Fachbereich 4: Informatik

Institute for Web Science and Technologies

Implementation of Modified Kneser-Ney Smoothing on Top of Generalized Language Models for Next Word Prediction

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Informatik

vorgelegt von
Martin Christian Körner

Erstgutachter:	Prof. Dr. Steffen Staab Institute for Web Science and Technologies
Zweitgutachter:	René Pickhardt Institute for Web Science and Technologies

Koblenz, im September 2013

Erklärung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐
Der Text dieser Arbeit ist unter einer Creative Commons Lizenz (CC BY-SA 3.0) verfügbar.	☐	☐
Der Quellcode ist unter einer Creative Commons Lizenz (CC BY-SA 3.0) verfügbar.	☐	☐
Die erhobenen Daten sind unter einer Creative Commons Lizenz (CC BY-SA 3.0) verfügbar.	☐	☐

....

(Ort, Datum) (Unterschrift)

Zusammenfassung

Next Word Prediction beschreibt die Aufgabe, das Wort vorzuschlagen, welches ein Nutzer mit der höchsten Wahrscheinlichkeit als Nächstes eingeben wird. Momentane Ansätze basieren auf der Analyse sogenannter Corpora (große Textdateien) durch empirischen Methoden. Die resultierende Wahrscheinlichkeitsverteilungen über die vorkommenden Wortsequenzen werden als Language Models bezeichnet und zur Vorhersage des wahrscheinlichsten Wortes genutzt. Verbreitete Language Models basieren auf n-gram Sequenzen und Smoohting Algorithmen wie beispielsweise dem modifizierten Kneser-Ney Smoothing zur Anpassung der Wahrscheinlichkeit von ungesehenen Sequenzen. Vorherige Untersuchungen haben gezeigt, dass das Einfügen von Platzhaltern in solche n-gram Sequenzen zu besseren Ergebnissen führen kann, da dadurch die Berechnung von seltenen und ungesehenen Sequenzen weiter verbessert wird. Das Ziel dieser Arbeit ist die Formalisierung und Implementierung dieses neuen Ansatzes, wobei zusätzlich das modifizierte Kneser-Ney Smoothing eingesetzt werden soll.

Abstract

Next word prediction is the task of suggesting the most probable word a user will type next. Current approaches are based on the empirical analysis of corpora (large text files) resulting in probability distributions over the different sequences that occur in the corpus. The resulting language models are then used for predicting the most likely next word. State-of-the-art language models are based on n-grams and use smoothing algorithms like modified Kneser-Ney smoothing in order to reduce the data sparsity by adjusting the probability distribution of unseen sequences. Previous research has shown that building word pairs with different distances by inserting wildcard words into the sequences can result in better predictions by further reducing data sparsity. The aim of this thesis is to formalize this novel approach and implement it by also including modified Kneser-Ney smoothing.

Contents

1 Introduction

The task of *next word prediction* (NWP) as a part of *natural language processing* (NLP) can be applied in many areas where text is inserted using a keyboard. Trnka et al. [TMY+09] survey the usage of NWP in the field of *Augmentative and Alternative Communication* (AAC) with the result that NWP can significantly increase the communication rate. Studies indicate that the spelling of children or individuals with spelling disorders can be improved by NWP applications [NAB+92]. Another use case is improving the typing experience on small mobile phone keyboards.
Currently there are various implementations of NWP in use. The T9[1] input method for 12-key keypads uses a disambiguating algorithm based on a dictionary [SMK00]. In contrast to multi-press or two-key input methods, it is possible to input a character with only one key press. In case a key sequence can lead to more than one word, T9 suggests the most common word [SMK00]. Another example for the usage of NWP is the SwiftKey[2] application for Android operation systems. It allows users to input words by swiping over the virtual keyboard. A positive user experience is achieved with an algorithm for NWP, context-based learning, personalization, and adaptive learning over time [Tou13].
In general, there are two different approaches on NLP: The rationalist and the empiricist approach. Manning and Schütze [MS99] describe the fundamental difference between those two approaches based on how the human mind processes language. In summary, the rationalist approach is based on the theory that a major part of the human knowledge is fixed in advance. The empiricist approach, on the other hand, assumes that the human knowledge is built by applying a smaller set of general operations for processing of sensory inputs.
Implementing computational methods for NLP, based on the rationalist approach, is achieved by manually encoding information about the language [BM97]. However, this hand coding of information could so far only be applied on very small 'toy' problems instead of resulting in practical solutions [MS99].
Recent developments in the area of NLP are based on the empiricist approach by using statistical, pattern recognition, and machine learning methods applied on corpora[3] to build a specified general language model which provides a knowledge base for NLP [MS99]. [CM93] argue that the revival of empiricism was accelerated by the increase of computational power, the numerous data collections available, and the general focus of industry and government on developing practical NLP applications. Based on the rapid growth of the Internet, large corpora like the Wikipedia XML dumps[4] or the Enron Corpus [KY04] can now be used for building language models.
When using language models for NWP, the conditional probability of a word w_i following the sequence a user has typed is based on the number of occurrences of w_i following

[1]Patented by Tegic Communications, Inc. (Seattle, WA)
[2]`http://www.swiftkey.net` (Accessed: September 4, 2013)
[3]corpus is Latin for body; in this case meaning "large set of texts"
[4]`http://dumps.wikimedia.org/` (Accessed: September 4, 2013)

this sequence. However, there are more possible combinations of words than any current text corpus can provide and this is why techniques are needed to approximate the probability of unseen or rarely seen sequences. Current state-of-the-art approaches are based on smoothing methods (see Section 3). In this thesis, a novel approach called *Generalized Language Models* (GLMs) is introduced. Also, the implementation of GLMs on relatively large corpora as well as the application of modified Kneser-Ney smoothing on current language models and GLMs is discussed.
In summary, the key contributions of this thesis are:

- Overview of language models in the context of NWP
- Introduction of GLMs
- Overview of state-of-the-art smoothing techniques and their application on GLMs
- Implementation of modified Kneser-Ney smoothing applied on GLMs
- Discussion of future work on this topic

The remainder of this thesis is organized as follows. In Section 2, language models as well as GLMs are introduced. Section 3 gives an overview of smoothing methods for language models. The implementation of GLMs and modified Kneser-Ney smoothing is discussed in Section 4 before summarizing the possible future work on this topic in Section 5.

2 Next Word Prediction with Language Models

In the context of natural language processing, the term language model is defined as "a probability distribution $P(s)$ over strings s that describes how often the string s occurs as a sentence in some domain of interest"[CG99]. For a sentence s consisting of the word sequence $w_1 \cdots w_l$, this probability $P(s)$ can be expressed as the product of its conditional probabilities:

$$\begin{aligned} P(s) =& P(w_1|\langle BOS\rangle) \times P(w_2|\langle BOS\rangle\ w_1) \times \cdots \\ & \times P(w_l|\langle BOS\rangle\ w_1 \cdots w_{l-1)} \times P(\langle EOS\rangle|\langle BOS\rangle\ w_1 \cdots w_l) \\ =& \prod_{i=1}^{l+1} P(w_i|\langle BOS\rangle\ w_1 \cdots w_{i-1}) \end{aligned} \tag{1}$$

where $\langle BOS\rangle$ is a token signalizing the beginning of a sentence and $\langle EOS\rangle$ the end of sentence [CG99].

NWP is not based on the probability of a whole sentence. Instead, the probability of a w_i appearing after the partial sentence $\langle BOS\rangle\ w_1 \cdots w_{i-1}$ the user has typed is calculated. Similar to the conditional probabilities in Equation 1, this is written as $P(w_i|\langle BOS\rangle\ w_i \cdots w_{i-1})$.

The task of finding the most likely next word based on a corpus that contains W different words is then formalized with

$$\mathrm{NWP}_1(\langle BOS\rangle\ w_1 \cdots w_{i-1}) := \Big\{\operatorname*{arg\,max}_{w_i \in W}\big(P(w_i|\langle BOS\rangle\ w_1 \cdots w_{i-1})\big)\Big\} \tag{2}$$

In order to retrieve a set of the k most likely words, this equation can be recursively extended to

$$\begin{aligned} &\mathrm{NWP}_k(\langle BOS\rangle\ w_1 \cdots w_{i-1}) := \mathrm{NWP}_{k-1}(\langle BOS\rangle\ w_1 \cdots w_{i-1}) \\ &\cup \Big\{ \operatorname*{arg\,max}_{w_i \in W \setminus \mathrm{NWP}_{k-1}(\langle BOS\rangle\ w_1 \cdots w_{i-1})} \big(P(w_i|\langle BOS\rangle\ w_1 \cdots w_{i-1})\big)\Big\} \end{aligned} \tag{3}$$

In other words, the word with highest $P(w_i|\langle BOS\rangle\ w_1 \cdots w_{i-1})$ in the set W, without the $k-1$ words that already have been retrieved, is added it to the recursively defined result set.

Yet, it is not practical to calculate $P(w_i|\langle BOS\rangle\ w_i \cdots w_{i-1})$ by applying empirical methods. The main reason is that empirical methods are based on counting how often a sequence occurs in a corpus (see Equation 7). Even with a large corpus, in many cases a sequence $\langle BOS\rangle\ w_i \cdots w_{i-1}$ is unseen or rarely seen in the corpus [MS99]. Thereby, the general definition of language models needs to be modified in order to reduce this data sparsity. One possible modification is provided by n-gram language models.

2.1 n-gram Language Models

An n-gram is the pair of a word sequence $w_{i-n+1} \ldots w_i$ containing n words and its according count, based on the occurrences of the sequence in a corpus. In order to simplify the following notations, a word sequence $w_x \ldots w_y$ is expressed as w_x^y [CG98]. The aim of n-gram language models is to approximate the probability $P(s)$ defined in Equation 1. Given a w_i, only the previous $n-1$ words are used as the context for calculation. This results in

$$P(s) = \prod_{i=1}^{l+1} P(w_i|\langle BOS\rangle w_1 \cdots w_{i-1}) \approx \prod_{i=1}^{l+1} P(w_i|w_{i-n+1}^{i-1}) \tag{4}$$

where w_{i-n+1}^{i-1} is the sequence $w_{i-n+1} \ldots w_{i-1}$. In addition, w_{-n+2} through w_0 are defined as $\langle BOS\rangle$ for sequences that start with a negative index, e.g. when $i = 1$ and $n = 3$ [CG98]. The word w_{l+1} is defined as $\langle EOS\rangle$ [CG98]. This approximation is referred to as a *Markov assumption* [JM80].

When applying this approximation on NWP, the conditional probability of a word following a sequence is not based on the whole sequence. Instead, it is based on the last $n-1$ words of the sequence [CG98]. Thereby, the definition in Equation 2 is approximated with

$$n\text{-gramNWP}_1(w_{i-n+1}^{i}) := \left\{\operatorname*{arg\,max}_{w_i \in W}\big(P(w_i|w_{i-n+1}^{i-1})\big)\right\} \tag{5}$$

Accordingly, Equation 3 is approximated with

$$\begin{aligned} n\text{-gramNWP}_k(w_{i-n+1}^{i}) :=& n\text{-gramNWP}_{k-1}(w_{i-n+1}^{i}) \\ & \cup \left\{\operatorname*{arg\,max}_{w_i \in W \setminus n\text{-gramNWP}_{k-1}(w_{i-n+1}^{i})}\big(P(w_i|w_{i-n+1}^{i-1})\big)\right\} \end{aligned} \tag{6}$$

Based on the sequence counts retrieved from a corpus, *maximum likelihood* (ML) estimation provides a simple way of calculating $P(w_i|w_{i-n+1}^{i-1})$ [MS99]:

$$P_{\text{ML}}(w_i|w_{i-n+1}^{i}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} \tag{7}$$

Here, $c(w_x^y)$ is the number of times the word sequence w_x^y occurs in the corpus [CG99]. A special case is the unigram[5] probability of the start tag. The analysis of the results of both the Kyoto Language Modeling Toolkit (*Kylm*)[6] and SRI Language Modeling Toolkit (*SRILM*)[7] has shown that the count of this tag is set to zero while computing the ML estimation (see Equation 8 for an example).

Figure 1 provides three sentences that will be used as an exemplary corpus for explaining NLP concepts in this thesis. They contain tags which mark the start (<s>) and end (</s>) of a sentence. Additionally, punctuation marks are separated from the preceding words.

[5]another term for 1-gram

[6]`http://www.phontron.com/kylm/` (Accessed: September 4, 2013)

[7]`http://www.speech.sri.com/projects/srilm/` (Accessed: September 4, 2013)

<s> This is a list containing the tallest buildings in San Francisco : </s>
<s> The Transamerica Pyramid is the tallest building in San Francisco . </s>
<s> 555 California Street is the 2nd-tallest building in San Francisco . </s>

Figure 1: Example sentences to demonstrate NLP concepts

The calculation of ML estimation for different sequence lengths is demonstrated in the following equations. A full result list can be found in Appendix A.2.

$$P_{\mathrm{ML}}(\textit{<s>}) = \frac{c(\textit{<s>})}{\sum_{w_i} c(w_i)} = \frac{0}{37} = 0 \tag{8}$$

$$P_{\mathrm{ML}}(\textit{Francisco}) = \frac{c(\textit{Francisco})}{\sum_{w_i} c(w_i)} = \frac{3}{37} \approx 0.08 \tag{9}$$

$$P_{\mathrm{ML}}(\textit{Francisco}|\textit{San}) = \frac{c(\textit{San Francisco})}{\sum_{w_i} c(\textit{San } w_i)} = \frac{c(\textit{San Francisco})}{c(\textit{San})} = \frac{3}{3} = 1 \tag{10}$$

$$\begin{aligned} P_{\mathrm{ML}}(\textit{building}|\textit{the tallest}) &= \frac{c(\textit{the tallest building})}{\sum_{w_i} c(\textit{the tallest } w_i)} \\ &= \frac{c(\textit{the tallest building})}{c(\textit{the tallest})} = \frac{1}{3} \approx 0.33 \end{aligned} \tag{11}$$

$$\begin{aligned} P_{\mathrm{ML}}(\textit{building}|\textit{is the 3rd-tallest}) &= \frac{c(\textit{is the 3rd-tallest building})}{\sum_{w_i} c(\textit{is the 3rd-tallest } w_i)} \\ &= \frac{c(\textit{is the 3rd-tallest building})}{c(\textit{is the 3rd-tallest})} = \frac{0}{0} = 0 \end{aligned} \tag{12}$$

This approach performs poorly when applied on typical corpora which can be explained with the statistical distribution of language described by Zipf's law [Zip49]. Essentially, it says that the human language consists of many word types with a low frequency and relatively few word types with a high frequency [MS99]. Word types that occur only once in a corpus are referred to as *hapax legomena* [MS99]. Table 1 shows that the percentage of sequences with count one increases with the length of the sequences. NWP prediction based on ML estimation suffers from this data sparsity. In case a sequence w_{i-n+1}^{i-1} occurs only once followed by w_i, $P_{\mathrm{ML}}(w_{i'}|w_{i-n+1}^{i-1})$ is 0 for every $w_{i'} \neq w_i$. If w_{i-n+1}^{i-1} does not occur in a corpus, no prediction can be made in a NWP task since $P_{\mathrm{ML}}(w_i|w_{i-n+1}^{i-1})$ is 0 for every w_i.
One approach to reduce the negative effects of data sparsity are smoothing techniques which are introduced in Section 3. Another approach is described in the following subsections.

Table 1: Percentage of n-grams with count one in the English Wikipedia (version date: February 4, 2013)

	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
$c(w_{i-n+1}^{i}) = 1$	0.5%	5.1%	21.1%	44.7%	64.4%

2.2 Generalized Language Models

A reduction of data sparsity in n-gram language models is achieved by replacing every word in a sequence w_{i-n+1}^{i} except the first and the last word with a wildcard word ($*$), resulting in the sequence t_{i-n+1}^{i} with $t_{i-n+1} = w_{i-n+1}$, $t_i = w_i$, and $t_j = *$ for $i - n + 1 < j < i$. The sequence name t is based on the working title "Typology" of a Jugend forscht[8] project by Paul Georg Wagner and Till Speicher[9] who introduced this approach on sequence counting for building language models. The sequence count $c(t_{i-n+1}^{i})$ is calculated by summarizing the counts of all n-grams that contain all the non-wildcard words on the according positions over all words on the wildcard positions. For example, $c(t_1^4)$ is calculated with

$$c(t_1^4) = c(w_1 \; * \; * \, w_4) = \sum_{w_2} \sum_{w_3} c(w_1 \, w_2 \, w_3 \, w_4) \tag{13}$$

Two concrete examples that are based on the sentences in Figure 1 are

$$c(\textit{the} \; * \; \textit{building}) = c(\textit{the tallest building}) + c(\textit{the 2nd-tallest building}) = 1 + 1 = 2 \tag{14}$$

$$\begin{aligned} c(\textit{<s>} \; * \; * \; * \; \textit{is}) =& c(\textit{<s> The Transamerica Pyramid is}) \\ &+ c(\textit{<s> 555 California Street is}) = 1 + 1 = 2 \end{aligned} \tag{15}$$

First empirical tests have shown that the insertion of wildcard words indeed reduces the number of sequences with low counts. Table 2 shows the percentage of t sequences that occur exactly once in the English Wikipedia in contrast to word sequences with the same length.

Table 2: Percentage of sequences with length n and count one in the English Wikipedia (version date: February 4, 2013)

	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
$c(w_{i-n+1}^{i}) = 1$	0.5%	5.1%	21.1%	44.7%	64.4%
$c(t_{i-n+1}^{i}) = 1$	0.5%	5.1%	8.0%	9.6%	10.1%

Building language models based on t sequences is achieved by splitting a word sequence w_{i-n+1}^{i} into $n - 1$ different t sequences (t_{i-n+1}^{i}, t_{i-n+2}^{i}, ..., t_{i-1}^{i}) and then calculating the conditional probabilities accordingly. The last step is to aggregate the different probabilities, for example by building the arithmetic mean.

The main reason why this approach could yield better results than the n-gram approach is its flexibility based on the splitting into t sequences. In case a sequence was unseen in the corpus, the probability in an n-gram language model is 0. Having the sequence from Equation 12, even sophisticated smoothing techniques like the modified Kneser-Ney interpolation (introduced in Section 3.3) cannot improve the result (see Appendix C for

[8]`http://www.jugend-forscht.de` (Accessed: September 4, 2013)

[9]`http://typology.de/about` (Accessed: September 4, 2013)

the Kneser-Ney calculation of Equation 12). In the same example, Typology language models are able to assign a probability to the sequence by building three t sequences

$$P_{\mathrm{ML}}(\textit{building}|\textit{is * *}) = \frac{c(\textit{is * * building})}{\sum_{w_i} c(\textit{is * * } w_i)} = \frac{c(\textit{is * * building})}{c(\textit{is * *})} = \frac{2}{3} \approx 0.67 \tag{16}$$

$$P_{\mathrm{ML}}(\textit{building}|\textit{the *}) = \frac{c(\textit{the * building})}{\sum_{w_i} c(\textit{the * } w_i)} = \frac{c(\textit{the * building})}{c(\textit{the *})} = \frac{2}{3} \approx 0.67 \tag{17}$$

$$\begin{aligned} P_{\mathrm{ML}}(\textit{building}|\textit{3rd-tallest}) &= \frac{c(\textit{3rd-tallest building})}{\sum_{w_i} c(\textit{3rd-tallest } w_i)} \\ &= \frac{c(\textit{3rd-tallest building})}{c(\textit{3rd-tallest})} = \frac{0}{0} = 0 \end{aligned} \tag{18}$$

As a last step, the results are aggregated by calculating the arithmetic mean.
It can be argued that by splitting a word sequence into independent t sequences, the semantic structure of the word sequence is not taken into account during the calculation. A trade-off between the reduced data sparsity of t sequences and the semantic information of the word sequence can be achieved by varying the number and position of inserted wildcard words.
In the example above, inserting only one wildcard word at the third position would result in

$$P_{\mathrm{ML}}(\textit{building}|\textit{is the *}) = \frac{c(\textit{is the * building})}{\sum_{w_i} c(\textit{is the * } w_i)} = \frac{c(\textit{is the * building})}{c(\textit{is the *})} = \frac{2}{2} = 1 \tag{19}$$

and thereby improve the result (compare with Appendix C.2).
In order to be able to express the variation of number and position of wildcard words, $\delta_{i-n+1}^{i}[x_1^n]$ is defined as a sequence $\delta_{i-n+1}[x_1]\ \delta_{i-n+2}[x_2]\ \ldots \delta_i[x_n]$ where

$$\delta_j[x_{j-i+n}] = \begin{cases} w_j & \text{if } x_{j-i+n} = 1 \\ * & \text{if } x_{j-i+n} = 0 \\ \emptyset & \text{if } x_{j-i+n} = \otimes \end{cases} \tag{20}$$

and where w_{i-n+1}^{i} is a word sequence and x_1^n is a sequence of the tokens 1, 0, and $\otimes$.
In other words, x_1^n encodes the position of the wildcard words in the sequence. The option to leave out words by using $\otimes$ has been included in order to simplify the later definition of Generalized Language Models (see Section 2.2.1). Given the sequence "*<s> 345 California Center is the 3rd-tallest building*", two examples are

$$\delta_5^8[1\ 1\ 0\ 1] = \textit{is the * building} \tag{21}$$

$$\delta_5^7[\otimes\ 1\ 0] = \emptyset\ \textit{the *} = \textit{the *} \tag{22}$$

As a next step, generalized n-grams are defined as sequences $\delta_{i-n+1}^{i}[x_1^n]$ and their according counts $c(\delta_{i-n+1}^{i}[x_1^n])$. They are called generalized n-grams since the definition

of $\delta_{i-n+1}^{i}[x_1^n]$ includes the word sequences w_{i-n+1}^{i} as well as the t sequence t_{i-n+1}^{i}. For example, $\delta_5^8[1111]$ equals w_5^8 and $\delta_5^8[1001]$ equals t_5^8. Similar to calculating the count of t sequences, the count of $\delta_{i-n+1}^{i}[x_1^n]$ is calculated by summarizing the counts of all n-grams that contain all the non-wildcard words on the according positions over all words at the wildcard positions (see Equations 13, 14, and 15).
Generalized n-grams can provide a trade-off between the reduced data sparsity of t sequences and the syntactic information in n-grams. The trade-off between data sparsity and semantic information can also be observed by looking at the percentage of sequences that occur once in the corpus in the total number of sequences of the same type (see Section 2.2.2).
Generalized Language Models allow the insertion of wildcard words at varying positions by using generalized n-grams. They are defined in the following subsection.

2.2.1 Definition

Based on the definition of $\delta_{i-n+1}^{i}[x_1^n]$, the conditional probability $P(w_i|w_1^{i-1})$ can be estimated by calculating the arithmetic mean over different generalized n-gram sequences. This is formalized with

$$P(w_i|\langle BOS\rangle w_1^{i-1}) \approx \frac{\sum_{x_1^n \in X} P(\delta_i[x_n]|\delta_{i-n+1}^{i-1}[x_1^{n-1}])}{|X|} \tag{23}$$

where X is a set of sequences x_1^n and $|X|$ is the cardinality of that set. Also, $x_n = 1$ for every $x_1^n \in X$ since it is always the conditional probability of a word that is calculated. Thereby, $\delta_i[x_n]$ can be replaced with w_i resulting in

$$\frac{\sum_{x_1^n \in X} P(\delta_i[x_n]|\delta_{i-n+1}^{i-1}[x_1^{n-1}])}{|X|} = \frac{\sum_{x_1^n \in X} P(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}])}{|X|} \tag{24}$$

Generalized Language Models (GLM) then use this approximation to estimate $P(s)$ (see Equation 1) with

$$\prod_{i=1}^{l+1} P(w_i|\langle BOS\rangle w_1^{i-1}) \approx \prod_{i=1}^{l+1} \frac{\sum_{x_1^n \in X} P(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}])}{|X|} \tag{25}$$

where X is similar defined to the X in Equation 23. By being able to use $\otimes$ in x_1^{n-1}, it is possible to also include results from lower order language models. For example, when $X = \{\otimes\ \otimes\ 1, \otimes\ 1\ 1, 1\ 1\ 1\}$, the resulting GLM represents the arithmetic mean of the results of unigram, bigram, and trigram language models (see Section 2.2.2).
Similar to n-gramNWP in Equation 5, it is possible to use the GLM approach to approximate the NWP task described in Equation 2. The word with the highest probability can be retrieved with

$$\text{genNWP}_1(w_{i-n+1}^{i}) := \Big\{\arg\max_{w_i \in W}\big(\frac{\sum_{x_1^n \in X} P(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}])}{|X|}\big)\Big\} \tag{26}$$

The set of the k words with the highest probability is again recursively retrieved with

$$\begin{aligned}\mathrm{genNWP}_k(w_{i-n+1}^i) :=& \mathrm{genNWP}_{k-1}(w_{i-n+1}^i) \\ & \cup \Big\{ \underset{w_i \in W \setminus \mathrm{genNWP}_{k-1}(w_{i-n+1}^i)}{\arg\max} \big(\frac{\sum_{x_1^n \in X} P(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}])}{|X|}\big)\Big\}\end{aligned} \quad (27)$$

Again, W is the set of different words in a corpus.
Arguably, the quality of that approximation depends on the choice of sequences in X. Empirical evaluation metrics like the perplexity of the different resulting GLMs can be used to optimize X [CG99]. The following subsection shows a possible classification of different combinations of x_1^n for X.

2.2.2 Classification

The definition in Equation 25 describes GLMs in a rather generic way. A classification of different types of GLMs based on the sequences in X is necessary for structuring the later evaluation. The Pascal's triangle provides a way of distinguishing between different types of GLMs. Figure 2 shows a modified version of Pascal's triangle. Instead of $\binom{n}{k}$ it shows the possible variations of x_1^{n-1} for $\delta_{i-n+1}^i[x_1^n]$ where k equals the number of words that are used in the sequence. Because x_n always equals 1 since $\delta_i[x_n]$ is always a w_i, it is not shown in the figure. The figure also does not include sequences that start with a zero. This is based on the assumption that the information of $\delta_{i-n+1}^i[x_1^n]$ with $x_1 = 0$ is also contained in the associated lower order generalized n-gram sequence $\delta_{i-n+2}^i[x_2^n]$ and thereby redundant. Having the second sentence from Figure 1, an example for this assumption is

$$\delta_5^8[0\ 1\ 1\ 1] = \textit{* the tallest building} \approx \textit{the tallest building} = \delta_6^8[1\ 1\ 1] \quad (28)$$

This assumption narrows down the number of possible variations and thereby simplifies the calculation and evaluation. However, the effects of this simplification have to be analyzed. As a last constraint, the figure does not include sequences that leave out words inside a sequence. For example, a sequence $\delta_6^8[1\ \otimes\ 1]$ is not allowed since this would corrupt the positioning of the words in the resulting word sequence. The triangle again shows that n-gram sequences and t sequences are a part of generalized n-gram sequences. All sequences with $k = 1$ equal the according t sequences and sequences with $n - k = 1$ are n-gram sequences.
In order to specify groupings of sequences for a later evaluation, it is helpful to analyze the percentage of resulting word sequences that occur only once in a corpus similar to Table 2. Figure 3 shows this percentage based on the English Wikipedia, again in a Pascal's triangle but in contrast to Figure 2 it builds the arithmetic mean over the groups with more than one possible sequence. For example, the percentage of $n = 4$ and $k = 2$ equals the percentage of the sequences with $x_1^n = 1\ 1\ 0\ 1$ and $x_1^n = 1\ 0\ 1\ 1$ divided by 2 (see Appendix B for all values). The results show that the percentage of sequences with low counts depends more on the number of words that are used for predicting than on

the actual sequence length. For example, the percentage of $n = 5$ and $k = 1$ is lower than $n = 3$ and $k = 2$. The percentages also underline that sequences that lie between the t sequences on the left and n-gram sequences on the right provide a trade-off between the two concepts in terms of data sparsity. Another way of reducing the data sparsity of n-grams is described in the following section.

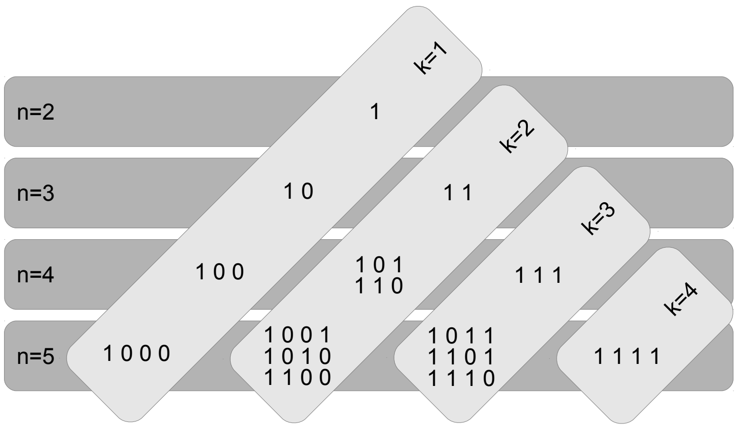

Figure 2: Pascal's triangle with variations of x_1^{n-1} instead of $\binom{n}{k}$ where k equals the number of used words in the sequence and $x_n = 1$

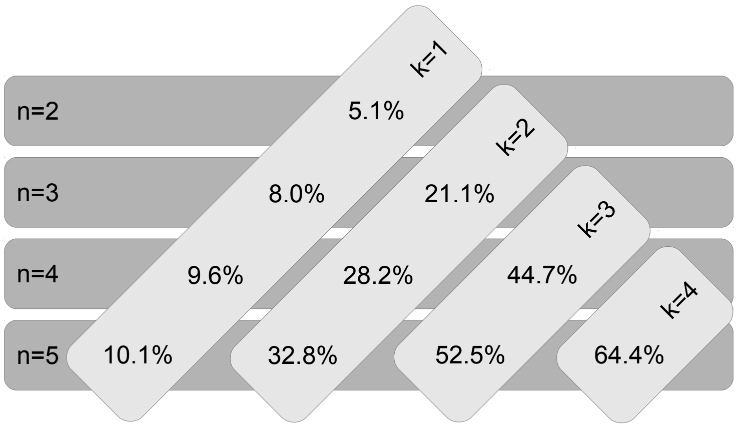

Figure 3: Pascal's triangle according to Figure 2 with average percentage of sequences that occur once in the English Wikipedia (version date: February 4, 2013)

3 Smoothing Algorithms for Language Models

Smoothing methods address the data sparsity problem of ML estimation based on word sequences by assigning reasonable probabilities to sequences that did not occur in the corpus. This requires the probability mass of occurring sequences to be reduced, resulting in a generally "smoother" distribution of probabilities [MS99]. In general, one can distinguish between backoff and interpolated smoothing models. If a sequence w_{i-n+1}^{i} does not occur in the corpus, backoff methods instead "back off" to a lower-order sequence w_{i-n+2}^{i}, using its probability reduced by a scaling factor $\gamma(w_{i-n+1}^{i-1})$ [CG99]. Otherwise, a distribution $\tau(w_i|w_{i-n+1}^{i-1})$ is used for calculating the probability $P_{\text{back}}(w_i|w_{i-n+1}^{i-1})$ with [CG99]

$$P_{\text{back}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \tau(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) P_{\text{back}}(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0 \end{cases} \tag{29}$$

In contrast to backoff models, interpolated smoothing models also use lower-order sequences for calculating the probability when the higher-order sequence occurs in the corpus resulting in [CG99]

$$P_{\text{inter}}(w_i|w_{i-n+1}^{i-1}) = \tau(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1}) P_{\text{inter}}(w_i|w_{i-n+2}^{i-1}) \tag{30}$$

In both models, the scaling factor $\gamma(w_{i-n+1}^{i-1})$ is used to let the probabilities sum up to one. It is also possible to transform a backoff smoothing method into an interpolated version [CG99].

Chen and Goodman [CG99] compare common smoothing models and also propose a *modified Kneser-Ney smoothing* (MKNS) algorithm which can be seen as the current state-of-the-art for smoothing in NLP as discussed in [FKJ06] and [CKFJ11]. It is thereby used in recent publications like [HKL13], [HSN13], [GJM$^+$13], and [ZTQG13]. For a later evaluation of Generalized Language Models, MKNS applied on n-gram language models is used as the base line.

MKNS is based on Kneser-Ney smoothing [KN95] which uses absolute discounting [NE91]. Since the intuition behind absolute discounting is based on the Good-Turing estimate [Goo53, CG99], all three models are described in the following subsections before introducing MKNS in Section 3.4. Other smoothing methods like additive smoothing or Witten-Bell smoothing are shown to perform worse than MKNS and are thereby not discussed in this thesis [CG98].

3.1 Good-Turing Estimate

The Good-Turing estimate [Goo53] is based on the assumption that the probability of unseen sequences can be estimated based on N_1 which is the number of n-grams that occurred once in the corpus [CG99]. The estimated probability of unseen n-grams p_0 is calculated with

$$p_0 = \frac{N_1}{N} \tag{31}$$

where N is the total number of n-grams observed in the corpus [GS95].
Since the probability mass of unseen n-grams is now greater than 0, the probability mass of seen n-grams needs to be reduced to let the total probability sum to 1 [GS95]. This is achieved with

$$p_r = \frac{r^*}{N} \tag{32}$$

where

$$r^* = (r+1)\frac{N_{r+1}}{N_r} \tag{33}$$

Here, N_r is the frequency of n-grams that were seen r times in the corpus and r^* is the adjusted frequency r [CG99]. Gale et al. [GS95] and Chen and Goodman [CG98] explain the Good-Turing estimate in more detail.
In practice, this smoothing method is not used in its original form. Chen and Goodman [CG98] argue that the Good-Turing estimate does not perform well, since higher level models are not combined with lower level models during the calculation. Thus, it is neither a backoff nor an interpolated smoothing algorithm. Still, the intuition behind the Good-Turing estimate is used in several other smoothing algorithms.

3.2 Absolute Discounting

Church et al. [CG91] empirically show that the discount $(r - r^*)$ applied at the Good-Turing estimate is generally constant for $r \geq 3$ [CG99]. Under this assumption, the calculation of r^* (see Equation 33) can be simplified to

$$r^* = r - D \tag{34}$$

where $0 \leq D \leq 1$.
Ney et al. [NEK94] estimate the discount value D based on the total number of n-grams occurring exactly once (n_1) and twice (n_2) [CG99]:

$$D = \frac{n_1}{n_1 + 2n_2} \tag{35}$$

Chen and Goodman [CG99] also calculate D by "optimizing the perplexity of held-out data"[CG99] and achieve better results than by using the above variant.
Absolute discounting [NE91, NEK94] is an interpolated smoothing method that uses the above assumption resulting in the following equation [CG99]:

$$P_{\text{abs}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{c(w_{i-n+1}^{i-1})} + \gamma(w_{i-n+1}^{i-1})P_{\text{abs}}(w_i|w_{i-n+2}^{i-1}) \tag{36}$$

where the scaling factor $\gamma(w_{i-n+1}^{i-1})$ is defined as

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D}{c(w_{i-n+1}^{i-1})}N_{1+}(w_{i-n+1}^{i-1}\bullet) \tag{37}$$

with

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i}) > 0\}| \tag{38}$$

Thereby, the lower order probability has a higher weight if the higher order sequence w_{i-n+1}^{i-1} has a low count and a high number of different preceding words.

3.3 Kneser-Ney Smoothing

Kneser-Ney smoothing [KN95] is a modified version of absolute discounting. The idea is to optimize the calculation of the lower-order n-gram probabilities in case the higher-order n-gram was unseen in the corpus. It is thereby originally a backoff smoothing algorithm [KN95]. The high-level motivation is that, using the backoff version of absolute discounting, the information that the higher-order n-gram was unseen is not taken into account when backing-off and calculating the lower-order probability [KN95]. Chen and Goodman [CG98] argue that for example the word *"Francisco"* appears relatively often, but only after a few words, most likely after *"San"*. Since *"Francisco"* is relatively frequent, absolute discounting assigns a relatively high probability to it. However , the unigram probability of *"Francisco"* is only used when backing-off from the higher bigram calculation, because the bigram count is 0. If the bigram is *"San Francisco"*, the count is greater than 0 and no backing-off is necessary in the first place. Thereby, the unigram probability should not be based on the number of occurrences, but on the number of different words it can follow [CG98]. In order to demonstrate this effect, the words *"San Francisco"* were used in every sentence in Figure 1 resulting in a high unigram probability of *"Francisco"* when using ML estimation (see Equation 9). When using Kneser-Ney smoothing, the resulting probability is smaller (see Equation 45).
Chen and Goodman [CG98] propose a variation of the original Kneser-Ney smoothing by transforming it into an interpolation smoothing method and also show that the interpolated version performs better than the backoff version. Thereby, the interpolated version is explained in more detail in the following.
At the lowest level (unigrams) of calculation, no interpolation is possible since there is no lower order probability available. This results in [CG99]

$$P_{\mathrm{KN}}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)} \tag{39}$$

where

$$N_{1+}(\bullet w_i) = |\{w_{i-1} : c(w_{i-1}^{i}) > 0\}| \tag{40a}$$

$$N_{1+}(\bullet\bullet) = |\{(w_{i-1}, w_i) : c(w_{i-1}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_i) \tag{40b}$$

In other words, $N_{1+}(\bullet w_i)$ is the number of words that precede w_i at least once in the corpus. At the higher levels (from bigrams to second-highest n-grams) of calculation, the following equation is used [CG99]:

$$\begin{aligned}P_{\text{KN}}(w_i|w_{i-n+1}^{i-1}) =& \frac{\max\{N_{1+}(\bullet w_{i-n+1}^{i}) - D, 0\}}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} \\ &+ \frac{D}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} N_{1+}(w_{i-n+1}^{i-1}\bullet) P_{\text{KN}}(w_i|w_{i-n+2}^{i-1})\end{aligned} \tag{41}$$

where

$$N_{1+}(\bullet w_{i-n+1}^{i}) = |\{w_{i-n} : c(w_{i-n}^{i}) > 0\}| \tag{42a}$$

$$N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet) = |\{(w_{i-n}, w_i) : c(w_{i-n}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_{i-n+1}^{i}) \tag{42b}$$

The discount value D is calculated similar to absolute discounting.
The weight $\frac{D}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} N_{1+}(w_{i-n+1}^{i-1}\bullet)$ determines the impact of the lower order value on the result. The fraction represents the amount that was subtracted from the higher order result using the discount value D. Following the intuition of absolute discounting, the lower order value is more relevant if the sequence w_{i-n+1}^{i-1} has a low count and a high number of different preceding words. Therefore, the weight is multiplied by $N_{1+}(w_{i-n+1}^{i-1}\bullet)$.
The highest level (highest n-grams) of calculation uses absolute counts instead of continuation counts and thereby equals the highest level of absolute discounting [CG98]:

$$\begin{aligned}P_{\text{KN}}(w_i|w_{i-n+1}^{i-1}) =& \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{c(w_{i-n+1}^{i-1})} \\ &+ \frac{D}{c(w_{i-n+1}^{i-1})} N_{1+}(w_{i-n+1}^{i-1}\bullet) P_{\text{KN}}(w_i|w_{i-n+2}^{i-1})\end{aligned} \tag{43}$$

A special case for calculating Kneser-Ney smoothing are sequences that start with a tag marking the beginning of a sentence. $N_{1+}(\bullet w_{i-n+2}^{i})$ is the number of different preceding words of w_{i-n+2}^{i}. Based on the definition of n-gram language models in Equation 4, it is possible to argue that only start tags can precede sequences that start with a start tag. Thereby, $N_{1+}(\bullet w_{i-n+2}^{i})$ would always be 1 for those sequences. Instead, the analysis of results of both *Kylm* and *SRILM* toolkits has shown that in this case $N_{1+}(\bullet w_{i-n+2}^{i})$ equals the count of the sequence w_{i-n+2}^{i}. The reason could be that in general any word can appear before the start of a sentence.
In case the unigram probability is the highest level of calculation, Kneser-Ney smoothing equals the calculation of ML estimation since the absolute counts are used and no interpolation is possible (see Equation 44 and 45). Based on the corpus in Figure 1 and in comparison to the examples for ML estimation in Equation 8, 9, and 10, the following equations show the calculation of Kneser-Ney smoothing:

$$P_{\text{KN}}(\text{<s>}) = \frac{c(\text{<s>})}{\sum_{w_i} c(w_i)} = \frac{0}{31} = 0 \tag{44}$$

$$P_{\text{KN}}(\textit{Francisco}) = \frac{c(\textit{Francisco})}{\sum_{w_i} c(w_i)} = \frac{3}{37} \approx 0.08 \tag{45}$$

$$\begin{aligned} P_{\text{KN}}(\textit{Francisco}|\textit{San}) =& \frac{\max\{c(\textit{San Francisco}) - D, 0\}}{c(\textit{San})} \\ &+ \frac{D}{c(\textit{San})} N_{1+}(\textit{San}\bullet) \frac{N_{1+}(\bullet\textit{Francisco})}{N_{1+}(\bullet\bullet)} \\ =& \frac{\max\{3 - \frac{21}{21+2*5}, 0\}}{3} + \frac{\frac{21}{21+2*5}}{3} * 1 * \frac{1}{28} \\ \approx& \frac{2.32}{3} + \frac{0.68}{3} * 0.04 \approx 0.78 \end{aligned} \tag{46}$$

The calculation of $P_{\text{KN}}(\textit{building}|\textit{the tallest})$ and $P_{\text{KN}}(\textit{building}|\textit{is the 3rd-tallest})$ is included in Appendix C.

3.4 Modified Kneser-Ney Smoothing

Chen and Goodman [CG98] introduce a variation of Kneser-Ney smoothing where, depending on the sequence count, one out of three different discounting values is used instead of just one D. This approach is based on studies by [CG98] showing that sequences with a count of 1 or 2 require a different discount value than sequences with a higher count.
Thereby, after $P_{\text{MKN}}(w_i)$ being defined similar to $P_{\text{KN}}(w_i)$ in Equation 39, the higher levels are calculated with [CG98]

$$\begin{aligned} P_{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) =& \frac{\max\{N_{1+}(\bullet w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i})), 0\}}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} \\ &+ \gamma_{low}(w_{i-n+1}^{i-1}) P_{\text{MKN}}(w_i|w_{i-n+2}^{i-1}) \end{aligned} \tag{47}$$

where

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c > 2 \end{cases} \tag{48}$$

The discounting values D_1, D_2, and D_{3+} are defined as [CG98]

$$D_1 = 1 - 2D\frac{n_2}{n_1} \tag{49a}$$

$$D_2 = 2 - 3D\frac{n_3}{n_2} \tag{49b}$$

$$D_{3+} = 3 - 4D\frac{n_4}{n_3} \tag{49c}$$

with D similar to Equation 35 and n_3 and n_4 analogous to n_1 and n_2. The scaling factor $\gamma_{low}(w_{i-n+1}^{i-1})$ is defined as [CG98]

$$\gamma_{low}(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)} \tag{50}$$

where $N_1(w_{i-n+1}^{i-1}\bullet)$, $N_2(w_{i-n+1}^{i-1}\bullet)$, and $N_{3+}(w_{i-n+1}^{i-1}\bullet)$ are analogously defined to $N_{1+}(w_{i-n+1}^{i-1}\bullet)$ (see Equation 38):

$$N_1(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i}) = 1\}| \tag{51a}$$

$$N_2(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i}) = 2\}| \tag{51b}$$

$$N_{3+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i}) > 2\}| \tag{51c}$$

The highest level of calculation is defined as[CG98]

$$\begin{aligned} P_{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) =& \frac{\max\{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i})), 0\}}{c(w_{i-n+1}^{i-1})} \\ &+ \gamma_{high}(w_{i-n+1}^{i-1}) P_{\text{MKN}}(w_i|w_{i-n+2}^{i-1}) \end{aligned} \tag{52}$$

where

$$\gamma_{high}(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{c(w_{i-n+1}^{i-1})} \tag{53}$$

Because of the rather complex calculation of the discount values which needs a larger corpus than the one in Figure 1, it is not useful to demonstrate the calculation with an example. Except for the discount values, γ_{low}, and γ_{high}, the calculation equals Kneser-Ney smoothing.

3.5 Smoothing Methods Applied on GLM

All relevant smoothing methods for n-gram language models can be applied on generalized language models as well. For example, the adapted version of Kneser-Ney smoothing for the highest level of calculation (see Equation 43):

$$\begin{aligned} P_{\text{KN}}(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}]) =& \frac{\max\{c(\delta_{i-n+1}^{i}[x_1^{n}]) - D, 0\}}{c(\delta_{i-n+1}^{i-1}[x_1^{n-1}])} \\ &+ \frac{D}{c(\delta_{i-n+1}^{i-1}[x_1^{n-1}])} N_{1+}(\delta_{i-n+1}^{i-1}[x_1^{n-1}]\bullet) P_{\text{KN}}(w_i|\delta_{i-n+2}^{i-1}[x_2^{n-1}]) \end{aligned} \tag{54}$$

Here, w_{i-n+1}^{i} and its subsequences are replaced with $\delta_{i-n+1}^{i}[x_1^{n}]$ (defined in Section 2.2) and its according subsequences. Again, x_n always equals 1.

In addition, the assumption from Section 2.2.2 results in the following modification of $P_{\text{KN}}(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}])$:

$$P_{\text{KN}}(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}]) = \begin{cases} P_{\text{KN}}(w_i|\delta_{i-n+1}^{i-1}[x_1^{n-1}]) & \text{if } \delta_{i-n+1}[x_1] = w_{i-n+1} \\ P_{\text{KN}}(w_i|\delta_{i-n+2}^{i-1}[x_2^{n-1}]) & \text{if } \delta_{i-n+1}[x_1] = * \end{cases} \tag{55}$$

Thereby, all lower order sequences that start with a wildcard word are being skipped during the calculation based on the approximation in Equation 28. Figure 4 illustrates the resulting interpolation steps. Here, all sequences that have no outgoing arrow are interpolated with the unigram probabilities.
Other smoothing methods can be applied on GLMs in a similar way. For example, Typology language models can be expressed as a type of Generalized Language Models and thereby can also be smoothed.
In the following section, the implementation of modified Kneser-Ney smoothing on top of Generalized Language Models is discussed.

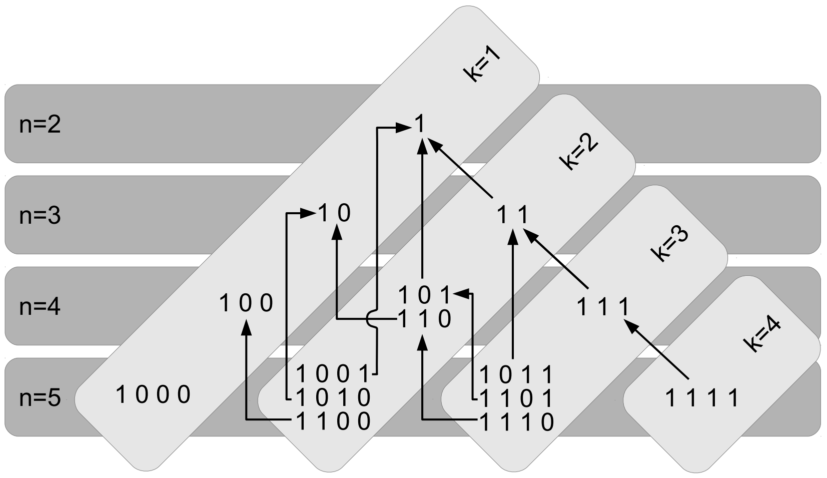

Figure 4: Illustration of interpolation steps for variations of x_1^{n-1} instead of $\binom{n}{k}$ where k equals the number of used words in the sequence and $x_n = 1$

4 Implementation

Modified Kneser-Ney smoothing (MKNS) is a state-of-the-art method for improving NWP based on n-gram language models. As described in Section 3.5, it can also be applied on generalized n-grams in order to build GLMs. The workflow of building smoothed GLMs consists of three steps:

1. Parse and prepare the dataset
2. Build generalized n-grams
3. Apply MKNS on generalized n-grams

Current implementations like the *SRILM* and *Kylm* toolkits only operate on n-gram language models. Another limitation of these toolkits is that the complete calculation of smoothed n-grams is performed in the main memory. For large datasets like the English Wikipedia, it is not possible to calculate language models on a computer with an average amount of main memory (8GB to 16GB). Thus, one requirement for the implementation that is discussed in this section is to be less dependent on the main memory during the calculation.
Before discussing the implementation of the three steps above, the different requirements for the implementation are listed in the following subsection.

4.1 Requirements

1. **The implementation has to be based on the programming language Java.**
   One reason is the existing experience with this programming language. Java also provides the possibility to write flexible and extensible code because of the object oriented approach.
2. **The source code has to be managed with the version control system Git[10] by using a public GitHub repository[11]**
   By using a version control system in a public repository, collaboration on the project can be simplified.
3. **The implementation has to scale in terms of memory usage.**
   Even for corpora like the English Wikipedia, the calculation has to be able to run on a computer with an average amount of main memory (8GB to max. 16GB).
4. **The implementation needs to cover n-gram language models as well as Generalized Language Models.**
   Both the language model calculation and modified Kneser-Ney smoothing need to be implemented for n-gram language models as well as Generalized Language Models.

[10] `http://git-scm.com/` (Accessed: September 4, 2013)
[11] `https://github.com/renepickhardt/typology` (Accessed: September 4, 2013)

5. **The implementation has to support the following datasets:** **Wikipedia**[12]**, Enron** [KY04]**, and JRC-Aquis** [SPW^{+}06].

6. **The implementation has to support datasets in the following languages: English, German, Italian, and French.**

7. **It should be possible to extend the implementation to support other datasets and languages.**

8. **The retrieval times have to be low enough to allow the evaluation of 100000 queries in less than 24 hours (less than 1 second per query).**

In the following subsections, the required tasks and design decisions for parsing the text, building generalized n-grams, and implementing modified Kneser-Ney smoothing will be discussed.

4.2 Text Parsing and Preparation

4.2.1 Required Tasks

Based on the rapid growth of the Internet, many large datasets are available for research. Some of them, for example the Wikipedia XML dumps[13], can be used under free licenses. In addition, Google provides collections of n-grams[14] that are based on the analysis of large amounts of books. The first version was published 2009 and is based on a corpus that contains more than 5 million digitized books [Mic11]. Yet, it is not possible to use the version of 2009 for calculating a language model that is smoothed with modified Kneser-Ney smoothing since there are no annotations of the beginning of a sentence in this dataset. The second version from 2012 contains these annotations and can thereby be used [LMA^{+}12]. Currently, this dataset is not used for building language models but it is a topic for the future work.
Most of the datasets that are available for research are stored in XML files. Thereby, the actual text needs to be extracted before building generalized n-grams. After extracting the text, sentences have to be separated in order to allow the correct tagging of the start and end of a sentence.

4.2.2 Design Decisions

The text extraction is based on the lexer-parser approach. Lexical analysis is used in compilers in order to provide information about the source program for syntax analysis [LSUA06]. This is achieved by grouping the input characters into lexemes which are then annotated with tokens [LSUA06]. These tokens describe the syntactic structure of the program.

[12]`http://dumps.wikimedia.org/` (Accessed: September 4, 2013)
[13]`http://en.wikipedia.org/wiki/Wikipedia:Copyrights` (Accessed: September 4, 2013)
[14]`http://books.google.com/ngrams/datasets` (Accessed: September 4, 2013)

This approach can also be used for describing the structure of XML files. Opening and closing XML tags are recognized during lexical analysis and then used as boundaries during the text extraction. For example, Figure 5 shows the structure of Wikipedia XML files which needs to be modeled in order to extract the content of each article. In the case of Wikipedia, specific syntactic annotations for tables, internal links, and text formation need to be removed as well. Table 3 shows an example annotation for parts of the XML structure in Figure 5.

```
<mediawiki xmlns="..." ... version="0.8" xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    ...
  </siteinfo>
  <page>
    <title>Berlin</title>
    <id>57</id>
    ...
    <revision>
      <id>327254</id>
      <timestamp>...</timestamp>
      ...
      <text xml:space="preserve">
        '''Berlin''' is the [[Capital of Germany|capital]]
        city of [[Germany]] and one of the 16
        [[states of Germany]]...
      </text>
    </revision>
  </page>
</mediawiki>
```

Figure 5: Example of Wikipedia XML structure

After the text has been extracted, line breaks are inserted after every sentence. As pointed out by Manning and Schütze [MS99], the task of correctly separating sentences is more complex than simply splitting after every period. For example, one has to take abbreviations like Prof. or etc. into account. In case the abbreviations appear at the end of a sentence, the period even has two functions [MS99]. Using a `BreakIterator`[15], Java provides a way for separating sentences by also taking language specific constructs into account. After separating the sentences, it is possible to separate punctuation marks from their preceding words in order to build generalized n-grams where punctuation marks are viewed as an entity.

[15]`http://docs.oracle.com/javase/7/docs/api/java/text/BreakIterator.html` (Accessed: September 4, 2013)

Table 3: Example annotations for Wikipedia XML (see Figure 5)

Lexeme	Token
`<text xml:space="preserve">`	Begin text
`'''Berlin'''`	Highlighting
`is`	Word
`[[Capital of Germany|capital]]`	Reference+Renaming
`[[Germany]]`	Reference
`</text>`	End text

4.3 Generalized n-gram Creation

4.3.1 Required Tasks

Before splitting the sentences into sequences, the start and end of a sentence has to be marked with a tag. On the one hand, this allows the computation of sentence probabilities. On the other hand, the additional information is needed for calculating Kneser-Ney smoothing (see Section 3.3).
In the next step, the sentences are broken down into sequences. For a maximal length n, there are 2^{n-1} different types of generalized n-grams possible. For example, if the maximal sequence is set to 5, 16 different types of sequences are created (see Table 11 in Appendix B). These sequences then have to be counted in a last step.

4.3.2 Design Decisions

For implementing Kneser-Ney smoothing, two start tags need to be added before each sentence. This is necessary for the calculation of continuation counts for sequences that start with a start tag. In order to simplify the calculation of smoothed generalized n-grams, two different start tags are being used. Having only one type of start tags, the position of this start tag is not clear in case the second start tag can be replaced by a wild card. For example, having $\delta_4^5[1011]$ with the resulting sequence "*<s> 555 California*" and only one type of start tag, it is not clear at which position the start tag was. For example, the original sequence could have been "*<s> The 555 California*" but also "*<s> <s> 555 California*"(as in Figure 1). In order to separate these two cases during calculation, the first inserted start tag is renamed to "*<fs>*" (for first start tag), resulting in sequences like "*<fs> <s> 555 California*".
When counting the sequences of large datasets like the English Wikipedia, they cannot be stored in main memory while counting when using a regular computer system (8GB to 16GB main memory). When extracting the text of the English Wikipedia, the resulting text file has a size of 8.2GB. During the calculation of 5-grams, the size of the resulting sequences is about five times higher and since 64% of these sequences occur only once (see Table 11 in Appendix B), the resulting set has the size of about 25GB. Thereby,

sequence counting is implemented by operating both on the hard disk drive and in the main memory. In order to do so, the counting task have to be separated into the following steps:

1. Build index based on word frequencies
2. Store occurring sequences into different files based on index
3. Sort the different files alphabetically
4. Count sequences and store sequences and scores

Since the resulting files of step 2 have to be small enough to be sorted efficiently in main memory in step 3, sequences are stored in different files. The decision which sequence is stored in what file is based on the first word of the sequence. In case, the size of the resulting file exceeds a defined limit, it is again divided into different files based on the second word of the sequence. After sorting the sequences inside the files alphabetically, the sequences are counted in step 4. In contrast to the Google n-gram collection that only includes n-grams that occurred at least 40 times [Mic11], sequences with a low frequency are currently not filtered. After all occurrences of a sequence are counted, it is stored together with the count, separated by a tab character (`\t`).

4.4 Modified Kneser-Ney Smoothing

4.4.1 Required Tasks

In general, the calculation of MKNS can be divided into the following three parts:

1. Calculate the lowest order result (see Equation 39)
2. Calculate the higher order results (see Equation 47)
3. Calculate the highest order results (see Equation 52)

In order to calculate these equations, several intermediate results have to be calculated first. Table 4 shows which results are used in the lowest, higher, and highest level of calculation. After calculating these values, the equations are calculated bottom up, starting with the lowest order calculation. This is necessary, since the equations are defined recursively and require the corresponding lower order results.

4.4.2 Design Decisions

In contrast to the *Kylm* and *SRILM* toolkits that calculate MKNS in main memory, the aim is to implement MKNS in a less memory-intensive way. Thereby, the calculation is divided into several steps that can be performed by iterating over sorted files. Figure 6 shows the order in which the intermediate results are calculated based on generalized n-grams. This can be performed by iterating over files that have been sorted alphabetically by their sequences. For aggregating the counts of $N_{1+}(\bullet w_{i-n+2}^{i})$, the generalized n-grams need to be sorted, starting at the second instead of the first word.

Table 4: Usage of intermediate results during different calculation phases

Type of result	Short	Lowest level	Higher levels	Highest level
$N_{1+}(\bullet w_{i-n+1}^{i})$	$N(\bullet w)$	x	x	
$N_{1+}(\bullet w_{i-n+1}^{i-1}\bullet)$	$N(\bullet w\bullet)$	x	x	
D_1, D_2, D_{3+}	D		x	x
$N_1(w_{i-n+1}^{i-1}\bullet)$, $N_2(w_{i-n+1}^{i-1}\bullet)$, $N_{3+}(w_{i-n+1}^{i-1}\bullet)$	$N(w\bullet)$		x	x

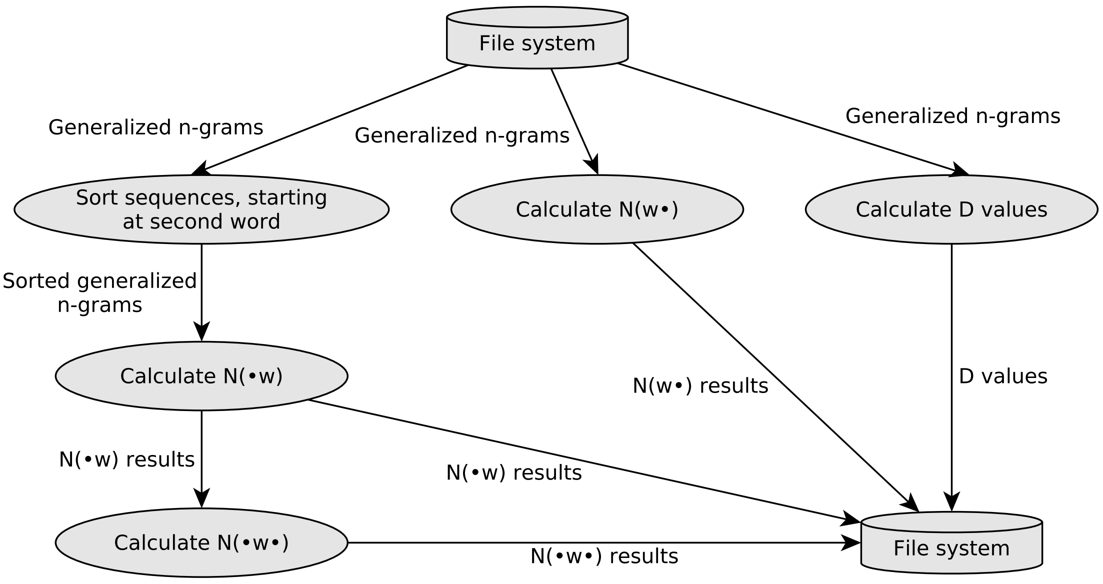

Figure 6: Calculating the intermediate results for MKNS (see Table 4 for the used abbreviations)

For calculating the lowest order MKNS results, the needed values are aggregated while iterating over the according files. Again, this is possible because the files are sorted alphabetically based on their sequences. While aggregating the higher and highest order MKNS results, the lower order results that were calculated previously need to be included due to the recursive step. At this point, the aggregation cannot be performed by iteration since the lower order results are based on the sequence w_{i-n+2}^{i} where as the higher order results are sorted by the longer sequence w_{i-n+1}^{i}.
One solution is to search the according lower order result. Yet, this approach is not practicable with the results not being stored in main memory. Even with good index structures, the lookup times are unacceptable.

Another solution requires the calculation of higher and highest order results to be broken down into three steps which are shown in Figure 7. After the first two parts have been calculated based on the sequence w_{i-n+1}^{i}, they are stored in text files on a hard disk drive. The next step is to sort the files alphabetically by they sequence, starting with the second word of the sequence. Thereby, it is possible to aggregate the higher order result, weight, and lower order result without searching. The final result is calculated by multiplying the weight with the lower order result and then adding the higher order result.

$$P_{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) = \underbrace{\frac{\max\{N_{1+}(\bullet w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i})), 0\}}{N_{1+}(\bullet w_{i-n+1}^{i-1} \bullet)}}_{\text{higher order probability}} + \underbrace{\gamma_{low}(w_{i-n+1}^{i-1})}_{\text{weight}} \underbrace{P_{\text{MKN}}(w_i|w_{i-n+2}^{i-1})}_{\text{lower order probability}}$$

Figure 7: Three parts of calculating MKNS equation (higher level)

By comparing the results of the *Kylm* and *SRILM* toolkit with this implementation for smaller datasets like the Bavarian Wikipedia[16], the smoothed probabilities can be verified. However, this is only possible for the standard n-gram models.
The main reason for not using popular file formats like the ARPA format[17] is the size of the datasets. Storing all results in one file reduces the usability of the results since the file cannot be loaded into main memory for reading or searching. In addition, results that are separated over a high number of files can also be loaded into databases more efficiently by creating a table for every file. This can reduce the lookup times when addressing the correct tables directly while retrieving. The next section discusses the future work on the topic of Generalized Language Models.

[16] `http://dumps.wikimedia.org/barwiki/` (Accessed: September 4, 2013)

[17] `http://www.speech.sri.com/projects/srilm/manpages/ngram-format.html` (Accessed: September 4, 2013)

5 Future Work

The implementation of MKNS on top of n-gram language models provides a baseline for evaluating Generalized Language models in the NWP task. Thereby, the next step in evaluating Generalized Language Models is to finalize the evaluation framework that compares smoothed n-gram language models with different versions of smoothed Generalized Language Models.
Besides evaluating Generalized Language Models in the NWP task, it is possible the apply evaluation metrics like the perplexity or cross-entropy on the resulting probability distribution. For example, Chen and Goodman [CG99] measure the cross-entropy in order to compare the performance of several smoothing techniques. Using these metrics allows the evaluation of Generalized Language Models in a more general way.
Especially the tasks of sorting and aggregating of intermediate results during the calculation of MKNS can be computed in parallel. This is possible because the results are independent and the calculation requires a relatively small amount of main memory. The *MapReduce* programming model [DG04] is one approach that could model the parallelization. In this case, the *map* step would be performed on different files in order to calculate the intermediate results. During the *reduce* step, the intermediate results would be aggregated to the final MKNS result.
Another topic for further research is the interpolation of lower order results in Generalized Language Models. Besides the interpolation that is discussed in Section 3.5 and shown in Figure 4, a different approach was suggested by Thomas Gottron. Instead of always removing the first word in a sequence, it is possible to instead replace another word with a wildcard word. Figure 8 illustrates the different options for the sequences $x_1^{n-1} = 11$ and $x_1^{n-1} = 1111$ and Figure 9 for $x_1^{n-1} = 1011$ and $x_1^{n-1} = 1100$. Every

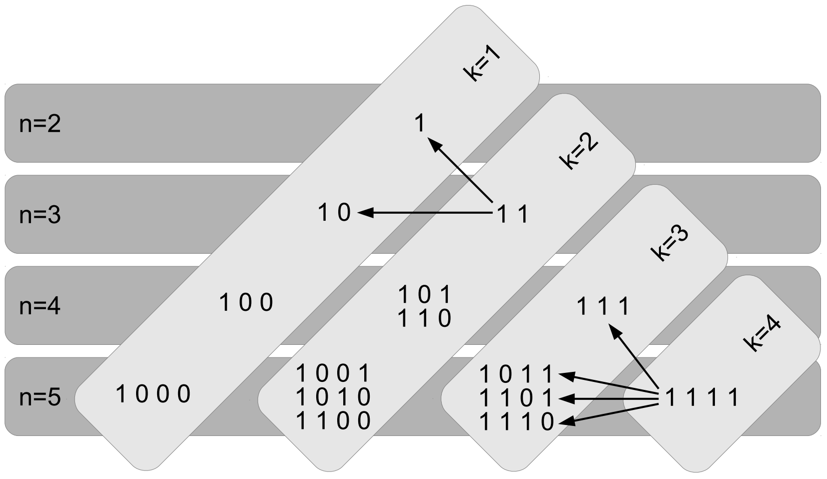

Figure 8: Illustration of alternative interpolation steps for the sequences $x_1^{n-1} = 1111$ and $x_1^{n-1} = 11$

word in a sequence can be replaced by a wildcard word except the first word, which is removed based on the assumption in Equation 55. This results in $k-1$ different possible ways of interpolating when k is the number of ones in the sequence x_1^{n-1}. Since the scaling factor γ (see Equation 30) is based on the higher order sequence, it does not have to be modified. Instead, the different interpolation results are aggregated, for example by calculating the arithmetic mean, before being multiplied with γ.

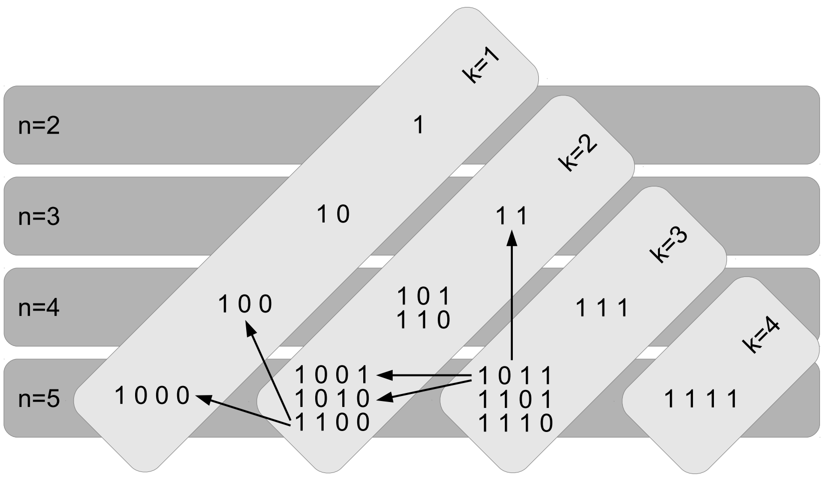

Figure 9: Illustration of alternative interpolation steps for the sequences $x_1^{n-1} = 1011$ and $x_1^{n-1} = 1100$

References

[BM97] Eric Brill and Raymond J. Mooney. An overview of empirical natural language processing. *AI magazine*, 18(4):13, 1997.

[CG91] Kenneth W. Church and William A. Gale. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech & Language*, 5(1):19–54, 1991.

[CG98] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical report, TR-10-98, Harvard University, August, 1998.

[CG99] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[CKFJ11] Boxing Chen, Roland Kuhn, George Foster, and Howard Johnson. Unpacking and transforming feature functions: New ways to smooth phrase tables. *Proceedings of MT Summit XIII, Xiamen, China*, 13:269–275, 2011.

[CM93] Kenneth W. Church and Robert L. Mercer. Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24, 1993.

[DG04] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *To appear in OSDI*, page 1, 2004.

[FKJ06] George Foster, Roland Kuhn, and Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 53–61, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[GJM$^+$13] Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. Contexttype: using hand posture information to improve mobile touch screen text entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2795–2798. ACM, 2013.

[Goo53] Irwin J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

[GS95] William Gale and Geoffrey Sampson. Good-turing smoothing without tears. *Journal of Quantitative Linguistics*, 2(3):217–237, 1995.

[HKL13] Kenneth Heafield, Philipp Koehn, and Alon Lavie. Grouping language model boundary words to speed k–best extraction from hypergraphs. In *Proceedings of NAACL-HLT*, pages 958–968, 2013.

[HSN13] Matthias Huck, Erik Scharwächter, and Hermann Ney. Source-side discontinuous phrases for machine translation: A comparative study on phrase extraction and search. *The Prague Bulletin of Mathematical Linguistics*, 99(1):17–38, 2013.

[JM80] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980.

[KN95] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE, 1995.

[KY04] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004*, volume 3201, pages 217–226, 2004.

[LMA$^+$12] Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. Syntactic annotations for the google books ngram corpus. In *Proceedings of the ACL 2012 System Demonstrations*, pages 169–174. Association for Computational Linguistics, 2012.

[LSUA06] Monica Lam, Ravi Sethi, Jeffrey Ullman, and Alfred Aho. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 2006.

[Mic11] Jean-Baptiste Michel. Quantitative analysis of culture using millions of digitized books. *science*, 1199644(176):331, 2011.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[NAB$^+$92] Alan Newell, John Arnott, Lynda Booth, William Beattie, Bernadette Brophy, and Ian Ricketts. Effect of the "pal" word prediction system on the quality and quantity of text generation. *Augmentative and Alternative Communication*, 8(4):304–311, 1992.

[NE91] Hermann Ney and Ute Essen. On smoothing techniques for bigram-based natural language modelling. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 825–828. IEEE, 1991.

[NEK94] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–28, 1994.

[SMK00] Miika Silfverberg, I. Scott MacKenzie, and Panu Korhonen. Predicting text entry speed on mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 9–16. ACM, 2000.

[SPW$^+$06] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. In *LREC'06: Proceedings of the 5th International Conference on Language Resources and Evaluation*, 2006.

[TMY$^+$09] Keith Trnka, John McCaw, Debra Yarrington, Kathleen F McCoy, and Christopher Pennington. User interaction with word prediction: The effects of prediction quality. *ACM Transactions on Accessible Computing (TACCESS)*, 1(3):17, 2009.

[Tou13] TouchType. Bringing swiftkey's innovation into focus. URL: `http://www.swiftkey.net/en/blog/bringing-swiftkeys-innovation-into-focus/`, 04 2013. Accessed: September 4, 2013.

[Zip49] George Kingsley Zipf. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, 1949.

[ZTQG13] Hui Zhang, Kristina Toutanova, Chris Quirk, and Jianfeng Gao. Beyond left-to-right: Multiple decomposition structures for SMT. In *Proceedings of NAACL-HLT*, pages 12–21, 2013.

Appendices

A Statistics of Corpus in Figure 1

A.1 Sequence Counts

Table 5: Absolute counts for n-grams with $1 \leq n \leq 3$

1-grams	cnt	2-grams	cnt	3-grams	cnt
.	2	. </s>	2	2nd-tallest building in	1
2nd-tallest	1	2nd-tallest building	1	555 California Street	1
555	1	555 California	1	<s> 555 California	1
:	1	: </s>	1	<s> The Transamerica	1
</s>	3	<s> 555	1	<s> This is	1
<s>	3 (0)	<s> The	1	California Street is	1
California	1	<s> This	1	Francisco . </s>	2
Francisco	3	California Street	1	Francisco : </s>	1
Pyramid	1	Francisco .	2	Pyramid is the	1
San	3	Francisco :	1	San Francisco .	2
Street	1	Pyramid is	1	San Francisco :	1
The	1	San Francisco	3	Street is the	1
This	1	Street is	1	The Transamerica Pyramid	1
Transamerica	1	The Transamerica	1	This is a	1
a	1	This is	1	Transamerica Pyramid is	1
building	2	Transamerica Pyramid	1	a list containing	1
buildings	1	a list	1	building in San	2
in	3	building in	2	buildings in San	1
is	3	buildings in	1	in San Francisco	3
list	1	in San	3	is a list	1
tallest	2	is a	1	is the 2nd-tallest	1
the	3	is the	2	is the tallest	1
containing	1	list containing	1	list containing the	1
		tallest building	1	tallest building in	1
		tallest buildings	1	tallest buildings in	1
		the 2nd-tallest	1	the 2nd-tallest building	1
		the tallest	2	the tallest building	1
		containing the	1	the tallest buildings	1
				containing the tallest	1
23	40 (37)	28	37	29	34

Table 6: Absolute counts for n-grams with $4 \leq n \leq 5$

4-grams	**cnt**	**5-grams**	**cnt**
2nd-tallest building in San	1	2nd-tallest building in San Francisco	1
555 California Street is	1	555 California Street is the	1
<s> 555 California Street	1	<s> 555 California Street is	1
<s> The Transamerica Pyramid	1	<s> The Transamerica Pyramid is	1
<s> This is a	1	<s> This is a list	1
California Street is the	1	California Street is the 2nd-tallest	1
Pyramid is the tallest	1	Pyramid is the tallest building	1
San Francisco . </s>	2	Street is the 2nd-tallest building	1
San Francisco : </s>	1	The Transamerica Pyramid is the	1
Street is the 2nd-tallest	1	This is a list containing	1
The Transamerica Pyramid is	1	Transamerica Pyramid is the tallest	1
This is a list	1	a list containing the tallest	1
Transamerica Pyramid is the	1	building in San Francisco .	2
a list containing the	1	buildings in San Francisco :	1
building in San Francisco	2	in San Francisco . </s>	2
buildings in San Francisco	1	in San Francisco : </s>	1
in San Francisco .	2	is a list containing the	1
in San Francisco :	1	is the 2nd-tallest building in	1
is a list containing	1	is the tallest building in	1
is the 2nd-tallest building	1	list containing the tallest buildings	1
is the tallest building	1	tallest building in San Francisco	1
list containing the tallest	1	tallest buildings in San Francisco	1
tallest building in San	1	the 2nd-tallest building in San	1
tallest buildings in San	1	the tallest building in San	1
the 2nd-tallest building in	1	the tallest buildings in San	1
the tallest building in	1	containing the tallest buildings in	1
the tallest buildings in	1		
containing the tallest buildings	1		
28	31	26	28

A.2 Maximum Likelihood Estimation

Table 7: Conditional probability of w_i in the context of w_1^{n-1} based on Maximum Likelihood estimation with $1 \leq n \leq 3$.

1-grams	P_{ML}	**2-grams**	P_{ML}	**3-grams**	P_{ML}
.	0.054	. </s>	1	2nd-tallest building in	1
2nd-tallest	0.027	2nd-tallest building	1	555 California Street	1
555	0.027	555 California	1	<s> 555 California	1
:	0.027	: </s>	1	<s> The Transamerica	1
</s>	0.081	<s> 555	0.333	<s> This is	1
<s>	0	<s> The	0.333	California Street is	1
California	0.027	<s> This	0.333	Francisco . </s>	1
Francisco	0.081	California Street	1	Francisco : </s>	1
Pyramid	0.027	Francisco .	0.667	Pyramid is the	1
San	0.081	Francisco :	0.333	San Francisco .	0.667
Street	0.027	Pyramid is	1	San Francisco :	0.333
The	0.027	San Francisco	1	Street is the	1
This	0.027	Street is	1	The Transamerica Pyramid	1
Transamerica	0.027	The Transamerica	1	This is a	1
a	0.027	This is	1	Transamerica Pyramid is	1
building	0.054	Transamerica Pyramid	1	a list containing	1
buildings	0.027	a list	1	building in San	1
in	0.081	building in	1	buildings in San	1
is	0.081	buildings in	1	in San Francisco	1
list	0.027	in San	1	is a list	1
tallest	0.054	is a	0.333	is the 2nd-tallest	0.5
the	0.081	is the	0.667	is the tallest	0.5
containing	0.027	list containing	1	list containing the	1
		tallest building	0.5	tallest building in	1
		tallest buildings	0.5	tallest buildings in	1
		the 2nd-tallest	0.333	the 2nd-tallest building	1
		the tallest	0.667	the tallest building	0.5
		containing the	1	the tallest buildings	0.5
				containing the tallest	1
23	1	28		29	

Table 8: **Conditional probability of w_n in the context of w_1^{n-1} based on Maximum Likelihood estimation with $4 \leq n \leq 5$.**

4-grams	P_{ML}	**5-grams**	P_{ML}
2nd-tallest building in San	1	2nd-tallest building in San Francisco	1
555 California Street is	1	555 California Street is the	1
<s> 555 California Street	1	<s> 555 California Street is	1
<s> The Transamerica Pyramid	1	<s> The Transamerica Pyramid is	1
<s> This is a	1	<s> This is a list	1
California Street is the	1	California Street is the 2nd-tallest	1
Pyramid is the tallest	1	Pyramid is the tallest building	1
San Francisco . </s>	1	Street is the 2nd-tallest building	1
San Francisco : </s>	1	The Transamerica Pyramid is the	1
Street is the 2nd-tallest	1	This is a list containing	1
The Transamerica Pyramid is	1	Transamerica Pyramid is the tallest	1
This is a list	1	a list containing the tallest	1
Transamerica Pyramid is the	1	building in San Francisco .	1
a list containing the	1	buildings in San Francisco :	1
building in San Francisco	1	in San Francisco . </s>	1
buildings in San Francisco	1	in San Francisco : </s>	1
in San Francisco .	0.667	is a list containing the	1
in San Francisco :	0.333	is the 2nd-tallest building in	1
is a list containing	1	is the tallest building in	1
is the 2nd-tallest building	1	list containing the tallest buildings	1
is the tallest building	1	tallest building in San Francisco	1
list containing the tallest	1	tallest buildings in San Francisco	1
tallest building in San	1	the 2nd-tallest building in San	1
tallest buildings in San	1	the tallest building in San	1
the 2nd-tallest building in	1	the tallest buildings in San	1
the tallest building in	1	containing the tallest buildings in	1
the tallest buildings in	1		
containing the tallest buildings	1		
28		26	

A.3 Kneser-Ney Smoothing

Table 9: Conditional probability of w_n in the context of w_1^{n-1} based on Kneser-Ney smoothing with $1 \leq n \leq 3$. For $n = 1$, P_{KN} is the probability of w_1.

1-grams	P_{KN}	**2-grams**	P_{KN}	**3-grams**	P_{KN}
.	0.054	. </s>	0.685	2nd-tallest building in	0.719
2nd-tallest	0.027	2nd-tallest building	0.371	555 California Street	0.417
555	0.027	555 California	0.347	<s> 555 California	0.417
:	0.027	: </s>	0.371	<s> The Transamerica	0.417
</s>	0.081	<s> 555	0.132	<s> This is	0.460
<s>	0	<s> The	0.132	California Street is	0.460
California	0.027	<s> This	0.132	Francisco . </s>	0.719
Francisco	0.081	California Street	0.347	Francisco : </s>	0.438
Pyramid	0.027	Francisco .	0.457	Pyramid is the	0.558
San	0.081	Francisco :	0.124	San Francisco .	0.479
Street	0.027	Pyramid is	0.395	San Francisco :	0.146
The	0.027	San Francisco	0.782	Street is the	0.558
This	0.027	Street is	0.395	The Transamerica Pyramid	0.417
Transamerica	0.027	The Transamerica	0.347	This is a	0.275
a	0.027	This is	0.395	Transamerica Pyramid is	0.460
building	0.054	Transamerica Pyramid	0.347	a list containing	0.417
buildings	0.027	a list	0.347	building in San	0.854
in	0.081	building in	0.685	buildings in San	0.708
is	0.081	buildings in	0.371	in San Francisco	0.806
list	0.027	in San	0.782	is a list	0.417
tallest	0.054	is a	0.124	is the 2nd-tallest	0.178
the	0.081	is the	0.473	is the tallest	0.447
containing	0.027	list containing	0.347	list containing the	0.438
		tallest building	0.210	tallest building in	0.719
		tallest buildings	0.185	tallest buildings in	0.438
		the 2nd-tallest	0.124	the 2nd-tallest building	0.438
		the tallest	0.457	the tallest building	0.241
		containing the	0.371	the tallest buildings	0.219
				containing the tallest	0.544
23	1	28		29	

Table 10: Conditional probability of w_n in the context of w_1^{n-1} based on Kneser-Ney smoothing with $4 \leq n \leq 5$.

4-grams	P_{KN}	**5-grams**	P_{KN}
2nd-tallest building in San	0.873	2nd-tallest building in San Francisco	0.874
555 California Street is	0.530	555 California Street is the	0.619
<s> 555 California Street	0.492	<s> 555 California Street is	0.535
<s> The Transamerica Pyramid	0.492	<s> The Transamerica Pyramid is	0.535
<s> This is a	0.369	<s> This is a list	0.497
California Street is the	0.615	California Street is the 2nd-tallest	0.324
Pyramid is the tallest	0.551	Pyramid is the tallest building	0.378
San Francisco . </s>	0.755	Street is the 2nd-tallest building	0.516
San Francisco : </s>	0.511	The Transamerica Pyramid is the	0.619
Street is the 2nd-tallest	0.317	This is a list containing	0.497
The Transamerica Pyramid is	0.530	Transamerica Pyramid is the tallest	0.555
This is a list	0.492	a list containing the tallest	0.607
Transamerica Pyramid is the	0.615	building in San Francisco .	0.664
a list containing the	0.511	buildings in San Francisco :	0.329
building in San Francisco	0.873	in San Francisco . </s>	0.758
buildings in San Francisco	0.746	in San Francisco : </s>	0.516
in San Francisco .	0.504	is a list containing the	0.516
in San Francisco :	0.170	is the 2nd-tallest building in	0.758
is a list containing	0.492	is the tallest building in	0.758
is the 2nd-tallest building	0.511	list containing the tallest buildings	0.359
is the tallest building	0.371	tallest building in San Francisco	0.874
list containing the tallest	0.603	tallest buildings in San Francisco	0.749
tallest building in San	0.873	the 2nd-tallest building in San	0.874
tallest buildings in San	0.746	the tallest building in San	0.874
the 2nd-tallest building in	0.755	the tallest buildings in San	0.749
the tallest building in	0.755	containing the tallest buildings in	0.516
the tallest buildings in	0.511		
containing the tallest buildings	0.352		
28		26	

B Statistics of the English Wikipedia

Table 11: Percentage of generalized n-grams with $c(\delta_1^n[x_1^n]) = 1$ in the English Wikipedia (version date: February 4, 2013)

x_1^n	Percentage of total count	Percentage of different sequences
1	0.500	64.000
11	5.140	68.220
101	8.0189	79.890
111	21.113	77.523
1001	9.583	72.126
1011	28.224	80.406
1101	28.214	80.696
1111	44.718	85.402
10001	10.133	72.741
10011	31.732	81.949
10101	35.331	83.038
10111	52.718	87.558
11001	31.547	82.240
11011	52.574	88.021
11101	52.307	87.678
11111	64.432	90.667

C Kneser-Ney Calculations

C.1 $P_{\mathrm{KN}}(\textbf{\textit{building}}|\textbf{\textit{the tallest}})$

$$
\begin{aligned}
&P_{\mathrm{KN}}(\textit{building}|\textit{the tallest})\\
=&\frac{\max\{c(\textit{the tallest building})-D,0\}}{c(\textit{the tallest})}+\frac{D}{c(\textit{the tallest})}N_{1+}(\textit{the tallest}\bullet)P_{KN}(\textit{building}|\textit{the})\\
=&\frac{\max\{c(\textit{the tallest building})-D,0\}}{c(\textit{the tallest})}\\
&+\frac{D}{c(\textit{the tallest})}N_{1+}(\textit{the tallest}\bullet)\Big(\frac{\max\{c(\textit{tallest building})-D,0\}}{c(\textit{tallest})}\\
&+\frac{D}{c(\textit{tallest})}N_{1+}(\textit{tallest}\bullet)P_{KN}(\textit{building})\Big)\\
=&\frac{\max\{c(\textit{the tallest building})-D,0\}}{c(\textit{the tallest})}\\
&+\frac{D}{c(\textit{the tallest})}N_{1+}(\textit{the tallest}\bullet)\Big(\frac{\max\{c(\textit{tallest building})-D,0\}}{c(\textit{tallest})}\\
&+\frac{D}{c(\textit{tallest})}N_{1+}(\textit{tallest}\bullet)\frac{N_{1+}(\bullet\textit{building})}{N_{1+}(\bullet\bullet)}\Big)\\
=&\frac{\max\{1-\frac{25}{25+2*3},0\}}{2}+\frac{\frac{25}{25+2*2}}{2}*2*\Big(\frac{\max\{1-\frac{21}{21+2*5},0\}}{2}+\frac{\frac{21}{21+2*5}}{2}*2*\frac{2}{28}\Big)\\
\approx&\frac{0.19}{2}+\frac{0.81}{2}*2*\Big(\frac{0.32}{2}+\frac{0.68}{2}*2*0.07\Big)\approx 0.1+0.81*0.2\approx 0.26
\end{aligned}
$$

(When calculated with higher precision, the result for $P_{\mathrm{KN}}(\textit{building}|\textit{the tallest})$ is 0.2408 or 24.08%)

C.2 $P_{\text{KN}}(\textbf{\textit{building}}|\textbf{\textit{is the 3rd-tallest}})$

$$
\begin{aligned}
&P_{\text{KN}}(\textit{building}|\textit{is the 3rd-tallest}) \\
=&\frac{\max\{c(\textit{is the 3rd-tallest building}) - D, 0\}}{c(\textit{is the 3rd-tallest})} \\
&+ \frac{D}{c(\textit{is the 3rd-tallest})} N_{1+}(\textit{is the 3rd-tallest}\bullet) P_{KN}(\textit{building}|\textit{the 3rd-tallest}) \\
=&\frac{\max\{c(\textit{is the 3rd-tallest building}) - D, 0\}}{c(\textit{is the 3rd-tallest})} \\
&+ \frac{D}{c(\textit{is the 3rd-tallest})} N_{1+}(\textit{is the 3rd-tallest}\bullet) \big(\\
&\frac{\max\{c(\textit{the 3rd-tallest building}) - D, 0\}}{c(\textit{the 3rd-tallest})} \\
&+ \frac{D}{c(\textit{the 3rd-tallest})} N_{1+}(\textit{the 3rd-tallest}\bullet) P_{KN}(\textit{building}|\textit{3rd-tallest}) \big) \\
=&\frac{\max\{c(\textit{is the 3rd-tallest building}) - D, 0\}}{c(\textit{is the 3rd-tallest})} \\
&+ \frac{D}{c(\textit{is the 3rd-tallest})} N_{1+}(\textit{is the 3rd-tallest}\bullet) \Big(\\
&\frac{\max\{c(\textit{the 3rd-tallest building}) - D, 0\}}{c(\textit{the 3rd-tallest})} + \frac{D}{c(\textit{the 3rd-tallest})} N_{1+}(\textit{the 3rd-tallest}\bullet) \big(\\
&\frac{\max\{c(\textit{3rd-tallest building}) - D, 0\}}{c(\textit{3rd-tallest})} \\
&+ \frac{D}{c(\textit{3rd-tallest})} N_{1+}(\textit{3rd-tallest}\bullet) P_{KN}(\textit{building}) \big) \Big) \\
=&\frac{\max\{c(\textit{is the 3rd-tallest building}) - D, 0\}}{c(\textit{is the 3rd-tallest})} \\
&+ \frac{D}{c(\textit{is the 3rd-tallest})} N_{1+}(\textit{is the 3rd-tallest}\bullet) \Big(\\
&\frac{\max\{c(\textit{the 3rd-tallest building}) - D, 0\}}{c(\textit{the 3rd-tallest})} + \frac{D}{c(\textit{the 3rd-tallest})} N_{1+}(\textit{the 3rd-tallest}\bullet) \big(\\
&\frac{\max\{c(\textit{3rd-tallest building}) - D, 0\}}{c(\textit{3rd-tallest})} \\
&+ \frac{D}{c(\textit{3rd-tallest})} N_{1+}(\textit{3rd-tallest}\bullet) \frac{N_{1+}(\bullet \textit{building})}{N_{1+}(\bullet\bullet)} \big) \Big) \\
=&\frac{\max\{0 - \frac{25}{25+2*3}, 0\}}{0} + \frac{\frac{25}{25+2*2}}{0} * 0 * (\dots) = 0
\end{aligned}
$$

Danksagung

Zunächst möchte ich mich bei René Pickhardt bedanken, der mich nicht nur während der Bachelorarbeit als Betreuer mit wertvollen Ratschlägen unterstützt, sondern auch schon im Vorfeld in einem anderen Projekt an das Themengebiet herangeführt hat.
Des Weiteren danke ich Dr. Thomas Gottron für seinen fachliche Meinung und die Vorlage für die Abbildungen des Pascalschen Dreiecks.
Vielen Dank auch an Jessica Krüger, Hannah Braun, Melanie Koloseike und Michael Ruster für die Bereitschaft, diese Arbeit Korrektur zu lesen.