grams from the training set). However, the one million subcorpus, although more representative, is far too small for the experiments to yield statistically significant results. Given this trade-off, it seemed logical to sacrifice representativeness for the sake of meaningfulness.

Table 3.2: Contents of the text corpus.

source	word count
books	68 465
newspapers	813 318
legal texts	4 608 863
total	5 490 619

Table 3.3: Contents of the speech corpus.

source	word count
conversational speech	80 629
investigation commissions	4 198 129
parliamentary proceedings	13 895 902
total	18 174 660

The XML files that constitute the NKJP were parsed using a Python implementation of the ElementTree XML API. The XML files contain several layers of annotation, so it was possible to build all corpora in a single run. Four types of modelling tokens were used:

1. words – plain text,
2. lemmas – lemmatised text,
3. POS tags – tags containing only information about the grammatical class (part of speech),
4. GNC tags – tags containing information about the grammatical class and selected grammatical categories that were shown in [46] to offer significant improvements over the POS-only model (gender, number, case).

Each corpus was split into speech and text. The twelve final training datasets are listed in Table 3.4. This nomenclature will be used consistently throughout the following chapters. The contents of the corpora were filtered during extraction. All corrupted and incomplete segments were discarded, all numerals expressed with digits were substituted with a common tag, and all sentences were converted to lowercase, stripped of punctuation, non-alphanumeric symbols, foreign words and abbreviations. Only segments longer than three words were selected.

Table 3.4: List of the training corpora.

word full	lemma full	POS full	GNC full
word text	lemma text	POS text	GNC text
word speech	lemma speech	POS speech	GNC speech

3.2.2. Training the n-gram models

The n-gram models were built using the SRILM toolkit. First, the n-gram counts were computed for each corpus. Only unigrams, bigrams, and trigrams were taken under consideration.

Table 3.5: Number of unique tokens in the word corpus.

	text	speech	full
unigrams	157 866	184 236	244 346
bigrams	1 540 010	4 083 052	5 111 189
trigrams	4 669 098	14 549 205	12 843 843

Table 3.6: Number of unique tokens in the lemma corpus.

	text	speech	full
unigrams	54 522	41 666	65 420
bigrams	1 012 669	2 241 798	2 835 191
trigrams	2 555 053	8 314 572	10 380 493

Table 3.7: Number of unique tokens in the POS corpus.

	text	speech	full
unigrams	34	34	34
bigrams	851	905	935
trigrams	10238	14030	14689

Table 3.8: Number of unique tokens in the GNC corpus.

	text	speech	full
unigrams	462	411	475
bigrams	34 874	38 107	45 689
trigrams	389 709	686 461	803 012

The results for the four types of corpora are presented in Tables 3.5-3.8. The counts include the start-of-sentence and end-of-sentence tokens. The statistics confirm the issues described in Subsection 1.2.3. The first evident problem is the size of the lexicon. Even though the corpus is limited to one domain, there are 244 346 distinct unigrams and 148 567 of them occur more than once. To achieve 99% unigram coverage, a lexicon of more than 92 000 words is required (see Figure 3.1). Note also the data sparsity – even for the POS corpus, with only 34 unigrams, less than 40% of potential trigrams occur in the corpus. In case of the word corpus, less than 0.0009% of potential bigrams are present in the training data. Another interesting observation is the relationship between the word and lemma corpus. There are almost four times more inflected unigrams than distinct lemmas – a word in Polish has, on average, four inflected forms. A vocabulary of 33 000 lemmas is enough to achieve 99% coverage. However, important information about grammatical agreement is lost in the lemmatisation process.

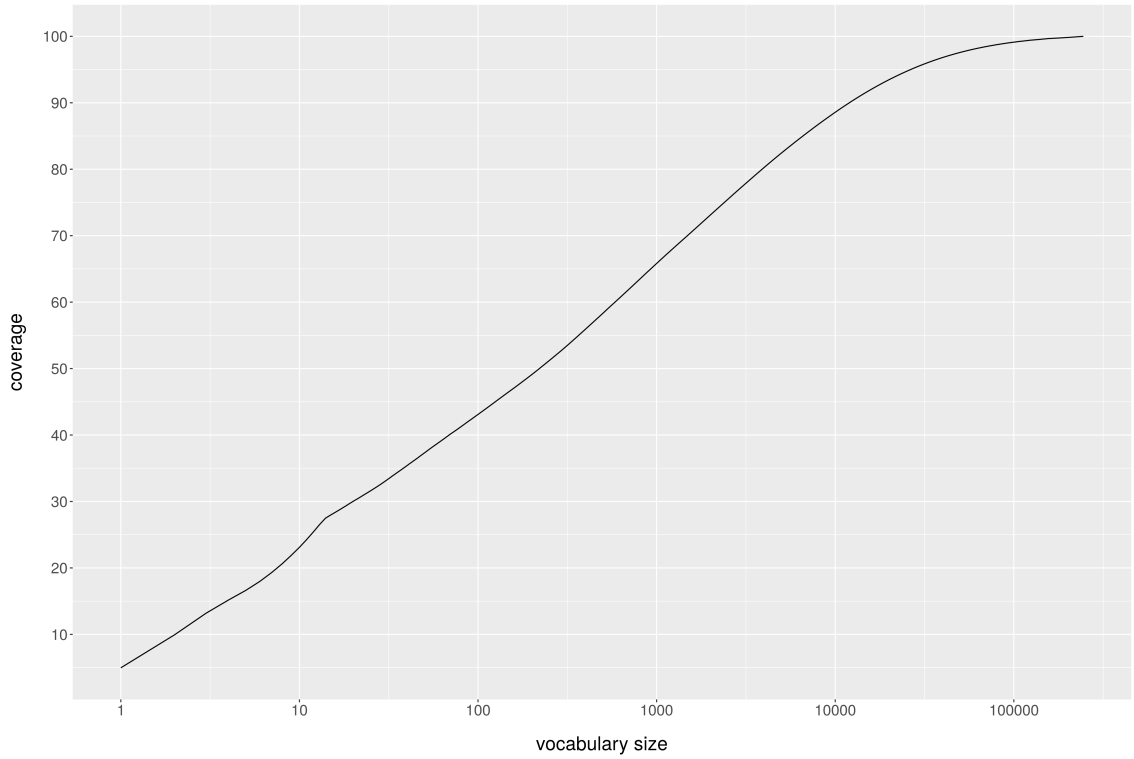

Figure 3.1: Unigram coverage of the full word corpus as a function of the vocabulary size.

The n-gram counts were then smoothed. Several smoothing methods were tested – Good-Turing, Kneser-Ney, and Chen-Goodman's modified Kneser-Ney, both in the back-off and interpolated variant. Perplexity of the word text model calculated on the testing set is presented in Table 3.9. The model discounted using the interpolated modified Kneser-Ney method has the lowest perplexity. This smoothing technique was used for all language models except for the POS and GNC models, where the size of the vocabulary is too small to apply it. However, in case of models with a small vocabulary, smoothing is less of an issue.

Table 3.9: Perplexity of a language model trained on the full word corpus with different smoothing methods – Good-Turing, Kneser-Ney (back-off), Kneser-Ney (interpolated), Chen-Goodman (back-off), Chen-Goodman (interpolated).

	GT	KNB	KNI	CGB	CGI
unigrams	2604.09	3735.08	3735.08	3767.08	3767.37
bigrams	387.11	471.07	452.72	484.31	458.40
trigrams	277.19	250.40	244.44	254.08	242.31

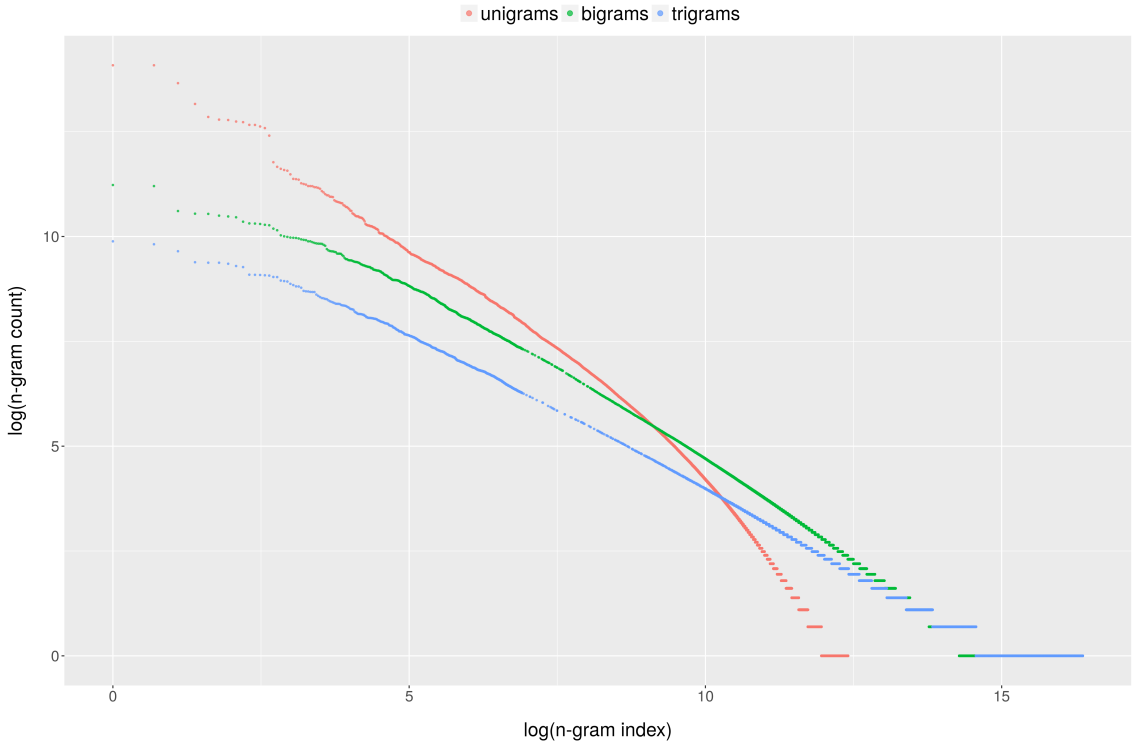

Figure 3.2: N-gram distribution in the full word corpus.

Figure 3.2 presents the distribution of n-grams in the full word corpus on a double logarithmic plot. Zipf's law states that the rank-frequency distribution of words in a natural language corpus is an inverse relation. More formally, the probability mass function of the term frequencies is of the form:

$$p = \alpha k^{-\beta}, \tag{3.1}$$

where k is the rank of the term from the most to the least frequent one, and (α, β) are distribution parameters. Except for the very common and very rare terms, n-grams in the full word corpus follow a Zipfian distribution. The exponential decay factor β decreases with n-gram order and so does α, the frequency of the most common term.

Figures 3.3-3.5 present the most common terms of the full word corpus. Although the popular

unigrams and bigrams are mostly stop words and general phrases, trigrams reveal a strong bias towards parliamentary jargon. For example, the phrase *radiofonii i telewizji* refers to a bill which was the subject of the so called Rywin affair, a large corruption scandal investigated by a parliamentary committee, proceedings of which consist a large portion of the training corpus. This lack of representativeness will probably be reflected during the model evaluation.

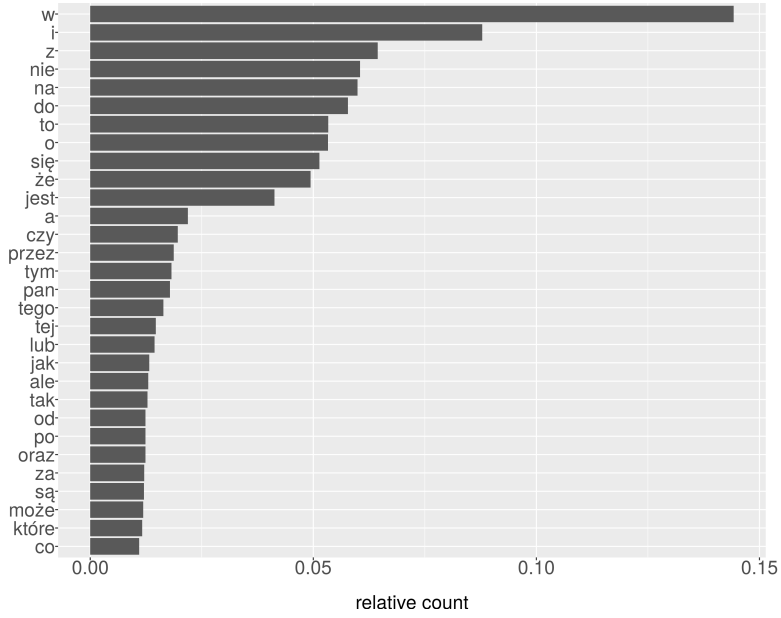

Figure 3.3: Most common unigrams of the full word corpus.

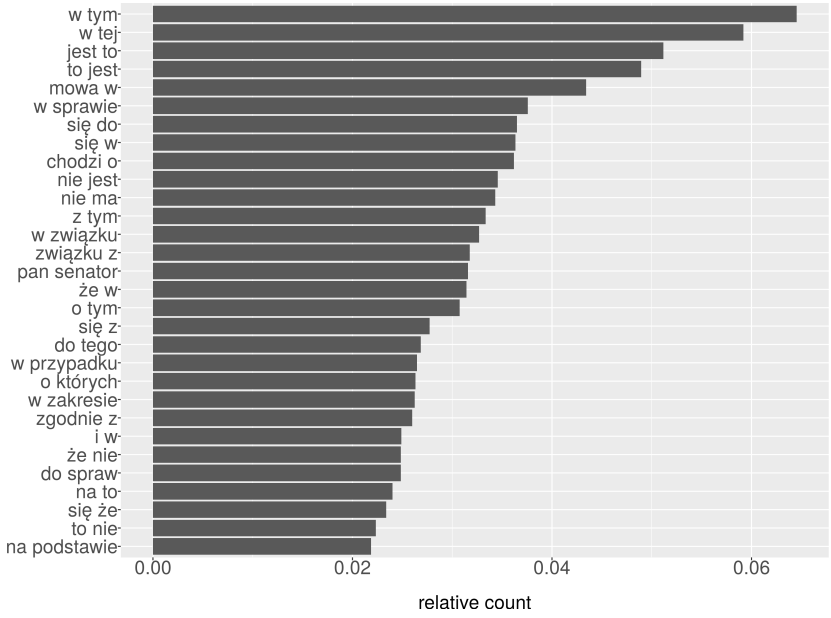

Figure 3.4: Most common bigrams of the full word corpus.

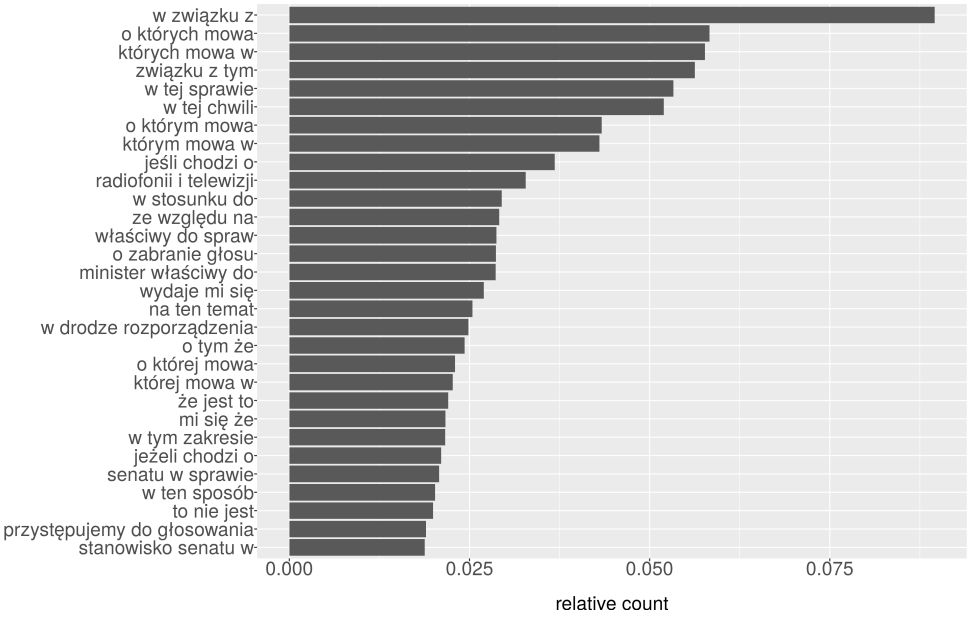

Figure 3.5: Most common trigrams of the full word corpus.

3.2.3. Training the neural network models

The neural network models were trained on the same datasets as the n-gram models. However, as the computation times for RNN models were extremely large, only the full corpora were used, resulting in four RNN models. Twenty percent of phrases from each training corpus were randomly assigned to a validation set, which was used for controlling the learning rate and early stopping. All models were trained using 300 neurons in the hidden layer and two-step BPTT. For word and lemma corpora, 500 word classes were used. For POS and GNC models, no additional clustering was necessary.

4. Results

4.1. Perplexity

The perplexity values presented in Tables 4.1-4.4 were calculated on a test set of 20000 utterances taken from the transcripted proceedings of the Polish parliament. The lowest perplexity achieved for a word n-gram model is 240.54. It should once again be noted that both the training and the test data come from a specific domain. This could significantly reduce the perplexity, as legal and parliamentary language is generally more predictable. For example, in [4] it is reported that a Kneser-Ney smoothed back-off trigram model achieved a perplexity of 323 on the Brown Corpus (general English) and only 127 on the AP News Corpus.

Table 4.1: Perplexity of the word language models with Chen-Goodman's modified Kneser-Ney smoothing.

	text	speech	full
unigrams	4409.47	3510.78	3767.37
bigrams	1380.31	445.76	458.40
trigrams	1231.23	240.54	242.31

Table 4.2: Perplexity of the lemma language models with Chen-Goodman's modified Kneser-Ney smoothing.

	text	speech	full
unigrams	1844.97	1309.19	1654.56
bigrams	744.72	299.82	367.30
trigrams	694.41	177.91	213.60

Because the models have different vocabularies, the perplexity scores are not comparable across different types of models. However, for every type of models, the perplexity of speech models is lower than that of text models. This could be attributed to a larger training set, but in case of the word, lemma, and POS corpus, the perplexity of a model trained on the full corpus is consistently higher than that of a model trained on the speech corpus alone. This

means that extending the training set by incorporating the text data actually lead to a worse model, assuming that perplexity is a valid metric in this context. Another, rather unsurprising observation is the large and consistent decrease in perplexity with the n-gram order. Tables 4.1 and 4.2 show that the drop in perplexity is more pronounced in case of models with a larger vocabulary. Interestingly, the difference in perplexity between trigram and unigram models is much greater in case of models trained on speech data. For example, the perplexity of the word text trigram model is approximately 3.6 times lower than that of a word text unigram model, while the perplexity of a word speech trigram model is 14.6 times lower than that of its unigram counterpart.

Table 4.3: Perplexity of the POS language models with Witten-Bell smoothing.

	text	speech	full
unigrams	11.25	10.40	10.37
bigrams	9.36	8.35	8.37
trigrams	8.71	7.71	7.73

Table 4.4: Perplexity of the GNC language models with Good-Turing smoothing.

	text	speech	full
unigrams	90.04	82.16	81.38
bigrams	38.73	35.61	34.52
trigrams	33.35	29.44	28.26

Tables 4.3 and 4.4 present the perplexity results for the models based on morphosyntactic tags. The effect of the vocabulary size on perplexity is clearly visible. In case of the POS model, the perplexity of the model trained on the speech corpus is again lower than that of a model trained on the full corpus. However, the opposite is true for the GNC model.

Generally, the perplexity scores hint that the models trained on speech transcripts are a better representation of the language for the purpose of ASR, as they tend to have a lower perplexity than models trained on written text, despite having a slightly larger vocabulary. To further verify this hypothesis by eliminating the effect of the training set size, two equal sized training sets were built, one consisting of speech transcripts and the other of written texts. Then, four pairs of n-gram models were trained on these sets, this time ensuring that the vocabularies of corresponding models are identical. The perplexity of these models is presented in Tables 4.5-4.8. The models based on speech data have a significantly lower perplexity for all four types of modeling units. The difference is more prominent in case of word and lemma models. For example, the perplexity of a speech-based word trigram model is more than four times lower than that of a text-based model. In case of POS and GNC models, the difference is still noticeable,

although much less pronounced. Moreover, in case of speech models, perplexity still decreases more rapidly with the n-gram order. For example, using trigrams instead of unigrams results in a tenfold drop in perplexity of a speech-based model, and only a threefold decrease in case of a text-based model.

Table 4.5: Perplexity of word text and speech models with equal vocabularies.

	text	speech
unigrams	3351.69	2287.98
bigrams	1051.36	367.79
trigrams	942.74	226.06

Table 4.6: Perplexity of lemma text and speech models with equal vocabularies.

	text	speech
unigrams	1520.89	1149.99
bigrams	560.73	260.05
trigrams	498.97	154.91

Table 4.7: Perplexity of POS text and speech models with equal vocabularies.

	text	speech
unigrams	11.23	10.29
bigrams	9.27	8.23
trigrams	8.62	7.56

Table 4.8: Perplexity of GNC text and speech models with equal vocabularies.

	text	speech
unigrams	159.14	156.54
bigrams	47.79	45.08
trigrams	27.88	20.86

Table 4.9 presents the perplexity of neural models evaluated on the same testing set. When compared to corresponding n-gram models with the same vocabulary, the perplexity of neural models is slightly lower, with an exception of the lemma model.

Table 4.9: Perplexity of neural models.

word	240.81
lemma	216.64
pos	7.12
gnc	24.87

4.2. Size

The size of a text representation of the model can serve as a proxy for the amount of memory it will require in the final application. As shown in Table 4.10, the size of the model seems to be dependent mostly on the size of the vocabulary, with an interesting exception of the neural GNC model, which not only takes less space than the neural POS model, but is also 25 times smaller than its n-gram counterpart.

Table 4.10: Model size in MB.

neural word	899
n-gram word	615
n-gram lemma	432
neural lemma	336
n-gram gnc	36
neural pos	3.7
neural gnc	1.4
n-gram pos	0.4

4.3. N-best list rescoring

The values of WERR presented in this section were calculated using the n-best list rescoring framework described in Subsection 2.3.2. The n-best lists were actual recognition hypotheses from the Sarmata ASR system [64]. The test utterances come from Polish television news programmes. The language models were tested on a set of 84 lists, each containing about 240 hypotheses. Figures 4.1-4.6 present the absolute WERR of language models as a function of the α coefficient. The results should be treated with a certain dose of caution, as the quality of the acoustic hypotheses leaves a lot to be desired – the minimal WER a language model can achieve on this corpus is 66.35%, and the maximal possible WERR is only 18.9%. Note that all WER and WERR values in this and the following section are given as a percentage.

Table 4.11 presents the maximum WERR values achieved by the respective models. Unsurprisingly, the models trained on words achieve the highest WERR, with the n-gram model

having a slight advantage over its neural counterpart. As far as the trigram models are concerned, those trained on speech data perform slightly better than those trained on written text and In case of the lemma and GNC model, they even achieve higher WERR than the models trained on the full corpus.

Table 4.11: Maximal WERR achieved by respective models in n-best list rescoring.

word full	15.67
lemma speech	15.32
neural word	15.28
word speech	15.22
word text	15.16
lemma full	14.88
lemma text	14.24
neural lemma	14.10
gnc speech	13.75
gnc full	13.54
gnc text	13.29
neural gnc	13.20
pos text	9.89
neural pos	9.58
pos full	9.54
pos speech	9.47

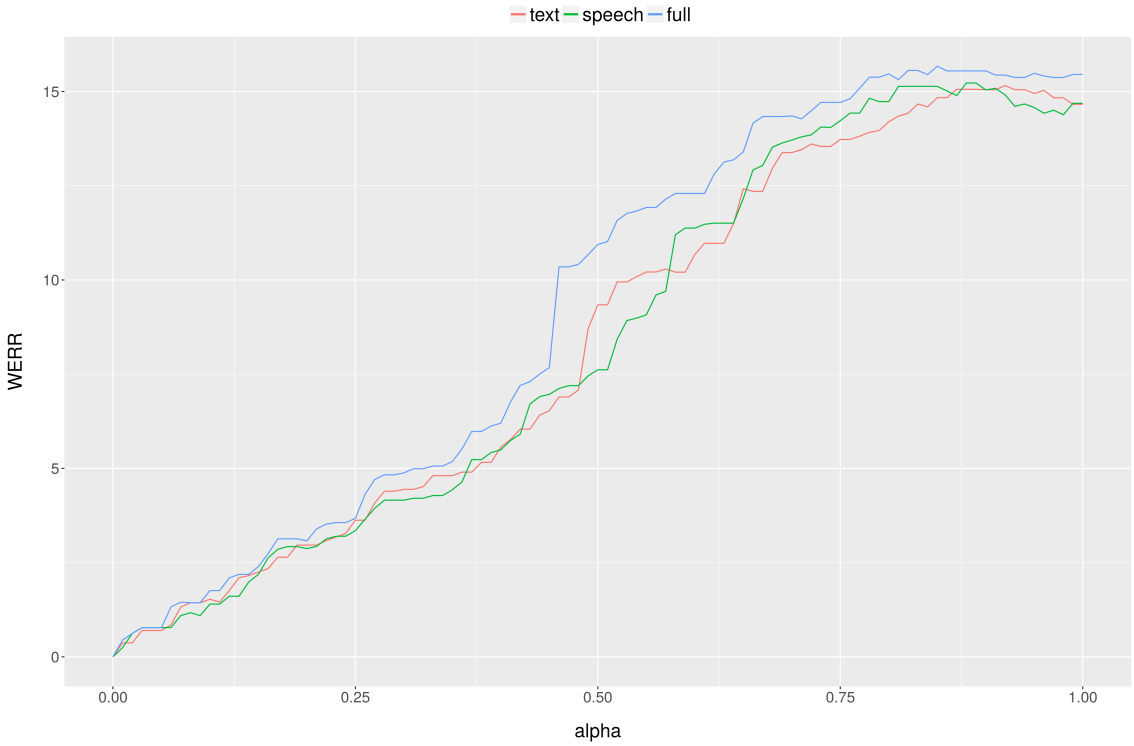

Figure 4.1: Absolute WERR of the word trigram model.

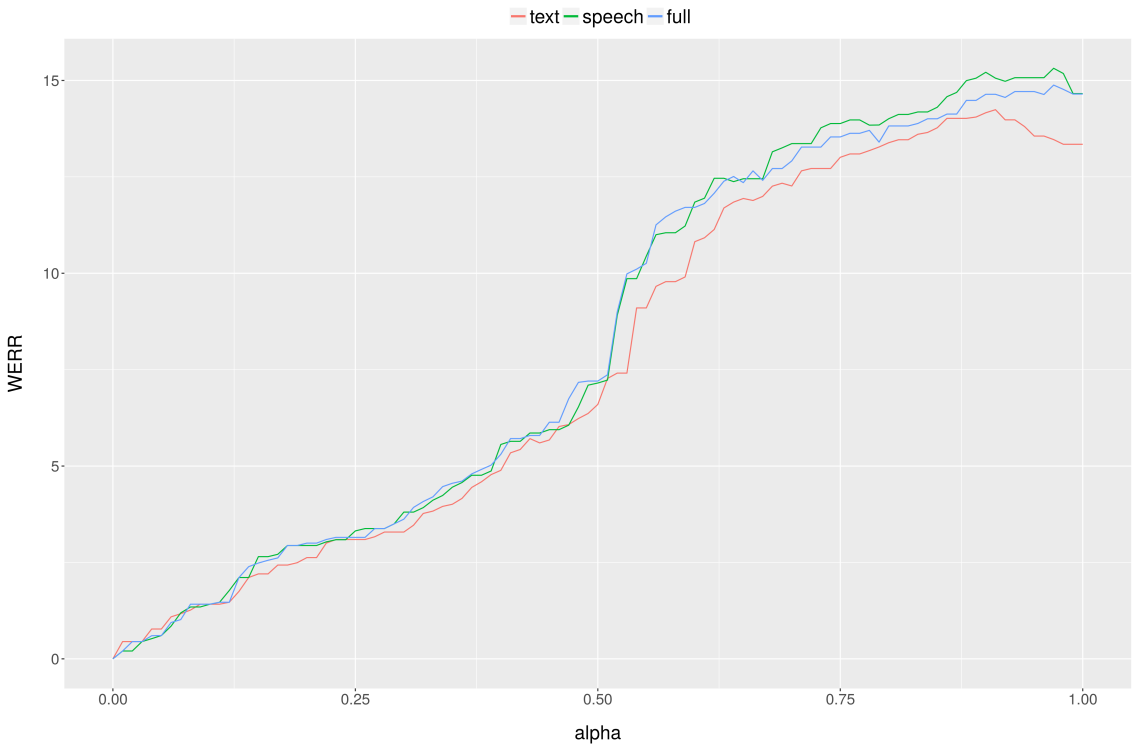

Figure 4.2: Absolute WERR of the lemma trigram model.

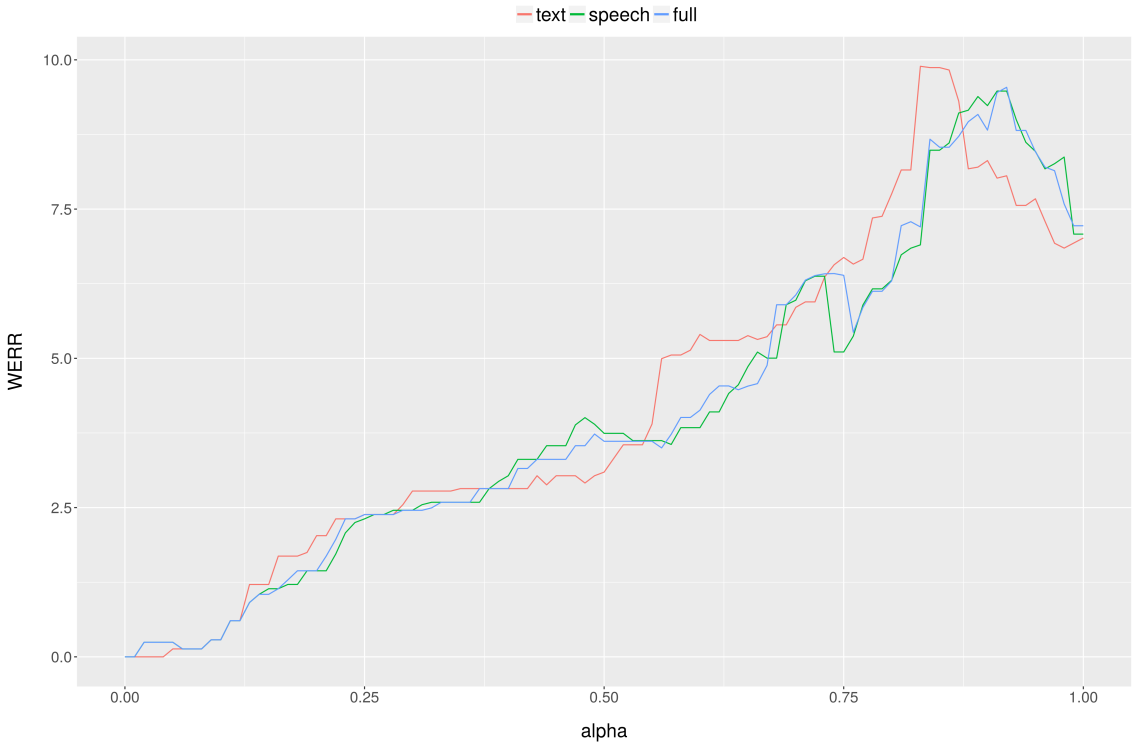

Figure 4.3: Absolute WERR of the POS trigram model.

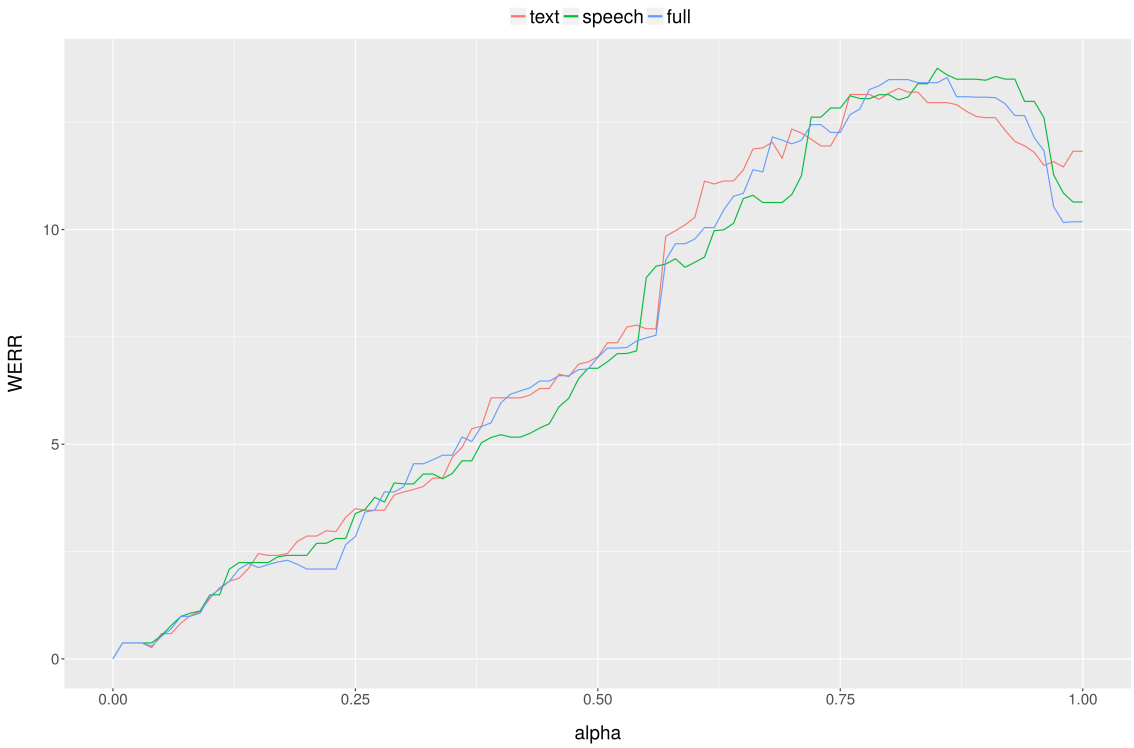

Figure 4.4: Absolute WERR of the GNC trigram model.

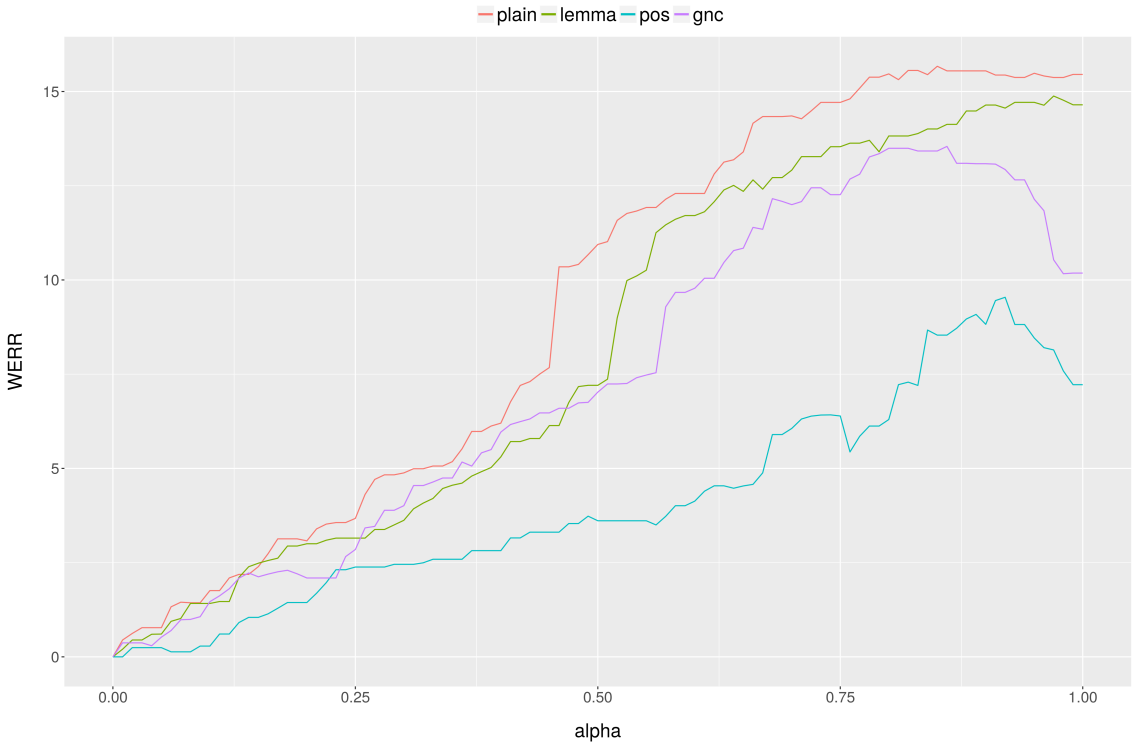

Figure 4.5: Absolute WERR of the trigram models.

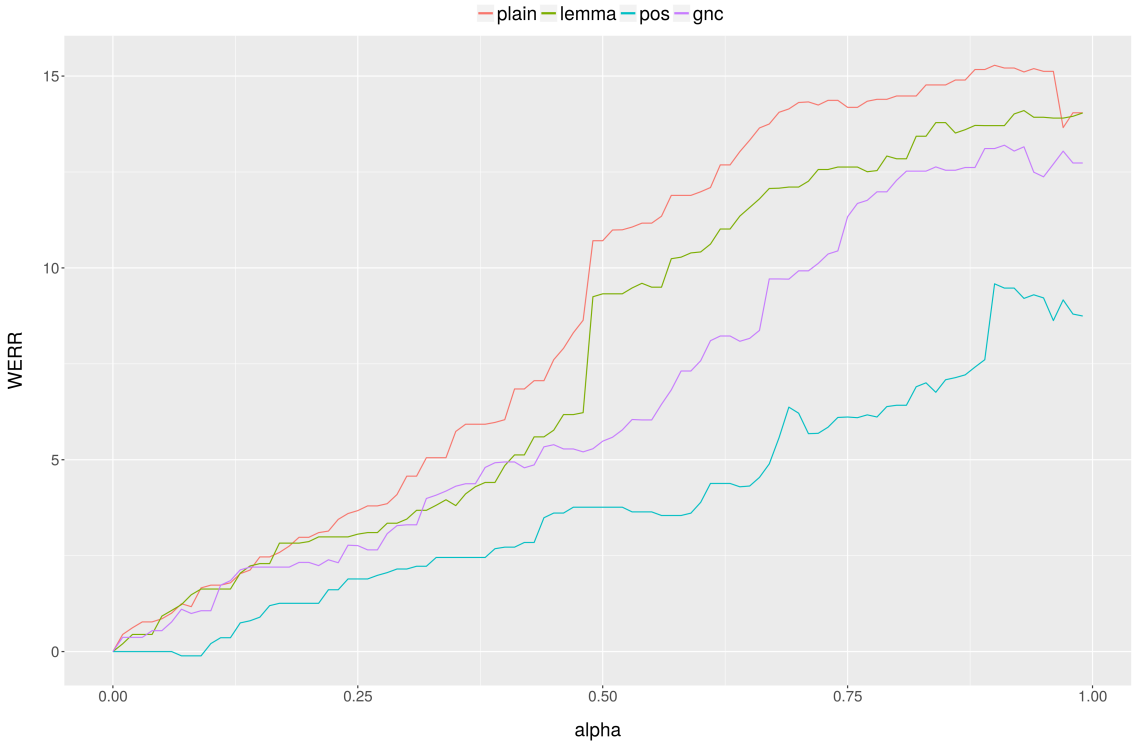

Figure 4.6: Absolute WERR of the neural models.

4.3.1. Mock n-best list rescoring

The WERR results presented in Subsection 4.3 were calculated on a real ASR system. However, the n-best lists were generated by an old version of Sarmata, when the performance of the acoustic model was rather poor, and they are therefore of limited value as a benchmark. As the recognition quality improves, it gets more important for a model to be able to correctly distinguish between very similar phrases. The generation of artificial n-best list is an implementation of the adversarial evaluation technique described in [53]. A simple algorithm was used to create a set of ten hypotheses for each of the 5000 phrases randomly selected from the testing set used for perplexity evaluation. The hypotheses were generated by substituting random words with phonetically similar ones. Only words occurring more than 5000 times in the training data were used for substitution. Each word in the reference phrase had the same probability of being substituted, but that probability was different across the hypotheses. Insertion and deletion errors were neglected. Apart from simplicity, this decision was made to avoid the n-gram models' bias for shorter phrases and to simulate the operation of an advanced acoustic model. Because the orthography of Polish is highly phonemic, it was assumed that graphemic similarity is a good proxy of phonemic similarity. Therefore, the grapheme-level Levenshtein distance served as a heuristic for finding similar words. Inspecting the mock acoustic hypotheses showed that there is indeed a large degree of acoustic similarity between them. Table 4.12 presents an example phrase along with the automatically generated hypotheses.

Table 4.12: An example phrase and generated mock hypotheses.

coraz więcej polaków wyjeżdża odwiedza kraje europejskie
coraz więc polaków wyjeżdża odwiedza kraju europejskie
wraz więcej polaków wyjeżdża odwiedza kraju europejskiej
oraz więcej polaków wyjeżdża odwiedza kraje europejskiej
raz więc polaków wyjeżdża zwiedza kraju europejskie

Table 4.13 presents the mean WER of the hypotheses chosen by the trigram models. The lowest WER achieved is 8.55 for the full word model. Generally, the word-based model has the lowest error rate. Interestingly, the full GNC model not only significantly outperforms the POS model, but also the lemma model, despite a much smaller vocabulary. Moreover, in case of the text-based corpus, the GNC model has a very similar error rate as the word-based model. This result proves that the morphosyntactic tags can be a viable alternative to words in case of highly inflected language. The models based on speech transcript outperform their text-based equivalents in all cases, but the difference is less pronounced in case of models with a smaller vocabulary. This may suggest that the dissimilarities between spoken and written language lie more in semantics than in syntax.

Table 4.13: WER of trigram models.

	text	speech	full
word	12.15	8.67	8.55
lemma	16.95	11.81	13.34
pos	20.96	19.38	19.54
gnc	12.58	12.27	11.89

Table 4.14 presents the mean WER of the neural models. Regardless of the type of the modelling unit, they outperform the n-gram models trained on the same corpus. In particular, the neural GNC model achieves a lower WER than all n-gram models except for the speech and full word-based trigram model. This result suggests that RNN morphosyntactic models can be a promising area of research as far as speech recognition is concerned, as they combine the compactness of class-based models and the ability to capture relationship spanning across multiple words, which is especially important in case of languages with a large degree of inflection.

Table 4.14: WER of neural models.

word	8.21
lemma	12.55
pos	17.65
gnc	11.61

Table 4.15 presents the results of the mock n-best list rescoring. The neural network models outperform their n-gram equivalents for every modelling unit. Interestingly, the neural GNC model achieves a better WERR than the word text trigram and all the lemma models, despite a much smaller vocabulary.

Table 4.15: WERR achieved by respective models in mock n-best list rescoring.

neural word	21.43
word full	21.09
word speech	20.97
neural gnc	18.03
lemma speech	17.83
gnc full	17.75
word text	17.49
gnc speech	17.37
neural lemma	17.09
gnc text	17.06
lemma full	16.30
lemma text	12.69
neural pos	11.99
pos speech	10.26
pos full	10.10
pos text	8.68

5. Conclusions

The purpose of this thesis was to conduct a comparative analysis of morphosyntactic and semantic language models in the context of automatic speech recognition of Polish. Sixteen n-gram, class n-gram and neural network models were built and evaluated using several performance metrics: perplexity, size, and word error rate reduction on two n-best list rescoring tasks – one using the output of a real ASR system, and the other involving artificially generated recognition hypothesis.

Perplexity evaluation, although often criticized as a metric of the language model quality, revealed some trends which were further confirmed in the WERR experiments. First of all, there is a noticeable difference between language models trained on speech transcripts and those trained on written texts, regardless of the modelling unit used. The comparison of models with identical vocabularies showed that models trained on speech transcripts achieve lower perplexity than their text-based counterparts. Moreover, in case of words and lemmas, the speech-based trigram models outperform those trained on the full corpora. Although more cumbersome, the WERR evaluation on a real ASR system is undoubtedly a more reliable evaluation method than perplexity. The results confirm what has been reported in [12], this time on a much larger training set – n-gram models trained on speech transcripts indeed perform better than text-based models, even when morphosyntactic tags are used as the modelling units. Figure 5.1 presents a summary of results.

The quality of the acoustic model may have been an important factor in the WERR calculation pipeline. Because a newer version of the recognition engine was unavailable, an alternative approach was proposed to simulate the operation of an advanced acoustic model. The mock n-best list rescoring is a compromise between proxy metrics like perplexity and using a real recognition system, as the generation of mock n-best lists is fully automated. It allows to simulate the rescoring task using any testing set without an actual ASR system. The results of the mock n-best list rescoring evaluation are presented in Figure 5.2. Note the very good score of the neural and n-gram GNC models, which both outperformed the word text model, despite a vocabulary order of magnitudes smaller.

To sum up, the experimental part provided some valuable insights into how to optimise the language modelling of highly inflected language for speech recognition. First of all, the speech data once again proved to be a more valuable source of information than written texts [12].

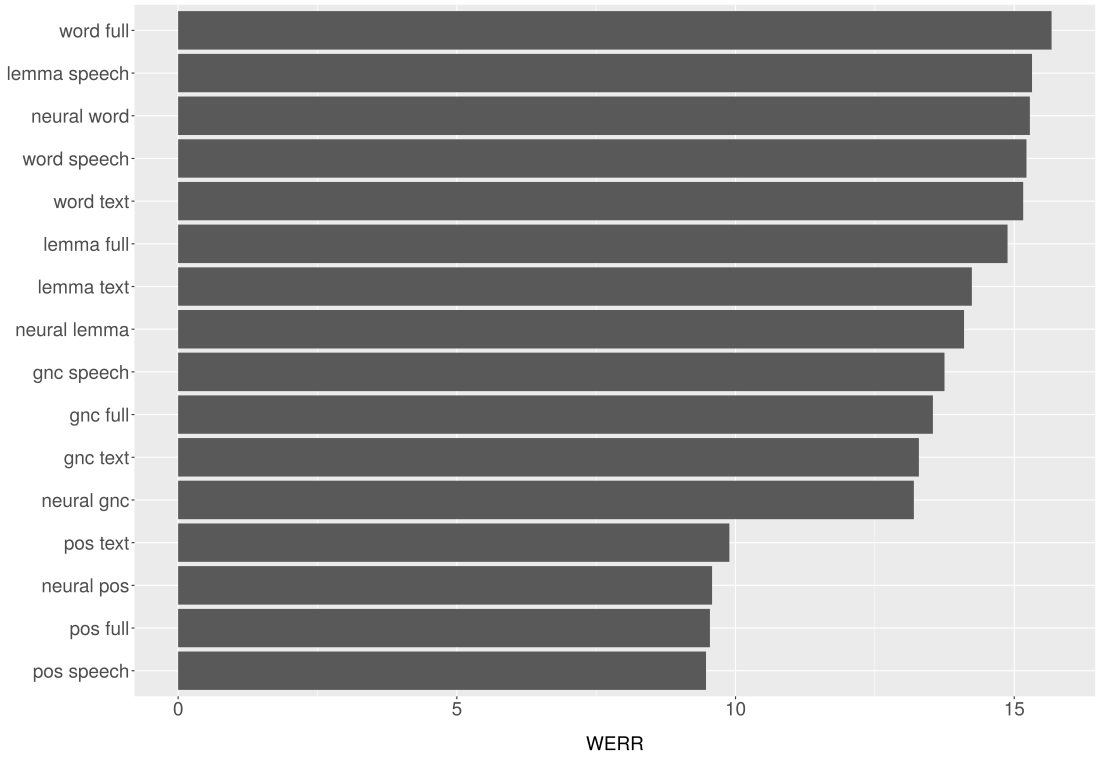

Figure 5.1: The absolute WERR achieved by respective models in n-best list rescoring.

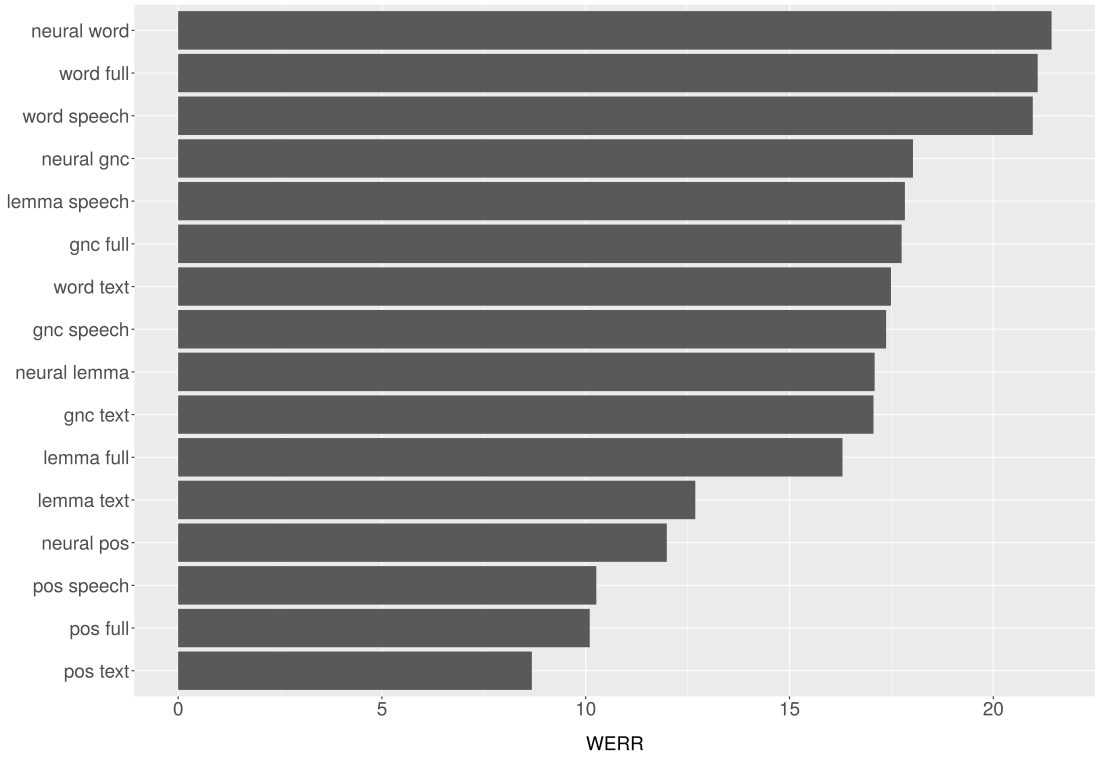

Figure 5.2: The absolute WERR achieved by respective models in mock n-best list rescoring.

In some cases, incorporating the text data into a speech-based model actually led to a degradation in performance. Unfortunately, the conversation subcorpus of the NKJP is too small to repeat the experiments on general spoken language.

The second important observation is that a morphosyntactic model based on grammatical classes and three grammatical categories (gender, number, case) turned out to be a very effective way of modelling Polish for speech recognition. The GNC model performs similarly to a lemma-based model and does so at a fraction of the computational cost. It is likely that it is possible to find an equally informative set of grammatical categories for other inflected languages.

The last conclusion that can be drawn from the experimental part is that although the neural network models in most cases outperform their n-gram counterparts, the differences are rather slight. This can be attributed to numerous factors. Firstly, the neural models were trained on full corpora instead of just the speech transcripts. Moreover, they were not optimised, as the values of parameters (the number of BPTT steps, the size of the hidden layer, the number of classes) were chosen so that the training part would be feasible on a consumer hardware. Additional experiments could definitely shed some light on the matter. It would be especially interesting to build and test a neural GNC model trained entirely on speech data, this time with a much larger hidden layer and a lot more BPTT steps, as it has the potential of being a viable alternative to word-based n-grams.

Appendices

A. Polish phonology

Table A.1: Polish vowels.

Polish	IPA	example
a	a	*atak* /atak/ (attack)
e	ɛ	*era* /ɛra/ (era)
i	i	*igła* /igwa/ (needle)
o	ɔ	*oko* /ɔkɔ/ (eye)
u/ó	u	*ucho* /uxɔ/ (ear), *ósmy* /usmɨ/ (eighth)
y	ɨ	*rym* /rɨm/ (rhyme)
ę	ɛ $^{\tilde{w}}$	gęsty /gɛ$^{\tilde{w}}$stɨ/ (thick)
	ɛ $^{\tilde{j}}$	gęś /gɛ$^{\tilde{j}}$ɕ/ (goose)
	ɛn	wędka /vɛntka/ (rod)
	ɛɲ	sędzia /sɛɲdʑa/ (judge)
	ɛm	tępy /tɛmpɨ/ (blunt)
ą	ɔ $^{\tilde{w}}$	wąs /wɔ$^{\tilde{w}}$s/ (moustache)
	ɔŋ	bąk /bɔŋk/ (bumblebee)
	ɔɲ	wziąć /vʑɔɲtɕ/ (to take)
	ɔn	wątroba /vɔntrɔba/ (liver)
	ɔm	kąpiel /kɔmp$^{\text{j}}$jel/ (bath)

Table A.2: Polish consonants

Polish	IPA	example	English approximation
b	b	*bar* /bar/ (bar)	**b**ar
c	t͡s	*co* /t͡sɔ/ (what)	ca**ts**
ć/c(i)	t͡ɕ	*ćma* /t͡ɕma/ (moth), *ciało* /t͡ɕawɔ/ (body)	**ch**eer*
cz	t͡ʂ	*czar* /t͡ʂar/ (spell)	**ch**ild*
d	d	*dom* /dɔm/ (home)	**d**og
dz	d͡z	*dzwon* /d͡zvɔn/ (bell)	be**ds**
dź/dz(i)	d͡ʑ	*dźwig* /d͡ʑvik/ (lift), *dzień* /d͡ʑeɲ/ (day)	**j**eep*
dż	d͡ʐ	*dżem* /d͡ʐɛm/ (jam)	**j**ug*
f	f	*fan* /fan/ (fan)	**f**an
g	g	*gol* /gɔl/ (goal)	**g**irl
g(i)	gʲ	*magia* /magʲa/ (magic)	ar**g**ue
h/ch	x	*hak* /xak/ (hook), *cham* /xam/ (boor)	**h**ope
h(i)/ch(i)	xʲ	*hit* /xʲit/ (hit), *chichot* /xʲixɔt/ (giggle)	**h**ue
j	j	*jak* /jak/ (how)	**y**es
k	k	*kat* /kat/ (executioner)	s**c**am
k(i)	kʲ	*kiedy* /kʲjɛdɨ/ (when)	s**k**ew
l	l	*lot* /lɔt/ (flight)	**l**ion
ł	w	*łoś* /wɔɕ/ (moose)	**w**ay
m	m	*mapa* /mapa/ (map)	**m**ile
m(i)	mʲ	*miód* /mʲjut/ (honey)	**m**ute
n	n	*nos* /nɔs/ (nose)	**n**o
ń/n(i)	ɲ	*koń* /kɔɲ/ (horse), *nie* /ɲɛ/ (no)	ca**ny**on
p	p	*park* /park/ (park)	**p**ut
r	r	*rap* /rap/ (rap)	–
s	s	*suma* /suma/ (sum)	**s**ong
ś/s(i)	ɕ	*śruba* /ɕruba/ (screw), *siano* /ɕano/ (hay)	**sh**e*
sz	ʂ	*szum* /ʂum/ (noise)	**sh**ore*
t	t	*tak* /tak/ (yes)	**t**op
w	v	*wór* /vur/ (bag)	**v**an
z	z	*zero* /zɛrɔ/ (zero)	**z**ebra
ź/z(i)	ʑ	*źrebak* /ʑrɛbak/ (foal), *zima* /ʑima/ (winter)	vi**si**on*
ż/rz	ʐ	*żal* /ʐal/ (sorrow), *rząd* /ʐɔnt/ (government)	a**z**ure*

* Both the laminal retroflex sounds /ʂ ʐ t͡ʂ t͡ʐ/ and corresponding alveolo-palatals /ɕ ʑ t͡ɕ t͡ʑ/ sound similar to the English palato-alveolar consonants /ʃ ʒ t͡ʃ t͡ʒ/, but are distinct sounds in Polish.

Several comments should be made to Table A.2:

- the sound /ʐ/ can be written either as <ż>, as in *może* /moʐe/ (he/she can) or <rz>, as in *morze* /moʐe/ (sea). however, <rz> can also denote a combination of /r/ and /z/, e.g. *zamarzać* /zamarzat͡c/ (to freeze),
- the sound /x/ is rendered orthographically either as <ch> or <h>. it has strongest friction before consonants, weaker friction before vowels and weakest friction intervocalically, where it may be realized as glottal /h/. it also has a voiced allophone /ɣ/ that occurs whenever /x/ is followed by a voiced obstruent, as in *niechby* (may it),
- the phoneme /n/ has a velar allophone ŋ, which occurs before velar consonants, as in *bank* /baŋk/ (bank),
- the affricatives /t͡ʂ d͡ʐ/ are written with digraphs <cz dż>. in contrast, transitions /ts dz/ are usually denoted by trigraphs <trz> and <drz>, e.g. *trzoda* /tʂoda/ (flock), *drzewo* /dʐevo/ (tree), with some rare exceptions, like *nadżerka* /nadʐerka/ (laceration),
- the alveolo-palatals /c ɟ t͡ɕ d͡ʑ ɲ/ are rendered <ś ź ć dź ń> pre-consonantally and word-finally. however, they are spelled with multigraphs <si zi ci dzi ni> before a vowel. there are some exceptions, especially in loanwords, e.g. *sinus* /sinus/ (sine), *siwert* /sivert/ (sievert),
- the retroflex fricatives and affricates /ʂ ʐ t͡ʂ t͡ʐ/ are sometimes transcribed as palato-alveolar consonants /s z t͡s t͡z/ [23],
- the phonemes /kʲ/ and /gʲ/ are less commonly transcribed as /c/ and /ɟ/.

Bibliography

[1] Algoet, P. H., and Cover, T. M. A sandwich proof of the Shannon-McMillan-Breiman theorem. *The annals of probability* (1988), 899–909.

[2] Baker, J. The DRAGON system – an overview. *Acoustics, speech and signal processing, IEEE transactions on 23*, 1 (1975), 24–29.

[3] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov Chains. *The annals of mathematical statistics* (1970), 164–171.

[4] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. A neural probabilistic language model. *Journal of machine learning research 3*, Feb (2003), 1137–1155.

[5] Besacier, L., Barnard, E., Karpov, A., and Schultz, T. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication 56* (2014), 85–100.

[6] Boden, M. A guide to recurrent neural networks and backpropagation. *The Dallas project, SICS technical report* (2002).

[7] Bourlard, H., and Morgan, N. *Connectionist speech recognition: a hybrid approach*, vol. 247. Springer Science & Business Media, 1994.

[8] Brocki, L., Marasek, K., and Korzinek, D. Connectionist language model for Polish, 2012.

[9] Broman, S., and Kurimo, M. Methods for combining language models in speech recognition. In *INTERSPEECH* (2005), pp. 1317–1320.

[10] Brown, P., Desouza, P., Mercer, R., Pietra, V., and Lai, J. Class-based n-gram models of natural language. *Computational linguistics 18*, 4 (1992), 467–479.

[11] Chen, S. F., and Goodman, J. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (1996), Association for Computational Linguistics, pp. 310–318.

[12] Dziadzio, S., Nabozny, A., Smywinski-Pohl, A., and Ziółko, B. Comparison of language models trained on written texts and speech transcripts in the context of automatic speech recognition. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on* (2015), IEEE, pp. 193–197.

[13] Fagan, S., and Gencay, R. An introduction to textual econometrics. *Handbook of empirical economics and finance* (2011), 133–54.

[14] Fink, G. Configuration of Hidden Markov Models. In *Markov Models for Pattern Recognition*. Springer Berlin Heidelberg, 2008, pp. 127–136.

[15] Fletcher, H. Loudness, masking and their relation to the hearing process and the problem of noise measurement. *The Journal of the Acoustical Society of America 9*, 4 (1938), 275–293.

[16] Gajecki, L. *Natural language modeling of Polish language for purposes of construction Large Vocabulary Continuous Speech Recognition system*. PhD thesis, 2013.

[17] Gale, W., and Church, K. What's wrong with adding one. In *Corpus-Based Research into Language. Rodolpi* (1994).

[18] Gales, M., and Young, S. The application of Hidden Markov Models in speech recognition. *Foundations and trends in signal processing 1*, 3 (2008), 195–304.

[19] Glass, J. Language modeling for speech recognition. 2003.

[20] Gussmann, E. *The phonology of Polish*. OUP Oxford, 2007.

[21] He, X., Deng, L., and Acero, A. Why word error rate is not a good metric for speech recognizer training for the speech translation task? In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), IEEE, pp. 5632–5635.

[22] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE 29*, 6 (2012), 82–97.

[23] Jassem, W. Polish. *Journal of the International Phonetic Association 33*, 01 (2003), 103–107.

[24] Jelinek, F. *Statistical methods for speech recognition*. MIT press, 1997.

[25] Juang, B.-H., and Rabiner, L. Mixture autoregressive Hidden Markov Models for speech signals. *Acoustics, Speech and Signal Processing, IEEE Transactions on 33*, 6 (1985), 1404–1413.

[26] Jurafsky, D., and Martin, J. *Speech and language processing. An introduction to natural language processing, computational linguistics, and speech.* Pearson Education„ 2000.

[27] Karpov, A., Kipyatkova, I. S., and Ronzhin, A. Speech recognition for east Slavic languages: the case of Russian. In *SLTU* (2012), pp. 84–89.

[28] Lewandowska-Tomaszczyk, B. Narodowy Korpus Języka Polskiego. *Wydawnictwo Naukowe PWN, Warsaw* (2012), 51–58.

[29] Lojka, M., and Juhár, J. Finite-state transducers and speech recognition in Slovak language. In *Signal Processing Algorithms, Architectures, Arrangements, and Applications Conference Proceedings (SPA), 2009* (2009), IEEE, pp. 149–153.

[30] Lu, L. Subspace Gaussian Mixture Models for automatic speech recognition.

[31] Majewski, P. Syllable based language model for large vocabulary continuous speech recognition of Polish. In *Text, Speech and Dialogue* (2008), Springer, pp. 397–401.

[32] Markoff, J. Scientists see promise in deep-learning programs, 2012. [Online; accessed: 14-August-2015].

[33] Michel, J., Shen, Y., Aiden, A., Veres, A., Gray, M., Pickett, J., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., et al. Quantitative analysis of culture using millions of digitized books. *science 331*, 6014 (2011), 176–182.

[34] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *Interspeech* (2010), vol. 2, p. 3.

[35] Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., and Khudanpur, S. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), IEEE, pp. 5528–5531.

[36] Mikolov, T., Kombrink, S., Deoras, A., Burget, L., and Cernocky, J. RNNLM-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop* (2011), pp. 196–201.

[37] Mohamed, A.-r., Dahl, G. E., and Hinton, G. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on 20*, 1 (2012), 14–22.

[38] Moravec, H. *Mind children: The future of robot and human intelligence.* Harvard University Press, 1988.

[39] Muda, L., Begam, M., and Elamvazuthi, I. Voice recognition algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) techniques. *arXiv preprint arXiv:1003.4083* (2010).

[40] Ney, H., Essen, U., and Kneser, R. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language 8*, 1 (1994), 1–38.

[41] Nouza, J., and Silovskỳ, J. Adapting lexical and language models for transcription of highly spontaneous spoken Czech. In *Text, Speech and Dialogue* (2010), Springer, pp. 377–384.

[42] Nouza, J., Zdansky, J., Cerva, P., and Silovsky, J. Challenges in speech processing of Slavic languages (case studies in speech recognition of Czech and Slovak). In *Development of Multimodal Interfaces: Active Listening and Synchrony*. Springer, 2010, pp. 225–241.

[43] Oparin, I., Glembek, O., Burget, L., and Černockỳ, J. Morphological random forests for language modeling of inflectional languages. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE* (2008), IEEE, pp. 189–192.

[44] Pan, J., Liu, C., Wang, Z., Hu, Y., and Jiang, H. Investigation of Deep Neural Networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMs in acoustic modeling. In *Chinese Spoken Language Processing (ISCSLP), 2012 8th International Symposium on* (2012), IEEE, pp. 301–305.

[45] Pauls, A., and Klein, D. Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1* (2012), Association for Computational Linguistics, pp. 959–968.

[46] Pohl, A., and Ziółko, B. Using part of speech n-grams for improving automatic speech recognition of Polish. In *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2013, pp. 492–504.

[47] Przepiórkowski, A. XML text interchange format in the National Corpus of Polish. In *The proceedings of Practical Applications in Language and Computers PALC 2009, Frankfurt am Main: Peter Lang* (2009), Citeseer.

[48] Rabiner, L. R., and Juang, B.-H. An introduction to Hidden Markov Models. *ASSP Magazine, IEEE 3*, 1 (1986), 4–16.

[49] Rosenfeld, R. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE 88*, 8 (2000), 1270–1278.

[50] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Cognitive modeling 5*, 3 (1988), 1.

[51] Schwenk, H., and Gauvain, J.-L. Training neural network language models on very large corpora. In *Proceedings of the conference on Human Language Technology and Empirical*

Methods in Natural Language Processing (2005), Association for Computational Linguistics, pp. 201–208.

[52] Singh, B., Rani, V., and Mahajan, N. Preprocessing in ASR for computer machine interaction with humans: A review. *International Journal of Advanced Research in Computer Science and Software Engineering 2*, 3 (2012), 396–399.

[53] Smith, N. A. Adversarial evaluation for models of natural language. *arXiv preprint arXiv:1207.0245* (2012).

[54] Stevens, S. S., Volkmann, J., and Newman, E. B. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America 8*, 3 (1937), 185–190.

[55] Stolcke, A., Zheng, J., Wang, W., and Abrash, V. SRILM at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop* (2011), vol. 5.

[56] Stuckless, R. Developments in real-time speech-to-text communication for people with impaired hearing. *Communication access for people with hearing loss* (1994), 197–226.

[57] Vetulani, Z., Obrębski, T., and Vetulani, G. Towards a lexicon-grammar of Polish: Extraction of verbo-nominal collocations from corpora. In *FLAIRS Conference* (2007), pp. 267–268.

[58] Waszczuk, J. Harnessing the CRF complexity with domain-specific constraints. the case of morphosyntactic tagging of a highly inflected language. In *COLING* (2012), pp. 2789–2804.

[59] Whittaker, E. W. D. *Statistical language modelling for automatic speech recognition of Russian and English*. PhD thesis, University of Cambridge, 2000.

[60] Yu, D., Deng, L., and Acero, A. The maximum entropy model with continuous features. In *NIPS Workshop, Whistler, BC, Canada* (December 2008), Microsoft.

[61] Yu, D., Deng, L., and Acero, A. Hidden conditional random field with distribution constraints for phone classification. In *Interspeech 2009* (September 2009), International Speech Communication Association.

[62] Zelasko, P., Ziółko, B., Jadczyk, T., and Pedzimąz, T. Linguistically motivated tied-state triphones for Polish speech recognition. In *Cybernetics (CYBCONF), 2015 IEEE 2nd International Conference on* (2015), IEEE, pp. 251–254.

[63] Ziółko, B., and Skurzok, D. N-grams model for Polish. *Speech and language technologies* (2011), 107–127.

[64] Ziółko, M., Gałka, J., Ziółko, B., Jadczyk, T., Skurzok, D., and Mąsior, M. Automatic speech recognition system dedicated for Polish. *Proceedings of Interspeech, Florence* (2011).