UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

WeST People and Knowledge Networks
Institute for Web Science and Technologies

haben wir den titel so schon festgelegt oder ist das nur ein arbeitstitel?

COMPUTING LANGUAGE MODEL ARG-MAX-QUERIES USING TOP-*K* JOINING TECHNIQUES

BACHELORARBEIT

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

vorgelegt von

LUKAS SCHMELZEISEN
lukas@uni-koblenz.de

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

Zweitgutachter: René Pickhardt
Institute for Web Science and Technologies

Koblenz, im April 2015

ABSTRACT

Write Abstract.

ZUSAMMENFASSUNG

Übersetzte Abtract auf Deutsch.

License!

Discussion with René:

- Is “we” ok, when single author? ok
- Are chapters ok? ggf. related work raus streichen. lass brainstormen
- Capitalize in document reference? ?
- Citation style? doesn't matter
- Conditions after Equations ok as in Chapter 3?
- Scope ok? Not too long or short? seems ok. eher zu umfangreich als zu kurz
- Terminology: Probability event? Interpolation weight/argument?

Use standalone documents for figures.

Use etoolbox for ifthenelse in class.

Kill warnings.

Rewrite math package. Use left/right braces for set.

Use subequations more often?

CONTENTS

1 INTRODUCTION

Introduction pargraph: We solve noisy channel argmax with Top-k joining techniques.

The *noisy channel model* is a commonly applied concept in natural language processing. As first described by Shannon (1948), it consists of a transmitter that tries to pass a message over a channel to a receiver. The channel is called noisy, because it may distort the message in any way. The task is then to reconstruct the original message, from the recieved, distorted one.

This metaphor has many applications, such as spelling correction (Jurafsky and Martin 2009; Kernighan et al. 1990; Manning et al. 2008; Mays et al. 1991), part-of-speech-tagging (Church 1988), machine translation (Brown et al. 1990), speech recognition or text compression.

Find newer citations. Maybe from Bickel et al. (2005)

Formalized, this task of the noisy channel model is to find the dispatched message m, which in our context is the sequence of words $s = w_1^n = w_1 \dots w_n$ in a langauge $\mathcal{L}$ that is most likely to be the source of the received, distorted observation o:

$$m = \arg\max_{s \in \mathcal{L}} P(s \mid o) \tag{1.1}$$

mir ist nicht klar, in welchem Wahrscheinlichkeitsraum du bist. das geht immer unter bei dem NCT. ich koennte mir vorstellen, dass wir es fuer die BT auch unter den tisch fallen lassen.

Using Bayes' theorem one can rewrite this to:

$$m = \arg\max_{s \in \mathcal{L}} \frac{P(o \mid s) \cdot P(s)}{P(o)} = \arg\max_{s \in \mathcal{L}} P(o \mid s) \cdot P(s) \tag{1.2}$$

With the observation likelihood $P(o \mid s)$ and the prior probability $P(s)$. Omitting the denominator $P(o)$ is valid because it is a constant that does not depend on the argument s. Most commonly the observation likelihood is some domain specific probability distribution $P_{\text{Domain}}(o \mid s)$; for the prior probability *statistical language models* are used. They assign a probability $P_{\text{LM}}(s)$ to a sequence of words $s = w_1^n = w_1 \dots w_n$ by means of a probability distribution, created through statistical analysis of text corpora. Thus the noisy channel model is a natural combintion of domain knowledge with language information.

For example, to translate French into English, Brown et al. (1990) defined a probability distribution $P_{\text{Translate}}(e \mid f)$, that would give the probability for each English sentence e to be the result of translating the French sentence f into English. To obtain the most likely translation, the e that maximizes that probability has to be found. With applying Bayes' theorem they combine their distribution with language models and arrive at $\arg\max_e P_{\text{Translate}}(f \mid e) \cdot P_{\text{LM}}(e)$, which is just Equation (1.2).

Lukas: It is possible to omit this paragraph, but I think it provides clarification.

A major problem in creating statistical language models from text corpora, is that even for very large corpora, most possible word sequences in a language will not be observed in training [Citation needed].

das nennt man markov assumption

A common approximation are n-*gram models*, in which it is assumed that the probability of a word only depdens on the $n-1$ previous words, thus only sequences of length n have to be considered.

State-of-the-art language models use n-grams with n as large as 5 [Citation: JurafskyMartin2009,Goodman2001?]. Equation (1.1) is hard to calculate for such large n. So we want to optimize that.

We try to use top-k joining.

We are interested in *next word prediction* NWP and *next keystroke savings* NKSS.

So we have to map our problem on the realm of top-k joining.

We apply that technique.

May be good because we do not have to prune, and pruning might be bad (Chelba et al. 2010, 2013; Siivola et al. 2007; Stolcke 2000).

Most non-trivial top-k joining algorithms require a monotone scoring function, some further improvements can be made if using a linear scoring function (Ilyas et al. 2008).

Read Bickel et al. (2005) for info.

Introduction to next word predition.

Overview of following chapters.

bis hier hin ist das klar und verstaendlich aber als einleitung fuer deine bachlor thesis noch unvollstaendig (bzw. die grauen sachen sind etwas kurz gehalten) wenn man deine sachen ausfuehrlicher macht koennte es sein dass die kontext infos ins grundlagen chapter muessten

2

RELATED WORK

Commonly to solve the described problem Viterbi or beam search algorithms are used [Citation: JurafskyMartin2009?].

Is there some work on expressing language models as weighted sums?

jelinek mercer die interpolieren machen das doch oder?

2.1 STATE-OF-THE-ART LANGUAGE MODELS

This section summarizes the state-of-the-art language models considered in this work. And their origins?

The currently most commonly used (Chelba et al. 2013; Jurafsky and Martin 2009) technique for estimating language models is *Modified Kneser-Ney Smoothing* by Chen and Goodman (1996, 1998, 1999), based on *Kneser-Ney Smoothing* by Kneser and Ney (1995). Very recently Pickhardt et al. (2014) provided a generalization of this technique, the *Generalized Language Model*.

Justify selection of MKN and GLM over recently popular LMs from (Chelba et al. 2013).

Mention *Nerual Network Language Models* (Bengio et al. 2003; Mikolov 2012) (do we not use them because they can not be represented as weighted sums).

An in-depth summary of Modified Kneser-Ney Smoothing and the Generalized Language Model is given in Chapter 3.

2.2 TOP-k JOINING TECHNIQUES

Lukas: I would much rather summarize top-k joining techniques in Chapter 5 as the task is clearly defined there, and we can have more in-depth discussion about it.

Reference to Chapter 5.

dann streich doch das related work chapter. es ist ok an den stellen wo man was macht related work aufzubrechen, der vorverweis auf section 3 war auch schon einbissel komisch (wenn man das nur einmal hat ist der trick immer related work umzusortieren. aber in dem fall wuerde ich es wie gesagt als eigenstaendiges kapitel streichen)

3

REVIEW OF CONSIDERED LANGUAGE MODELS

This chapter will recall language modeling techniques considered in this work, and introduce the used notation (Section 3.1). Namely, *Modified Kneser-Ney Smoothing* (Section 3.2) and the *Generalized Language Model* (Section 3.3) are described.

in a section like this you would have a sentence like: we follow [...] and review basics. for unpublished stuff that is work in progress I think marking it and having a statementin the forword that states that some of the stuff in here is new and not your IP seams reasonable

How do I declare citations for this section?

- Do I cite Rene or ChenGoodman?
- Do I cite once or on every definition?
- Some stuff is not published anywhere else yet. (For example new skip notation or new glm notation)

3.1 NOTATION: COUNTS AND SKIPS

Let $c(w_1^n)$ denote the frequency count of a sequence's $w_1^n = w_1 \dots w_n$ occurence in training data.

Let $\square$ be a wildcard (called *"skip"*), that can take the place of any word at its location. Thus, for example $c(\square w_1^n)$ is the frequency count of sequences in training data that start with any word $x \in \Sigma$ and where the remaining words are w_1^n:

$$c(\square w_1^n) = \sum_{x \in \Sigma} c(x\, w_1^n) \tag{3.1}$$

Where Σ is the set of all words in a language $\mathcal{L} \subseteq \Sigma^*$. It is easy to extend this definition to sequences with multiple skips:

$$c(w_1 \square \square w_2) = \sum_{x,y \in \Sigma} c(w_1\, x\, y\, w_2) \tag{3.2}$$

fuer das was du machst brauchst du das wohl nicht.

Do we mention the difference of $c(\square w_1^n)$ to $c(w_1^n)$ in absence of Beginning of Sentence tags?

Additionally we define *continuation* counts $N_k(\cdot)$ as the number of sequences from the training data that occur exactly k times and can be constructed by replacing $\bullet$ (called *"continuation skip"*) with any word $x \in \Sigma$. For example $N_2(\bullet w_1^n)$ denotes the number of words that can precede w_1^n in training data, and occur exactly 2 times:

$$N_2(\bullet w_1^n) = |\{x \in \Sigma \,|\, c(x\, w_1^n) = 2\}| \tag{3.3}$$

Further let $N_{k+}(\cdot)$ count sequences that occur k or more times:

$$N_{1+}(\bullet w_1^n) = |\{x \in \Sigma \,|\, c(x\, w_1^n) \geqslant 1\}| \tag{3.4}$$

k+ und 1+ intermixed. konsistent bleiben

In continuation counts regular skips and continuation skips can be mixed, for example:

Check this equation!

hier wuerde ggf. ein beispiel helfen? 3 bis 4 sequenzen und mal cont skips und absolut counts zaehlen. ist beim simplen lesen vermutlich nicht 100% klar

3.2 Modified Kneser-Ney Smoothing

Modified Kneser-Ney smoothed language models are computed by interpolating between higher and lower order n-gram language models. The highest order distribution is interpolated with lower order distirbutions as follows:

(Copied from Rene)

$$P_{MKN}(w_n \mid w_1^{n-1}) = \frac{c^d(w_1^n) + \gamma(w_1^{n-1})\,\hat{P}_{MKN}(w_n \mid w_1^{n-2})}{c(w_1^{n-1}\,\square)} \quad (3.6)$$

With the *discounted* count $c^d(w_1^n) = \max\{c(w_1^n) - D(c(w_1^n)), 0\}$.

The exact definitions of $D(\cdot)$ and $\gamma(w_1^{n-1})$ are not important in our context, however for the sake of completeness they are given in Section 3.4.

Naturally Equation (3.6) is only defined if the denominator $c(w_1^{n-1}\,\square) > 0$, that is if the history $w_1^{n-1}\,\square$ was seen in training. If $w_1^{n-1}\,\square$ was not seen, we perform a so called *"back-off"* and set:

w_2^{n-1} beimbackoff

$$P_{MKN}(w_n \mid w_1^{n-1}) = P_{MKN}(w_n \mid w_1^{n-2}) \quad (\text{if } c(w_1^{n-1}\,\square) = 0) \quad (3.7)$$

It is common to perform multiple back-offs until the history was actually seen.

This should actually not be true, because we increase n-gram length for first lower order. How do we handle this in current implementation?

Essentially, interpolation with a lower order model corresponds to leaving out the first word in the considered sequence. Lower order models are computed differently, and in turn interpolated with further lower orders:

let us discuss this

$$\hat{P}_{MKN}(w_n \mid w_1^{n-1}) = \frac{N_{1+}^d(\bullet\, w_1^n) + \gamma(w_1^{n-1})\,\hat{P}_{MKN}(w_n \mid w_1^{n-2})}{N_{1+}(\bullet\, w_1^{n-1}\,\bullet)} \quad (3.8)$$

With the *discounted* continuation count $N_{1+}^d(\bullet\, w_1^n) = \max\{N_{1+}(\bullet\, w_1^n) - D(c(w_1^n)), 0\}$. Backing-off to compute lower orders is never necessary, because if history w_1^{n-1} was seen, w_1^{n-2} was seen as well.

The lowest order (the probability of a single word) can not be further interpolated and we use the relative continuation frequency of that word. Similarly if we only want to compute the probability of a single word on the highest order, we compute it as the relative frequency of that word:

$$P_{MKN}(w) = \frac{c(w)}{c(\,\square\,)} \quad (3.9a)$$

$$\hat{P}_{MKN}(w) = \frac{N_{1+}(\bullet\, w)}{N_{1+}(\bullet\,\bullet)} \quad (3.9b)$$

3.3 Generalized Language Model

The Generalized Language Model is a natural extension to Modified Kneser-Ney Smoothing. It is again based on interpolation, though it

differs in the way lower order models are interpolated. The highest order is computed as:

bin nicht sicher ob ich #\partial richtig verstehen (also die semantik)

$$P_{GLM}(w_n \mid w_1^{n-1}) = \frac{c^d(w_1^n) + \frac{\gamma(w_1^{n-1})}{\#_\partial(w_1^{n-1})} \sum_{j=1}^{\#_\partial(w_1^{n-1})} \hat{P}_{GLM}(w_n \mid \partial_j w_1^{n-1})}{c(w_1^{n-1}\,\square)} \tag{3.10}$$

Auxiliary definitions c^d, N_{1+}^d, γ are the same as for Modified Kneser-Ney Smoothing and given in Section 3.2.

The skip operator $\partial_j w_1^{n-1}$ can be any mapping that relates w_1^{n-1} to exactly $\#_\partial(w_1^{n-1})$ many derived sequences. It is easy to see that for $\#_\partial(w_1^{n-1}) = 1$ and $\partial_1 w_1^{n-1} = w_2^{n-1}$ Equation (3.10) is an instantiation of Equation (3.6) (Modified Kneser-Ney Smoothing), and thus a true generalization.

In practice only one definition for ∂ is used though. That is $\#_\partial(w_1^{n-1})$ is set to the number of non-skip words in w_1^{n-1} and $\partial_j(w_1^{n-1})$ replaces the jth non-skip word with a skip. For instance: $\partial_2(w_1^3) = w_1 \square w_3$.

Again as with Modified Kneser-Ney Smoothing, Equation (3.10) is not defined if w_1^{n-1} is unseen and thus our denominator would be zero. The back-off step is then performed as:

$$P_{GLM}(w_n \mid w_1^{n-1}) = \frac{1}{\#_\partial(w_1^{n-1})} \sum_{j=1}^{\#_\partial(w_1^{n-1})} P_{GLM}(w_n \mid \partial_j w_1^{n-1}) \quad (\text{if } c(w_1^{n-1}\,\square) = 0) \tag{3.11}$$

Subsequently lower order models are computed as:

$$\hat{P}_{GLM}(w_n \mid w_1^{n-1}) = \frac{N_{1+}^d(\bullet\, w_1^n) + \frac{\gamma(w_1^{n-1})}{\#_\partial(w_1^{n-1})} \sum_{j=1}^{\#_\partial(w_1^{n-1})} \hat{P}_{GLM}(w_n \mid \partial_j w_1^{n-1})}{N_{1+}(\bullet\, w_1^{n-1}\,\bullet)} \tag{3.12}$$

Replace $\square$ with $\bullet$ in N_{1+} in GLM?

$\bullet$ at end of N_{1+} correct?

The case of the lowest order occurs when $\#_\partial(w_1^{n-1}) = 0$ and is handled similarly to Modified Kneser-Ney Smoothing. In practice this is the case when w_1^{n-1} only consists of skips.

$$P_{GLM}(w_n \mid w_1^{n-1}) = \frac{c(w_1^n)}{c(w_1^{n-1}\,\square)} \qquad (\text{if } \#_\partial(w_1^{n-1}) = 0) \tag{3.13a}$$

$$\hat{P}_{MKN}(w_n \mid w_1^{n-1}) = \frac{N_{1+}(\bullet\, w_1^n)}{N_{1+}(\bullet\, w_1^{n-1}\,\bullet)} \qquad (\text{if } \#_\partial(w_1^{n-1}) = 0) \tag{3.13b}$$

Explain Discounting and Smoothing.

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c \geqslant 3 \end{cases} \tag{3.14}$$

$$D_1 = 1 - 2Y\frac{n_2}{n_1} \tag{3.15a}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2} \tag{3.15b}$$

$$D_{3+} = 3 - 4Y\frac{n_4}{n_3} \tag{3.15c}$$

$$Y = \frac{n_1}{n_1 + n_2} \tag{3.16}$$

$$\gamma(w_1^{n-1}) = D_1\, N_1(w_1^{n-1}\bullet) + D_2\, N_2(w_1^{n-1}\bullet) + D_{3+}\, N_{3+}(w_1^{n-1}\bullet) \tag{3.17}$$

4

FORMULATING LANGUAGE MODELS AS WEIGHTED SUMS

This chapter will show how to represent $P_{MKN}(w_n \mid w_1^{n-1})$ and $P_{GLM}(w_n \mid w_1^{n-1})$ as weighted sums of terms depending on w_n for a fixed history $h = w_1^{n-1}$ and an arbitrary argument $w = w_n$. In other words we will express the formulas given in Chapter 3 as equations of the following form:

$$P(w \mid h) = \sum_{i=1}^{N} \lambda_i^h \cdot \alpha_i^h(w_n) \tag{4.1}$$

ich bin nicht sicher, ob du das wirklich so beschreiben kannst bzw. willst. du willst ja eigentlich noch die info geben dass auch die alphas recht leicht in absteigender order vorberechnet werden koennen. ist es sinnvoll das allgemeine problem hier zwischen speicherplatz und indizierung und laufzeit an unserem beispiel durchzusprechen?

Using this it is possible to calculate queries of type "Noisy Channel Argmax" (Equation (1.1)) much more efficiently, as it is possible to perform the (potentially expensive) calculation of λ_i^h in advance. To compute every $P(w \mid h)$ we now just have to calculate the $\alpha_i^h(w)$.

Another benefit of this representation is that it is a prerequisite of Top-k joining to supply a monotone scoring function, which only depends on the joined arguments. Weighted sums are trivially monotone and we will be able to select our join arguments to exactly match α_i^h. In addition further optimized top-k joining algorithms exists, that require weighted sum scoring functions. However we will not use these algorithms in this work. For a detailed discussion see Chapter 5.

really?

naive vielleicht durch rekursive ersetzen?

In the case of Generalized Language Models our solution will also have a considerably better computational complexity to calculate probabilities than the naïve approach of just computing the formulas given in Section 3.3. This will counter biggest disadvantage of GLM.

We will now present the basic idea of how to transform $P(w_n \mid w_1^{n-1})$ into weighted sums, that applies to both Modified Kneser-Ney Smoothing and the Generalized Language Model as well:

Note that in all Equations (3.6), (3.8), (3.10) and (3.12) the probability event w_n only appears in the argument terms of c^d or N_{1+}^d. Additionally w_n occurs as arguments for c and N_{1+} in the lowest order Equations (3.9) and (3.13).

Our general idea is thus, to first expand recursive calls to $P(\cdot)$ or $\hat{P}(\cdot)$, and second to factor out those terms that do not depend on w_n by the distributive property. Examples of this idea are given in Appendix A. However they assume the history to be seen in training. Forfeiting this assumption will add considerable complexity in the case of Generalized Language Models.

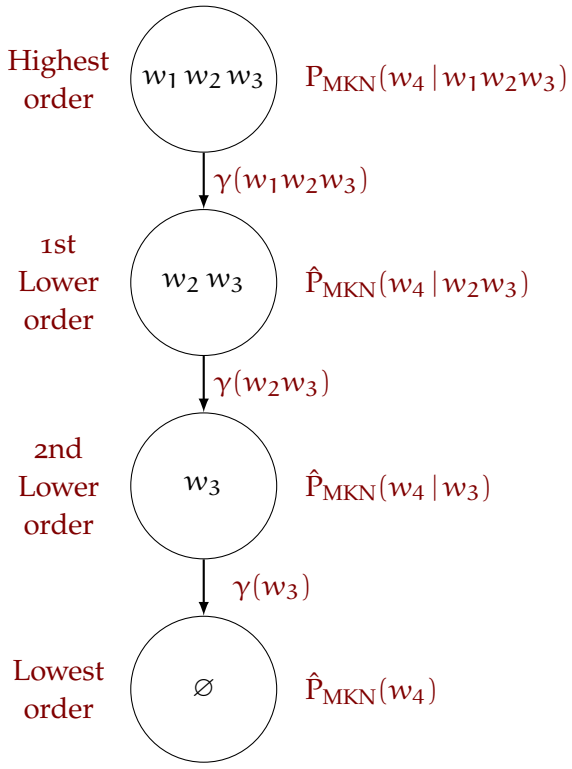


Figure 4.1: Interpolation of different orders during computation of $P_{MKN}(w_4 \mid w_1 w_2 w_3)$. Centered are the histories for each probability. It is assumed that the history $w_1 w_2 w_3$ was seen during training, so that no backing-off is necessary. Orders are combined with with interpolation weights $\gamma(\cdot)$.

4.1 MODIFIED KNESER-NEY SMOOTHING

Modified Kneser-Ney Smoothed probabilities $P_{MKN}(w \mid h)$ are calculated as follows:

1. Backing-off: Leave out words at the beginning of the history h, until $h\,\square$ is seen for the first time (Equation (3.7)).
2. Highest order: Take frequency counts $c^d(h\,w)$ and $c(h\,\square)$ (Equation (3.6)) and shorten the history by one word.
3. Lower orders: Take continuation counts $N^d_{1+}(\bullet\, h\, w)$ and $N_{1+}(\bullet\, h\, \bullet)$ (Equation (3.8)) and shorten the history further until it is empty.
4. Lowest order: Take continuation counts $N_{1+}(\bullet\, w)$ and $N_{1+}(\bullet\, \bullet)$ (Equation (3.9)).

All counts except those of the lowest order are discounted using c^d and N^d_{1+} to interpolate that order with the following by the weight $\gamma(h)$. This process of interpolation different orders is visualized in Figure 4.1.

The first step in expressing $P_{MKN}(w \mid h)$ as a weighted sum is finding the number N of sum weights. In the case of Modified Kneser-Ney Smoothing this is exactly the number of interpolation orders. Let h' be the first seen history that may be the result of backing off given

history h a number of times. The number N of sum weights is then the number of words in the first seen history h' plus one.

$$N = |h'| + 1 \tag{4.2}$$

The next step is to find the actual terms $\alpha_i^h(w)$ that shall be weighted added. When looking at the equations that define Modified Kneser-Ney we note the fact that only the count terms in the numerators depend on the probability event w. Exactly these terms compose our sum arguments:

$$\alpha_i^h(w) = \begin{cases} c(w) & \text{if } N = 1 \\ c^d(h'\,w) & \text{if } N \neq 1 \wedge i = 1 \\ N_{i+}^d(\bullet\,\eta_i(h')\,w) & \text{if } N \neq 1 \wedge 1 < i < N \\ N_{i+}(\bullet\,w) & \text{if } N \neq 1 \wedge i = N \end{cases} \tag{4.3}$$

Where $\eta_i(w_1^n) = w_i^n$ is a helper mapping that gives the history of the ith interpolation order.

Lastly we need to define the actual sum weights λ_i^h. We note that — because each order is interpolated with the weight γ of the previous order — each order is in total weighted by the product of the γs of all previous orders. Additionally lower order models occur in the numerators of fractions, because of that they are weighted by all previous denominator counts as well. Building on this we can define:

$$\lambda_i^h = \frac{\prod_{j=1}^{i-1} \gamma(\eta_j(h'))}{c(h'\,\square)\prod_{k=2}^{i} N_{i+}(\bullet\,\eta_k(h')\,\bullet)} \tag{4.4}$$

Or better recursively? Or alternatively?

$$\lambda_i^h = \frac{1}{c(h'\,\square)}\prod_{j=2}^{i}\frac{\gamma(\eta_{j-1}(h'))}{N_{i+}(\bullet\,\eta_j(h')\,\bullet)} \tag{4.5}$$

Do we prove that this is actually MKN?

Specifying an algorithm to compute interpolation weights λ_i^h that minimizes frequency count lookup (using the iterative definition of λ_i^h) is straightforward and given in Algorithm 4.1.

Do we actually show this algorithm, as it is rather simple?

4.2 GENERALIZED LANGUAGE MODEL

The difference between a Modified Kneser-Ney language model and a Generalized Language Model is the way in which orders are interpolated. Instead of just one probability — that of shortening the history by one — being factored in, in the Generalized Language Model the average of multiple probabilities of the next lower order is factored into the higher order probability.

This section will not consider the general case were one probability $P_{GLM}(w\,|\,h)$ incorporates exactly $\#_\partial(h)$ probabilities $\hat{P}_{GLM}(w\,|\,\partial_j h)$ of

ich verstehe wie man zu formel 4.4. kommt frage mich aber ob es mir hilft, dass ich die theorie gut verstehe bzw. mit dir zusammen die loesung entwickelt habe. ich bin auch ein fan von abstraktion aber vielleicht wuerde ein beispiel (nicht nur um anhang, inden ichnoch nicht geschaut habe) helfen?

sollte sich per induktion recht leicht beweisen lassen. waere super

was heisst denn simple in unserem fall?

Algorithm 4.1 Computing Modified Kneser-Ney sum weights

Input: h ▷ History for which to determinate interpolation weights
Output: λ_i^h ▷ List of interpolation weights

h'.removeFirstWord() als vorschlag?

$h' \leftarrow h$ ▷ Find first seen history h'
while $c(h'\,\square\,) = 0$ **do**
    $h' \leftarrow h'$.backoff()

Better notation for backoff

$N = |h'| + 1$
$\lambda^h \leftarrow$ new List of size N
$\lambda_1^h \leftarrow \frac{1}{c(h'\,\square\,)}$ ▷ Set highest order weight

hier haste rekursive definition im vgl. zur formel 4.5 iritierend

for $i \leftarrow 2, \ldots, N$ **do**
    $\frac{\gamma(\eta_{i-1}(h'))}{N_{1+}(\bullet\,\eta_i(h')\,\bullet\,)}$ ▷ Compute lower orders using

on with x^{-1} for better visual?

the next lower order. Instead only the special case described by Pickhardt et al. (2014), that is actually used in practice, is examined. That is the number of lower order probabilities incorporated $\#_\partial(h)$ is set to the number of non-skip words in h, and $\partial_j(h)$ is defined as replacing the jth non-skip word in h with a skip $\square$.

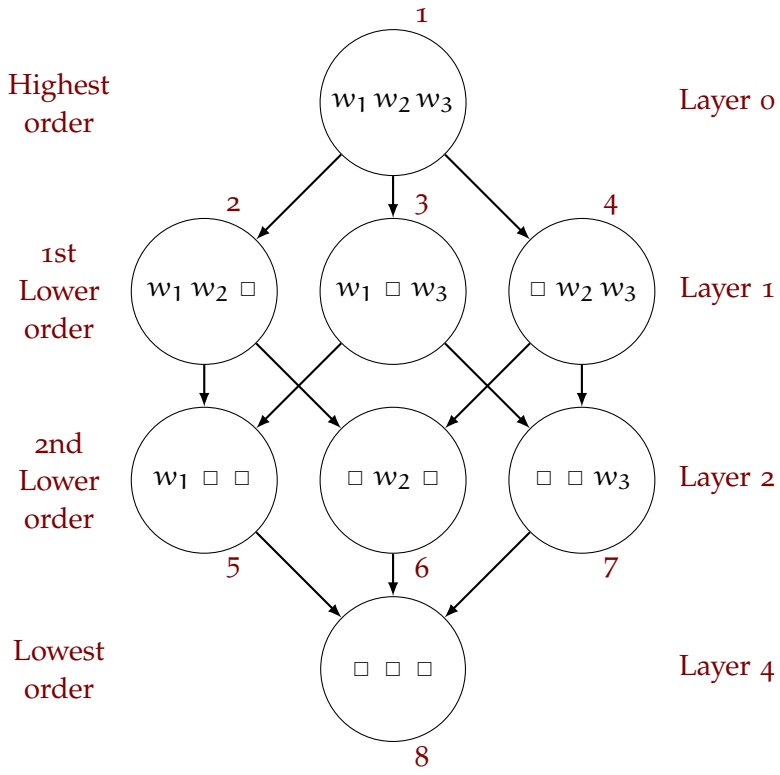
4.2.1 *Binomial Diamond*

ich kannte den term hasse diagramm nicht habe es gerade auf wikipedia nachgelesen. entweder diamant oder hasse diagramm, nicht aber beides. lass pro und cons diskutieren. ich vermute diamant ist vorteilhafter.

grammar

1. If the is partitioned graph into layers, where layer k contains the histories that have exactly k skip words, the graph has $n+1$ layers.
2. Layer k contains exactly $\binom{n}{k}$ nodes.
3. Nodes in layer k have exactly $n-k$ outgoing and k incoming edges.
4. There are 2^n nodes in the graph.

Item 1 is due to the fact that layer k includes exactly those histories that are part of the kth highest order of probability.



sprung von layer 2 zu layer 4 ?

Figure 4.2: Binomial diamond of order 3 for history $h = w_1\ w_2\ w_3$.

imho nicht noetig, aber wenn du zeit hast: why not

To explain Item 2, it is helpful to imagine replacing words in a history as laying a mask of either a skip or a non-skip on top of each word. Then the number of nodes in each layer can be understood as permuting the skip and non-skip masks on top of the history. It is a well known fact that there are exactly $\binom{n}{k}$ permutations of k tokens of type A (skips) and l tokens of type B (non-skips) when $n = k + l$.

Add small figure to visualize this idea?

Item 3 directly follows from the fact, that each non-skip word in a history can be replaced by a skip, and that each skip is the result of those replacements.

With this argumentation $\binom{n}{l}$ would be true as well.

Lastly, we can explain Item 4 because every history is just the result of laying a mask that is a combination of skips or non-skips on top of the original history. All possible masks are then all permutations of two elements (skips or non-skips) with repetition of length n, which there are 2^n arrangements of. We can also infer this from Item 2 via the binomial theorem $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$, if we set $x = y = 1$, we obtain $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.

yo das ist doch symmetrie von binominal koeffizienten

Following the model of the binomial diamond will be a major aid in understanding the weighted sum representation of the Generalized Language Model.

4.2.2 *Number of sum weights*

In the previous subsection, it was reasoned that, in the Generalized Language Model, for a history of length n there are exactly 2^n possible derivative histories. Following this, one might make the assumption, that to evaluate $P_{GLM}(w|h)$ it is necessary to evaluate at most

$2^n - 1$ different probabilities $\hat{P}_{GLM}(w \mid \partial^* h)$ where $\partial^* h$ is any of the $2^n - 1$ derived histories of h excluding h itself. This assumption how-

solche ankuendigungen sind schwierig fuer den lesen. entweder diskutiere es wenn es kommt oder hier aber nicht auf 2 stellen verteilen

Let us for example calculate the probability $P_{GLM}(c \mid a\ b)$ on hypothetical training data. Were the history ab not to occur in that training corpus, we would have to substitute the average of back-off probabilities following Equation (3.11):

$$P_{GLM}(c \mid a\ b) = \frac{1}{2}\left(P_{GLM}(c \mid \square\ b) + P_{GLM}(c \mid a\ \square)\right) \tag{4.6a}$$

Note that these back-off probabilities for histories $\square\ b$ and $a\ \square$ are still of the highest order P_{GLM}. We now further assume history $\square\ b$ to also not occur in training, so that further back-off is necessary. History $a\ \square$ however is treated as seen:

$$P_{GLM}(c \mid \square\ b) = P_{GLM}(c \mid \square\ \square) \tag{4.6b}$$

$$P_{GLM}(c \mid a\ \square) = \frac{c(a\ \square\ c) + \gamma(a\ \square)\,\hat{P}_{GLM}(c \mid \square\ \square)}{c(a\ \square\ \square)} \tag{4.6c}$$

We can see, that both $P_{GLM}(c \mid \square\ \square)$ (Equation (4.6b)) and $\hat{P}_{GLM}(c \mid \square\ \square)$ (Equation (4.6c)) factor into probability $P_{GLM}(c \mid ab)$. They would be evaluated as:

$$P_{GLM}(c \mid \square\ \square) = \frac{c(\square\ \square\ c)}{c(\square\ \square\ \square)} \tag{4.6d}$$

$$\hat{P}_{GLM}(c \mid \square\ \square) = \frac{N_{1+}(\bullet\ \square\ \square\ c)}{N_{1+}(\bullet\ \square\ \square\ \bullet)} \tag{4.6e}$$

Because of this, it is possible that one history h^* might both occur as highest order probability $P_{GLM}(w \mid h^*)$ and lower order probability $\hat{P}_{GLM}(w \mid h^*)$. It is non-trivial to decide exactly how many derived histories occur both as highest and lower order probabilities, and to fit algorithms to this finding. Consequently we only employ an upper bound of sum weights N and set all unnecessary sum weights to zero. An upper bound for the number of sum weights is the number of derived histories times two, as histories can occur both in highest and lower order probabilities:

$$N = 2 \cdot 2^{|h|} \tag{4.7}$$

4.2.3 *Sum factors*

warum gibt es im fall von =0 keinen discount? das ist ja nicht der unigram fall oder doch?

$$\begin{cases} c^d(\ldots) \ldots \text{if } \#(\ldots) > 0 \\ \ldots = 0 \end{cases} \tag{4.8a}$$

$$\begin{cases} \ldots(h)) > 0 \\ \ldots(h)) = 0 \end{cases} \tag{4.8b}$$

Or alternatively?

$$\alpha^{h}_{2i+j}(w) = \begin{cases} c(\eta_i(h)w) & \text{if } j = 0 \wedge \#_\partial(\eta_i(h)) = 0 \\ N_{1+}(\bullet\eta_i(h)w) & \text{if } j = 1 \wedge \#_\partial(\eta_i(h)) = 0 \\ c^d(\eta_i(h)w) & \text{if } j = 0 \wedge \#_\partial(\eta_i(h)) > 0 \\ N^d_{1+}(\bullet\eta_i(h)w) & \text{if } j = 1 \wedge \#_\partial(\eta_i(h)) > 0 \end{cases} \tag{4.9}$$

Or alternatively?

$$\alpha^{h}_{2i+j}(w) = \begin{cases} c(\eta_i(h)w) & \text{if } j = 0 \wedge N - i \leqslant 1 \\ N_{1+}(\bullet\eta_i(h)w) & \text{if } j = 1 \wedge N - i \leqslant 1 \\ c^d(\eta_i(h)w) & \text{if } j = 0 \wedge N - i > 1 \\ N^d_{1+}(\bullet\eta_i(h)w) & \text{if } j = 1 \wedge N - i > 1 \end{cases} \tag{4.10}$$

μ is coefficient.

Formulate μ as direct formula with faculties.

λ^h_{2i} is $\frac{\mu}{c(\eta_i(h)\square)}$ times number of unseen paths to node.
λ^h_{2i+1} is $\frac{\mu}{N_{1+}(\bullet\eta_i(h)\bullet)}$ times the sum of paths to node with each node on path weighted.

1. Weight unseen nodes as zero if they are direct parents of targets and if target is seen.
2. Else weight unseen nodes as one.
3. Weight first seen node as $\frac{\gamma(\eta_i(h^*))}{c(\eta_i(h^*)\square)}$.
4. Weight following nodes as $\frac{\gamma(\eta_i(h^*))}{N_{1+}(\bullet\eta_i(h^*)\bullet)}$.

Lukas: I have no idea for a good notation for this concept of a path in binomial diamond.

4.2.4 *Algorithm*

Counts usually stored in hash maps, so we get them to avoid look up.
Use fact that we can apply distributive property on common roots in paths for one node.

Can we improve algorithm if we just walk each path once?

lass uns das 2f2 diskutieren. ich habe dich hier gerade ein wenig verloren

4.2.5 *Computational Complexity*

Find complexity of naive approach.

Find complexity of my algorithm. Need to find complexity of finding paths for this.

finding paths ist be breitensuche bzw. tiefensuche. das sollte nicht so wild sein. ich denke was dem text i.a. fehlt ist zu unterscheiden zwischen trainingstime und query time. der ganze ansatz den du hier ja faehrst ist query time zu optimieren. das steckt zwar implizit im argmax und so

Algorithm 4.2 Computing Generalized Language Model sum weights

Input: h ▷ History for which to determinate interpolation weights
Output: λ_i^h ▷ List of interpolation weights

$N \leftarrow 2 \cdot 2^{|h|}$
for $i \leftarrow 1 .. \frac{N}{2}$ **do**
  $C_i^a \leftarrow c(\eta_i(h)\,\square\,)$
  $C_i^c \leftarrow N_{1+}(\bullet\,\eta_i(h)\,\bullet\,)$
  $\Gamma_i \leftarrow \gamma(\eta_i(h))$
  $\mu \leftarrow$ calcCoefficient($level(i)$, $|h|$)
  if $C_i^a = 0$ **then**
    $\lambda_{2i}^h \leftarrow 0$
  else if $i = 0$ **then**
    $\lambda_{2i}^h \leftarrow \frac{\mu}{C_i^a}$
  else
    $\lambda_{2i}^h \leftarrow \frac{\mu}{C_i^a}$ calcAbsoluteWeight(0, i)
  if $C_i^c = 0$ **then**
    $\lambda_{2i+1}^h \leftarrow 0$
  else
    $\lambda_{2i+1}^h \leftarrow \frac{\mu}{C_i^c}$ calcContinuationWeight(0, i, **true**)

function calcCoefficient($level$, $order$)
  $\mu = 1$
  for $i \leftarrow 1 .. level$ **do**
    $\mu \leftarrow \mu \cdot (order - i)$
  return μ

die funktion hast du doch hoffentlich gecached oder?

function calcAbsoluteWeight($ancestor$, $node$)
  if $C^a_{absolute} \neq 0$ **then**
    return 0
  if $level(ancestor) = level(node) - 1$ **then**
    return 1
  $\# \leftarrow 0$
  for $i \leftarrow 1 .. numChilds(ancestor)$ **do**
    $child \leftarrow child(ancestor, i)$
    if $isAncestorOf(child, node)$ **then**
      $\# \leftarrow \# +$ calcAbsoluteWeight($child$, $node$)
  return #

```
1: function calcContinuationWeight(ancestor, node, absolute)
2:     if absolute then
3:         if C^a_ancestor = 0 then
4:             if level(ancestor) = level(node) − 1 ∧ C^a_node <> 0
   then
5:                 mult ← 0
6:
   Quick exit with return 0?
7:             else
8:                 mult ← 1
9:         else
10:            mult ← Γ_ancestor / C^a_ancestor
11:            absolute ← false
12:    else if C^c_ancestor = 0 then
13:        mult ← 1
14:    else
15:        mult ← Γ_ancestor / C^c_ancestor
16:    if level(ancestor) = level(node) − 1 then
17:        return mult
18:    sum = 0
19:    for i ← 1..numChilds(ancestor) do
20:        child ← child(ancestor, i
21:        if isAncestorOfchild, node then
22:            sum ← sum + calcContinuationWeight(child,
   node, absolute)
23:    return mult · sum
```

ist legitim und spart i.a. zeit von daher: why not?

4.2.6 *Bitmagic*

Better subsection name.

5

TOP-*K* JOINING LANGUAGE MODELS

This chapter explains how to efficiently compute arg-max-queries of language model P_{LM} using top-k joining techniques.

The task is to efficiently compute the following expression:

$$\arg\max_{t \in T} P_{LM}(t \mid h) \tag{5.1}$$

Where t is a token in the set of all tokens T, P_{LM} is a language model, that can be expressed as a weighted sum (as described in Chapter 4), and h is the conditional history, a sequence of tokens.

Given a language model P_{LM} as a scoring function, if we imagine our input data forming database relations, we can express our arg-max-query as a SQL-`SELECT`-Request, as given in Listing 5.1.

Where `history` and plus is concat.

Explain **`LEFT OUTER JOIN`**.

Our data model is thus: One input relation `Token` that stores all possible tokens *`arg`*, over which we want to compute:

$$\arg\max_{arg \in \texttt{Token}} P_{LM}(arg \mid \texttt{history}) \tag{5.2}$$

Additionally we have n input relations *`Pattern1`* to *`PatternN`* that contain pairs of `sequences` and the sequence's occurrence `count` as columns. Each *`Pattern`* relation contains exactly these sequences that fit the relations pattern.

For an example database state and fully expanded SQL-Request see Appendix B.

A naïve method to solve this query would (1) produce all possible object combinations that satisfy the join condition, (2) order results

Listing 5.1 Example SQL-Request Caption]

Example SQL-Request Caption

```
1:  SELECT
2:    Token.token AS arg,
3:    P_LM(Pattern1.count, ..., PatternN.count) AS score
4:  FROM Token
5:  LEFT OUTER JOIN Pattern1
6:    ON history(Pattern1) + arg = Pattern1.sequence
7:  ...
8:  LEFT OUTER JOIN PatternN
9:    ON history(PatternN) + arg = PatternN.sequence
10: ORDER BY score DESC
11: LIMIT k;
```

by scoring function P_{LM}, and (3) discard non top-k objects (those that do not have the k largest scores).

Considering that in reality only small values of k are queried, this approach can be deemed as rather wasteful, as large numbers of objects would have to be discarded. Optimally we would like to cherry-pick the top-k join results and only score and order those. Relying on the fact that we are dealing with a *monotone* scoring function we approximate this by sorting our input relations by their counts. Ilyas et al. (2008) survey a wide number of techniques to produces top-k results with a monotone scoring function from sorted relations.

Among other things, top-k joining techniques differ on the type of join conditions they can be applied to. One particularly optimizable join operation is the *equi-join*: the join that checks for equality over a unique key attribute present in all relations. Equi-joins have the advantage that it is possible to form equivalence classes over an object's key attribute to determine which join results an object is part of. Using this, the *hash join* algorithm can efficiently generate join results.

It is possible to view our join operations as an instance of equi-join: By filtering each input relation `Pattern` to only contain objects whose sequences begin with the requested history, objects' sequences only differ on the last token (our argument token `arg`). Our join condition thus reduces to `arg = lastToken(Pattern.sequence)`.

Read *"2013 BlanasPatel Memory Footprint Matters Efficient Equi-Join Algorithms fro Main Memory Data Processing.pdf"* for possibly better join algorithm

.

The technique for obtaining these both sorted and filtered relations is explained in Section 5.2.

Do we want discussion here rather than explanation?

The algorithm used to produce top-k results is presented in Section 5.3.

5.1 REMARKS

5.2 SORTED AND FILTERED ACCESS

As explained we require a data structure for efficiently retrieving all sequences whose first to second to last words match a requested string h. Additionally we want to specify that the last word has to contain p as a prefix, in order to support efficient Next-Keystroke-Savings NKSS computation. $\{ w_1^n \mid w_1^{n-1} = h \wedge b \text{ is prefix of } w_n \}$ which is equivalent to $\{ w_1^n \mid \}$

$$\underbrace{w_1 \quad w_2 \quad \dots \quad w_{n-1}}_{\text{matches } h} \quad \underbrace{w_n}_{\text{starts with } b}$$

NKSS requires Top-k Completion.

Use Completion-Trie (Hsu and Ottaviano 2013).

Explain Completion-Trie here?

Can we save all patterns in one trie?

5.3 ALGORITHM

5.3.1 *Requirements*

1. Dataset remains the same.
2. Scoring function changes on each query.
3. Join condition changes on each query!

5.3.2 *Algorithm*

There are several techniques for indexing Rank Join Indices [Citation: Tsaparas et al 2003?], Onion Indices [Citation: Chang et al 2000?]) which we don't use as history changes to often.

Ranked Join Indicies[Citation: Tsaparas et al 2003?])

Algorithms that precompute stuff may be general speed up, but not in our case since we never expect the same thing to be queried again.

TA (Fagin et al. 2001).

Rank-Join (Ilyas et al. 2004).

Evaluate following algorithms:

1. Traditional Rank-Join that generates all possible join combinations after each input.
2. Improved version that uses random access for early tuple completion.

Use observation that if a pattern is seen, it's child patterns' will be seen as well.

Clever: How to find T as max of (p_1^{max}, p_2^{last}) and (p_1^{last}, p_2^{max}).

Clever: Priority-Queue with constant contains and update.

6

EVALUATION

7

CONCLUSION

REFERENCES

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin (2003). "A Neural Probabilistic Language Model." In: *J. Mach. Learn. Res.* 3, pp. 1137–1155 (cit. on p. 3).

Bickel, Steffen, Peter Haider, and Tobias Scheffer (2005). "Predicting Sentences Using N-gram Language Models." In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. HLT '05. Vancouver, British Columbia, Canada: Association for Computational Linguistics, pp. 193–200 (cit. on pp. 1 sq., 39).

Brown, Peter F., John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin (1990). "A Statistical Approach to Machine Translation." In: *Computational Linguistics* 16.2, pp. 79–85 (cit. on p. 1).

Chelba, Ciprian, Thorsten Brants, Will Neveitt, and Peng Xu (2010). "Study on Interaction between Entropy Pruning and Kneser-Ney Smoothing." In: *Proceedings of Interspeech*, pp. 2242–2245 (cit. on p. 2).

Chelba, Ciprian, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson (2013). *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*. Tech. rep. Google (cit. on pp. 2 sq.).

Chen, Stanley F. and Joshua T. Goodman (1996). "An Empirical Study of Smoothing Techniques for Language Modeling." In: *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*. ACL '96. Santa Cruz, California: Association for Computational Linguistics, pp. 310–318 (cit. on p. 3).

– (1998). *An Empirical Study of Smoothing Techniques for Language Modeling*. Tech. rep. Computer Science Group, Harvard University (cit. on p. 3).

– (1999). "An empirical study of smoothing techniques for language modeling." In: *Computer Speech & Language* 13.4, pp. 359–393 (cit. on p. 3).

Church, Kenneth Ward (1988). "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text." In: *Proceedings of the Second Conference on Applied Natural Language Processing*. ANLC '88. Austin, Texas: Association for Computational Linguistics, pp. 136–143 (cit. on p. 1).

Fagin, Ronald, Amnon Lotem, and Moni Naor (2001). "Optimal Aggregation Algorithms for Middleware." In: *Proceedings of the Twenti-*

eth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. PODS '01. Santa Barbara, California, USA: ACM, pp. 102–113 (cit. on p. 21).

Hsu, Bo-June (Paul) and Giuseppe Ottaviano (2013). "Space-efficient Data Structures for Top-k Completion." In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, pp. 583–594 (cit. on p. 20).

Ilyas, Ihab F., Walid G. Aref, and Ahmed K. Elmagarmid (2004). "Supporting Top-k Join Queries in Relational Databases." In: *The VLDB Journal* 13.3, pp. 207–221 (cit. on p. 21).

Ilyas, Ihab F., George Beskales, and Mohamed A. Soliman (2008). "A Survey of Top-k Query Processing Techniques in Relational Database Systems." In: *ACM Comput. Surv.* 40.4, 11:1–11:58 (cit. on pp. 2, 20).

Jurafsky, Daniel and James H. Martin (2009). *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. (cit. on pp. 1, 3).

Kernighan, Mark D., Kenneth Ward Church, and William A. Gale (1990). "A Spelling Correction Program Based on a Noisy Channel Model." In: *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*. COLING '90. Helsinki, Finland: Association for Computational Linguistics, pp. 205–210 (cit. on p. 1).

Kneser, R. and H. Ney (1995). "Improved backing-off for M-gram language modeling." In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1, 181–184 vol.1 (cit. on p. 3).

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press (cit. on p. 1).

Mays, Eric, Fred J. Damerau, and Robert L. Mercer (1991). "Context Based Spelling Correction." In: *Inf. Process. Manage.* 27.5, pp. 517–522 (cit. on p. 1).

Mikolov, Tomáš (2012). "Statistical Language Models Based on Neural Networks." PhD thesis. Ph. D. thesis, Brno University of Technology (cit. on p. 3).

Pickhardt, René, Thomas Gottron, Martin Körner, Paul Georg Wagner, Till Speicher, and Steffen Staab (2014). "A Generalized Language Model as the Combination of Skipped n-grams and Modified Kneser Ney Smoothing." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. abs/1404.3377. Baltimore, Maryland: Association for Computational Linguistics, pp. 1145–1154 (cit. on pp. 3, 12).

Shannon, Claude Elwood (1948). "A mathematical theory of communication." In: *Bell System Technical Journal, The* 27.3, pp. 379–423 (cit. on p. 1).

Siivola, V., T. Hirsimaki, and S. Virpioja (2007). "On Growing and Pruning Kneser ndash;Ney Smoothed N -Gram Models." In: *Audio, Speech, and Language Processing, IEEE Transactions on* 15.5, pp. 1617–1624 (cit. on p. 2).

Stolcke, Andreas (2000). "Entropy-based Pruning of Backoff Language Models." In: *CoRR* cs.CL/0006025 (cit. on p. 2).

A

EXAMPLE LANGUAGE MODEL EXPANSION

Examples for Chapter 4.

A.1 MODIFIED-KNESER-NEY

Sample expansion for $P_{MKN}(c \mid a\ b)$, assuming the sequence "a b c" was seen:

$$P_{MKN}(c \mid a\ b) = \frac{c^d(a\ b\ c) + \gamma(a\ b)\,\hat{P}_{MKN}(c \mid b)}{c(a\ b\ \square)} \tag{A.1}$$

$$\hat{P}_{MKN}(c \mid b) = \frac{N^d_{1+}(\bullet\, b\ c) + \gamma(b)\,\hat{P}_{MKN}(c)}{N_{1+}(\bullet\, b\ \bullet)} \tag{A.2}$$

$$\hat{P}_{MKN}(c) = \frac{N_{1+}(\bullet\, c)}{N_{1+}(\bullet\ \bullet)} \tag{A.3}$$

Together:

$$P_{MKN}(c \mid a\ b) = \frac{c^d(a\ b\ c) + \gamma(a\ b)\frac{N^d_{1+}(\bullet\, b\ c) + \gamma(b)\frac{N_{1+}(\bullet\, c)}{N_{1+}(\bullet\ \bullet)}}{N_{1+}(\bullet\, b\ \bullet)}}{c(a\ b\ \square)} \tag{A.4}$$

By using the distributive property:

$$\begin{aligned} P_{GLM}(c \mid a\ b) = \quad & c^d(a\ b\ c) \cdot \frac{1}{c(a\ b\ \square)} \\ + \quad & N^d_{1+}(\bullet\, b\ c) \cdot \frac{\gamma(a\ b)}{c(a\ b\ \square)\, N_{1+}(\bullet\, b\ \bullet)} \\ + \quad & N_{1+}(\bullet\, c) \cdot \frac{\gamma(a\ b)\gamma(b)}{c(a\ b\ \square)\, N_{1+}(\bullet\, b\ \bullet)\, N_{1+}(\bullet\ \bullet)} \end{aligned} \tag{A.5}$$

A.2 GENERALIZED LANGUAGE MODEL

Sample expansion for $P_{GLM}(c \mid a\ b)$, assuming the sequence "a b c" was seen:

$$P_{GLM}(c \mid a\ b) = \frac{c^d(a\ b\ c) + \frac{\gamma(a\ b)}{2}\left(\hat{P}_{GLM}(c \mid \square b) + \hat{P}_{GLM}(c \mid a\ \square)\right)}{c(a\ b\ \square)} \tag{A.6}$$

$$\hat{P}_{GLM}(c \mid \square b) = \frac{N_{1+}^d(\bullet\ \square b\ c) + \frac{\gamma(\square b)}{1}\hat{P}_{GLM}(c \mid \square\ \square)}{N_{1+}(\bullet\ \square b\ \bullet)} \tag{A.7}$$

$$\hat{P}_{GLM}(c \mid a\ \square) = \frac{N_{1+}^d(\bullet a\ \square c) + \frac{\gamma(a\ \square)}{1}\hat{P}_{GLM}(c \mid \square\ \square)}{N_{1+}(\bullet a\ \square\ \bullet)} \tag{A.8}$$

$$\hat{P}_{GLM}(c \mid \square\ \square) = \frac{N_{1+}(\bullet\ \square\ \square\ c)}{N_{1+}(\bullet\ \square\ \square\ \bullet)} \tag{A.9}$$

Together:

$$P_{GLM}(c \mid a\ b) = \tag{A.10}$$
$$\frac{c^d(a\ b\ c) + \frac{\gamma(a\ b)}{2}\left(\frac{N_{1+}^d(\bullet\ \square b\ c) + \frac{\gamma(\square b)}{1}\frac{N_{1+}(\bullet\ \square\ \square\ c)}{N_{1+}(\bullet\ \square\ \square\ \bullet)}}{N_{1+}(\bullet\ \square b\ \bullet)} + \frac{N_{1+}^d(\bullet a\ \square c) + \frac{\gamma(a\ \square)}{1}\frac{N_{1+}(\bullet\ \square\ \square\ c)}{N_{1+}(\bullet\ \square\ \square\ \bullet)}}{N_{1+}(\bullet a\ \square\ \bullet)}\right)}{c(a\ b\ \square)}$$

By using the distributive property:

$$\begin{aligned} P_{GLM}(c \mid a\ b) = \quad & c^d(a\ b\ c) \cdot \frac{1}{c(a\ b\ \square)} \\ + \quad & N_{1+}^d(\bullet\ \square b\ c) \cdot \frac{\gamma(a\ b)}{2\,c(a\ b\ \square)\,N_{1+}(\bullet\ \square b\ \bullet)} \\ + \quad & N_{1+}^d(\bullet a\ \square c) \cdot \frac{\gamma(a\ b)}{2\,c(a\ b\ \square)\,N_{1+}(\bullet a\ \square\ \bullet)} \\ + \quad & N_{1+}(\bullet\ \square\ \square\ c) \cdot \frac{\gamma(a\ b)}{2 \cdot 1\,c(a\ b\ \square)}\left(\frac{\gamma(\square b)}{N_{1+}(\bullet\ \square b\ \bullet)\,N_{1+}(\bullet\ \square\ \square\ \bullet)} + \right. \\ & \left. \frac{\gamma(a\ \square)}{N_{1+}(\bullet a\ \square\ \bullet)\,N_{1+}(\bullet\ \square\ \square\ \bullet)}\right) \end{aligned} \tag{A.11}$$

B

SQL-EXAMPLE

With database examples, and fully expanded SQL code, probably MKN query.

C

NOTES

Remove Notes before releasing.

C.1 WEIGHTED SUM MKN ALGORITHM FROM CODE

Algorithm C.1 Weighted Sum MKN algo from Code (likely optimizable)

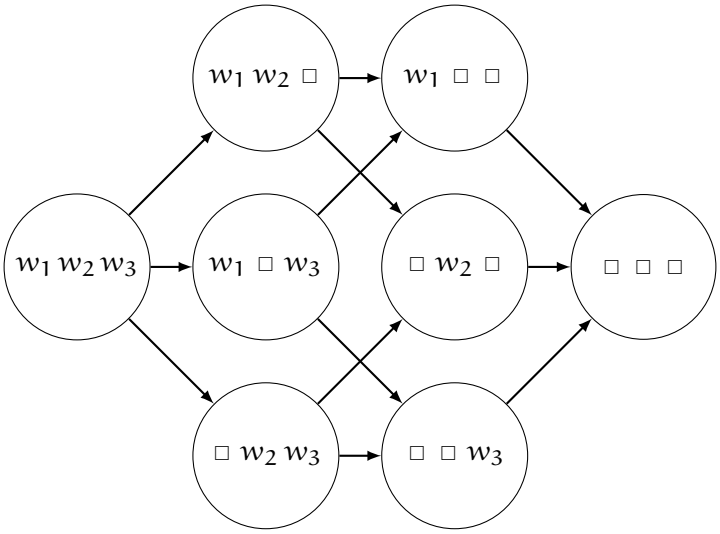
Input: h ▷ History for which to determinate interpolation weights
Output: λ_i^h ▷ Array of interpolation weights

▷ Find first seen history h'
$h' \leftarrow h$
while $c(h'\,\square\,) = 0$ **do**
    $h' \leftarrow h'$.backoff()

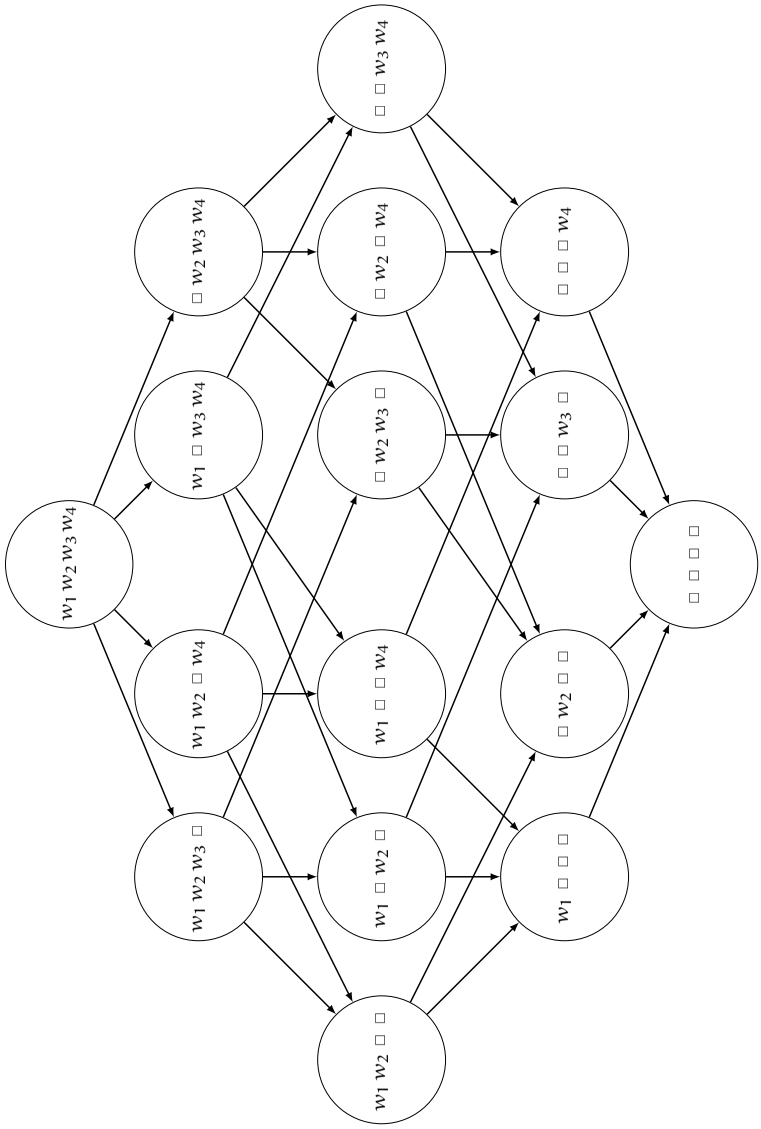
Better notation for backoff

$N = |h'| + 1$
$\lambda^h \leftarrow$ new array of size N
$\mathrm{nominator} \leftarrow 1$
$\mathrm{denominator} \leftarrow c(h'\,\square\,)$
$\lambda_1^h \leftarrow \frac{1}{\mathrm{denominator}}$
$\eta \leftarrow h'$
for $i \leftarrow 2, N$ **do**
    $\mathrm{nominator} \leftarrow \gamma(\eta)\frac{\mathrm{nominator}}{\mathrm{denominator}}$
    $\eta \leftarrow \eta$.backoff()
    $\mathrm{denominator} \leftarrow N_{1+}(\bullet\,\eta\,\bullet)$
    $\lambda_i^h \leftarrow \frac{\mathrm{nominator}}{\mathrm{denominator}}$

C.2 ALTERNATIVE BINOMIAL DIAMOND OF ORDER 3



C.3 BINOMIAL DIAMOND OF ORDER 4



C.4 SQL QUERY EXPANSION

C.4.1 *Modified Kneser Ney Language Model*

```
SELECT
  token.token AS arg,
  func( 1.count,  11.count, 111.count,
       x1.count, x11.count)
    AS score
FROM  token
LEFT OUTER JOIN    1
  ON history(  1) + token =   1.sequence
LEFT OUTER JOIN   11
  ON history( 11) + token =  11.sequence
LEFT OUTER JOIN  111
  ON history(111) + token = 111.sequence
LEFT OUTER JOIN   x1
  ON history( x1) + token =  x1.sequence
LEFT OUTER JOIN  x11
  ON history(x11) + token = x11.sequence
ORDER BY score DESC;
LIMIT k;
```

C.4.2 *Generalized Language Model*

```
SELECT
  token.token AS arg,
  func( 001.count,  011.count,  101.count, 111.count,
       x001.count, x011.count, x101.count)
    AS score
SELECT token, score
FROM token
LEFT OUTER JOIN  001
  ON history( 001) + token =  001.sequence
LEFT OUTER JOIN  011
  ON history( 011) + token =  011.sequence
LEFT OUTER JOIN  101
  ON history( 101) + token =  101.sequence
LEFT OUTER JOIN  111
  ON history( 111) + token =  111.sequence
LEFT OUTER JOIN x001
  ON history(x001) + token = x001.sequence
LEFT OUTER JOIN x011
  ON history(x011) + token = x011.sequence
LEFT OUTER JOIN x101
  ON history(x101) + token = x101.sequence
ORDER BY score DESC;
LIMIT k;
```

TODO LIST