Language guesser, language model, and smoothing

Ein **Sprachidentifizierer** (Language Guesser) bestimmt die wahrscheinlichste Sprache $\hat{L}$ eines gegebenen Textes T. Er berechnet:

$$\hat{L} = \arg\max_{L} p(L|T)$$

A language guesser takes as input a text T and compute p(L|T) for every language candidate L and return L that yields the highest p(L|T)

p is the language model of language L

Text T

Konfliktsituationen zwischen Bund und Ländern gab es in der Geschichte der Bundesrepublik immer wieder.

Language candidate L

English
German
Danish
Dutch
Finnish
French
Irish
Italian
Spanish
Swedish

The language guesser computes

p(L=Eng | Konflikts..)
p(L=De | Konflikts..)
p(L=Danish | Konflikts..)
...

best L = German

this probability distribution is the language model of each language

Language model

From Wikipedia, the free encyclopedia

A statistical **language model** is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability $P(w_1, \ldots, w_m)$ to the whole sequence.

examples

- n-gram language model
- neural network based language model

p_eng (I want to eat pizza) = 0.9
p_eng(I want to drink pizze) = 0.0023
p_eng(ich mag Pizza) = 0.000002

Ein **Sprachidentifizierer** (Language Guesser) bestimmt die wahrscheinlichste Sprache $\hat{L}$ eines gegebenen Textes T. Er berechnet:

$$\hat{L} = \arg\max_L p(L|T)$$

How to compute p(L|T) ?

- adjust the formula(model) using Bayes theorem
- remove terms that are constant
- interpret T as a seqeunce of characters or words
- apply chain rule
- simplify the formula using Markov assumption

Trick: Anwendung des Bayes'sche Theorems:

$$\arg\max_L p(L|T) = \arg\max_L \frac{p(T|L)p(L)}{p(T)}$$

Die Textwahrscheinlichkeit $p(T)$ ist eine Konstante, die keinen Einfluss auf das Ergebnis der arg-max-Operation hat und daher weggelassen werden kann:

$$\arg\max_L p(L|T) = \arg\max_L p(T|L)p(L)$$

Falls keine Information über die Apriori-Wahrscheinlichkeiten $p(L)$ der Sprachen verfügbar ist, können wir sie als gleichverteilt annehmen und ebenfalls ignorieren:

$$\arg\max_l p(L|T) = \arg\max_l p(T|L)$$

Angenommen der Text T besteht aus der Zeichenfolge $a_1, a_2, ..., a_n =: a_1^n$.

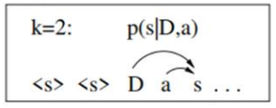
Wir zerlegen $p(T|L)$ in ein Produkt von bedingten Wahrscheinlichkeiten:

$$\begin{aligned} p(T|L) &= p_L(a_1^n) = p_L(a_1, ..., a_n) \\ &= p_L(a_1)p_L(a_2|a_1)p_L(a_3|a_1, a_2)...p_L(a_n|a_1, ..., a_{n-1}) \\ &= \prod_{i=1}^{n} p_L(a_i|a_1, ..., a_{i-1}) \end{aligned}$$

Markowmodelle

Nun machen wir die **vereinfachende** Annahme, dass a_i nur von den k vorhergehenden Zeichen abhängt

$$p_L(T) = \prod_{i=1}^{n+1} p_L(a_i|a_{i-k}...a_{i-1})$$



Ein solches Modell heißt **Markowmodell** der Ordnung k.

Was wir oft benutzen werden

bedingte Wahrscheinlichkeit: $p(x|y) = \frac{p(x,y)}{p(y)}$

Kettenregel $p(x,y,z) = p(x)p(y|x)p(z|xy)$

Bayes'sches Theorem: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

Schätzung von Wahrscheinlichkeiten:

$$\tilde{p}(x) = \frac{f(x)}{N} \qquad N = \sum_{x'} f(x')$$

$$\tilde{p}(x|y) = \frac{f(x,y)}{f(y)} \qquad f(y) = \sum_{x'} f(x',y)$$

Kettenregel

Eine gemeinsame Wahrscheinlichkeit kann in ein Produkt bedingter Wahrscheinlichkeiten umgewandelt werden.

$$\begin{aligned} p(X_1 = x_1, X_2 = x_2, ..., X_n = x_n) &= p(x_1, x_2, ..., x_n) \\ &= p(x_1)p(x_2|x_1)...p(x_n|x_1, ..., x_{n-1}) \\ &= \prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1}) \end{aligned}$$

Beispiel: Für Folgen von 3 Wörtern gilt:

$p(W_1{=}Es, W_2{=}gibt, W_3{=}ein) =$
$p(W_1{=}Es)\ p(W_2{=}gibt|W_1{=}Es)\ p(W_3{=}ein|W_1{=}Es, W_2{=}gibt)$

kürzer aber weniger eindeutig: $p(Es, gibt, ein) = p(Es)\ p(gibt|Es)\ p(ein|Es, gibt)$

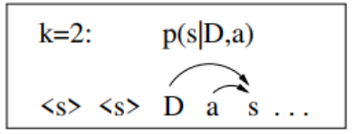
$p(W_i = w), i \in \{1,2,3\}$: Summe der Wahrscheinlichkeiten aller Wort-Tripel, bei denen an Position i das Wort w steht.

The initial formula: $\hat{L} = \arg\max_L p(L|T)$

Markowmodelle

Nun machen wir die **vereinfachende** Annahme, dass a_i nur von den k vorhergehenden Zeichen abhängt

$$p_L(T) = \prod_{i=1}^{n+1} p_L(a_i|a_{i-k}...a_{i-1})$$



Ein solches Modell heißt **Markowmodell** der Ordnung k.

Für $k = 2$ bekommen wir:
p(Er ölt) =
p(E|⟨s⟩,⟨s⟩) p(r|⟨s⟩,E) p(_ |E,r) p(ö|r,_) p(l|_,ö) p(t|ö,l) p(⟨/s⟩|l,t)

⇒ Es werden k Startsymbole $\langle s\rangle$ und ein Endesymbol $\langle/s\rangle$ hinzugefügt:
$a_{-1} = a_0 = \langle s\rangle$ und $a_{n+1} = \langle/s\rangle$

Die Startsymbole sind notwendig, damit $p(a_1|a_{-1}, a_0)$ definiert ist.
Ohne das Endesymbol würden die Wahrsch. aller Zeichenfolgen zu mehr als 1 summieren.

Markov model computes p(text) as follows.
The probability of a text in this language model is the product of the conditional probability of each chracter a_i given the previous characters a_i-k to a_i-1
and we compute the product for a_i=1 until a_i=n+1

$$p_L(T) = \prod_{i=1}^{n+1} p_L(a_i|a_{i-k}...a_{i-1})$$

Example:

- T= a1,a2,a3
- we pick k=2
  - k is the number of characters/words/items in the context)
  - k=2 means we are using 3-grams Markov model

p(a1,a2,a3) = p(a1|<s>, <s>) *
p(a2| <s>, a1) *
p(a3| a1,a2) *
p(</s>|a2,a3)

The order of the Markov model (k) is a parameter that we can set before we define the model.

Here you see how different k values impact the performance of the language model

Für zunehmende Ordnung n modellieren Markowmodelle die Sprache immer genauer:

Modell	Crossentropie
Uniforme Verteilung	4.76 Bit ($= \log_2 27$)
Ordnung 0	4.03 Bit
Ordnung 1	2.8 Bit
Mensch	1.3 Bit (Shannon)

language models with a **low cross entropy** are considered better/more accurate than those with a higher cross entropy

Aber, bei Modellen höherer Ordnung nimmt die Zahl der Wk.-Parameter schnell zu, und es wird immer schwieriger, deren Werte zu schätzen.

Reasons to compute until n+1 (adding end symbol </s>)

Markowmodelle

Angenommen der Text T besteht aus der Zeichenfolge $a_1, a_2, ..., a_n =: a_1^n$.

Wir zerlegen $p(T|L)$ in ein Produkt von bedingten Wahrscheinlichkeiten:

$$\begin{aligned} p(T|L) &= p_L(a_1^n) = p_L(a_1, ..., a_n) \\ &= p_L(a_1)p_L(a_2|a_1)p_L(a_3|a_1, a_2)...p_L(a_n|a_1, ..., a_{n-1}) \\ &= \prod_{i=1}^{n} p_L(a_i|a_1, ..., a_{i-1}) \end{aligned}$$

Problem: Die Kettenregel ist eigentlich nur anwendbar, wenn alle Texte dieselbe Länge n besitzen. Bei variablem n ist $\prod_{i=1}^{n} p_L(a_i|a_1, ..., a_{i-1})$ nur die Gesamtwahrscheinlichkeit aller Strings mit **Präfix** a_1^n.

Daher hätte *"Er liest ein Buc"* eine größere Wk. als *"Er liest ein Buch"*.

Wir lösen das Problem durch Hinzufügen einer eindeutigen Endemarkierung $a_{n+1} = \langle /s \rangle$:

$$p_L(a_1^n) = \prod_{i=1}^{n+1} p_L(a_i|a_1, ..., a_{i-1})$$

Ohne das Endesymbol würden die Wahrsch. aller Zeichenfolgen zu mehr als 1 summieren.

Exercise:

- We want to identify the language of the text „Es war“.

How is p(Es war) defined in **a character-level** language model with k = 2 (trigram model)?

p(Es_war) =

- How is p(Sprach) defined in a **character-level** language model with k = 4 (5-gram model)?

p(Sprach) = ...

- How is p(the man sleeps) defined in a **word level** language model with k = 2 (trigram model)?

p(the man sleeps) = ...

Solution:

- We want to identify the language of the text „Es war“.

How is p(Es war) defined in **a character-level** language model with k = 2 (trigram model)?

p(Es_war) =

```
['<s>', '<s>', 'E', 's', '_', 'w', 'a', 'r', '</s>']
 i=-1    i=0   i=1                      i=n   i=n+1
 p(E|<s>,<s>)
 p(s|<s>,E)
 p(_|E,s)
 p(w|s,_)
 p(a|_,w)
 p(r|w,a)
 p(</s>|a,r)
```

$$p_L(T) = \prod_{i=1}^{n+1} p_L(a_i | a_{i-k} \ldots a_{i-1})$$

- How is p(Sprach) defined in a character level language model with k = 4 (5-gram model)?

```
['<s>', '<s>', '<s>', '<s>', 'S', 'p', 'r', 'a', 'c', 'h', '</s>']
```

p(Sprach) = ...

```
p(S|<s>,<s>,<s>,<s>)
p(p|<s>,<s>,<s>,S)
p(r|<s>,<s>,S,p)
p(a|<s>,S,p,r)
p(c|S,p,r,a)
p(h|p,r,a,c)
p(</s>|r,a,c,h)
```

- How is p(the man sleeps) defined in a **word level** language model with k = 2 (trigram model)?

```
['<s>', '<s>', 'The', 'man', 'sleeps', '</s>']
```

p(the man sleeps) =

```
p(The|<s>,<s>)
p(man|<s>,The)
p(sleeps|The,man)
p(</s>|man,sleeps)
```

Parameterschätzung

- Assume we want to compute p of the text „Sprach“ for every language candidate.
- First, we have to define the language model (every language will use the same type of model but having different parameters).
- We decide we will use a 5-gram character language model (Markov language model with order k =4)

Our language model defines p(Sprach) as follows.
p(Sprach) = p(S|<s>,<s>,<s>,<s>) p(p|<s>,<s>,<s>,S) p(r|<s>,<s>,S,p) p(a|<s>,S,p,r) p(c|S,p,r,a) p(h|p,r,a,c) p(</s>|r,a,c,h)

- This means in oder to compute p(Sprach), we have to compute all these conditional probabilities
- the conditional probabilities are the **parameters** of our language model
- A classical approach to estimate p(w5| w1,w2,w3,w4) is to use **the relative frequency (relative Häufigkeiten)**

Die Parameter werden mit relativen Häufigkeiten aus Trainingsdaten geschätzt:

$$p(c|Spra) = \frac{f(Sprac)}{f(Spra)}$$

the frequency of „Sprac“ we found in the corpus

the frequency of the context „Spra“

p(Sprach) = p(S|<s>,<s>,<s>,<s>) p(p|<s>,<s>,<s>,S) p(r|<s>,<s>,S,p) p(a|<s>,S,p,r) p(c|S,p,r,a) p(h|p,r,a,c) p(</s>|r,a,c,h)

For example, we want to estimate p(c|S,p,r,a)

Die Parameter werden mit relativen Häufigkeiten aus Trainingsdaten geschätzt:

$$p(c|Spra) = \frac{f(Sprac)}{f(Spra)}$$

We have to extract the freq of 5-grams from the corpus (here German corpus)

text

```
Der Unterschied zu heute: Während es damals
eindeutige Mehrheiten gab, ist das Bild in den
Ländern heute völlig bunt. In den 16
Bundesländern gibt es, je nachdem wie man
rechnet, mindestens elf verschiedene
Farbkombinationen. Schwarz-rot, grün-schwarz,
rot-rot-grün, Jamaika, die Ampel, und einige
andere mehr. Nur einem Land, in Bayern, ist
keine der drei Ampel-Parteien mit in der
Regierung.
```

n-grams

n-grams	Freq
d e r _ u	2
e r _ u n	2
r _ u n t	5
_ u n t e	31
u n t e r	12
n t e r s	42
t e r s c	
e r s c h	
r s c h i	
s c h i e	
c h i e d	

Assume f(Sprac) = 5 and f(Spra) = 15
then p(c | Spra) = 5 / 15
= 0.33

We would then do the same for other components in the formula

p(Sprach) = p(S|<s>,<s>,<s>,<s>) p(p|<s>,<s>,<s>,S) p(r|<s>,<s>,S,p) p(a|<s>,S,p,r) p(c|S,p,r,a) p(h|p,r,a,c) p(</s>|r,a,c,h)

Problem with the relative frequency method: if any f(5gram) is zero, then the whole probability will be zero - --- > to solve this problem, we have to smooth the probabilty

Sprachidentifizierung

Vorgehen:

- Man sammelt für jede relevante Sprache ein **Textkorpus**.
- Man trainiert für jede Sprache ein Markowmodell, d.h. man **schätzt** seine Parameter aus dem Korpus.
- Man berechnet die **Wahrscheinlichkeit des Textes** für jedes Sprachmodell (als Produkt bedingter Wahrscheinlichkeiten). (Falls bekannt, multipliziert man noch die Apriori-Wk. p(L) der Sprache.)
- Man gibt die Sprache aus, bei der die **Wahrscheinlichkeit** am größten war.

1

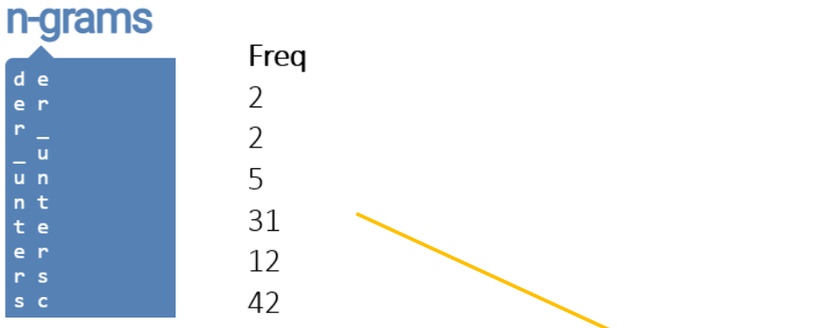
English text corpus

German text corpus

French text corpus

2

extract n-gram frequency



$$p(c|Spra) = \frac{f(Sprac)}{f(Spra)}$$

3

new text to classify

"Sprach"

4

compute the prob p(L | T) for each language (here we use a bigram model)

p_eng(Sprach) = p(S|<s>,<s>) p(p|<s>,S) p(r|S,p) p(a|p,r) p(c|r,a) p(h|a,c) p(</s>|c,h)
p_de(Sprach) = p(S|<s>,<s>) p(p|<s>,S) p(r|S,p) p(a|p,r) p(c|r,a) p(h|a,c) p(</s>|c,h)
p_fr(Sprach) = p(S|<s>,<s>) p(p|<s>,S) p(r|S,p) p(a|p,r) p(c|r,a) p(h|a,c) p(</s>|c,h)

then argmax -> return the best language (having the highest p(L|T))
or convert p to cross entropy, then argmin -> return the best language (having the lowest cross entropy)

Sprachidentifizierung

Beispiele:

Das ist ein deutscher Satz.

Modell	Bits/Symbol
Dänisch	4.048654
Niederländisch	3.027868
Englisch	3.604354
Finnisch	3.976386
Französisch	3.426143
Deutsch	**1.754679**
Italienisch	4.256318
Portugiesisch	4.362893
Spanisch	4.314057
Schwedisch	3.879200

This is an English sentence.

Modell	Bits/Symbol
Dänisch	3.588310
Niederländisch	3.558052
Englisch	**1.578621**
Finnisch	4.080841
Französisch	3.091803
Deutsch	2.806066
Italienisch	3.980872
Portugiesisch	3.839128
Spanisch	3.833479
Schwedisch	3.492607

Statt der Wahrscheinlichkeit $p(a_1^n)$ wird hier die Crossentropie gezeigt:

$$H(a_1^n, p) = -\frac{1}{n} log_2 p(a_1^n) \text{ Bits/Symbol}$$

convert the prob to cross entropy and argmin over L

Advantages of applying log to the prob

- p can be very small. We get a bigger value after applying log to it
- log can turn multiplication to addition which is a cheaper operation
- this makes the computation faster
- the final value is a big value (easy to interpret)

Example

log p(a1a2a3) = log [p(a1| ..) p(a2|..) p(a3|...) p(</s>|..)]

= log p(a1|...) + log p(a2|...) + log p(a3| ..) + log p(</s>|..)

https://chrispiech.github.io/probabilityForComputerScientists/en/part1/log_probabilities/

Log Probabilities

A log probability $\log \mathrm{P}(E)$ is simply the log function applied to a probability. For example if $\mathrm{P}(E) = 0.00001$ then $\log \mathrm{P}(E) = \log(0.00001) \approx -11.51$. Note that in this book, the default base is the natural base e. There are many reasons why log probabilities are an essential tool for digital probability: (a) computers can be rather limited when representing very small numbers and (b) logs have the wonderful ability to turn multiplication into addition, and computers are much faster at addition.

You may have noticed that the log in the above example produced a negative number. Recall that $\log b = c$, with the implied natural base e is the same as the statement $e^c = b$. It says that c is the exponent of e that produces b. If b is a number between 0 and 1, what power should you raise e to in order to produce b? If you raise e^0 it produces 1. To produce a number less than 1, you must raise e to a power less than 0. That is a long way of saying: if you take the log of a probability, the result will be a negative number.

$$0 \leq \mathrm{P}(E) \leq 1 \qquad \text{Axiom 1 of probability}$$
$$-\infty \leq \log \mathrm{P}(E) \leq 0 \qquad \text{Rule for log probabilities}$$

1. Products become Addition

The product of probabilities $\mathrm{P}(E)$ and $\mathrm{P}(F)$ becomes addition in logarithmic space:

$$\log(\mathrm{P}(E) \cdot \mathrm{P}(F)) = \log \mathrm{P}(E) + \log \mathrm{P}(F)$$

Parameterschätzung

Parameterschätzung

Maximum-likelihood estimation (MLE)

Die Parameter werden gewöhnlich mit relativen Häufigkeiten aus Trainingsdaten geschätzt.

Beispiel: Wort n-Gramm-Wahrscheinlichkeiten

$$p(w_n|\underbrace{w_1 \ldots w_{n-1}}_{\text{context}}) = \frac{f(w_1 \ldots w_n)}{\sum_w f(w_1 \ldots w_{n-1} w)}$$

This part shows explicitly how f(context) should be calculated

example: $p(c|Spra) = \frac{f(Sprac)}{f(Spra)}$

Aber: Neue Texte können n-Gramme enthalten, die nicht in den Trainingsdaten auftauchten. Ihre relative Häufigkeit ist 0.

Smoothing methods

Addiere-1 Glättung (Laplace-Glättung)

Bei der **Addiere-1** Glättung wird 1 zu allen Häufigkeiten (auch Nullhäufigkeiten) addiert:

$$p(w) = \frac{f(w)+1}{N+B}$$

N = Zahl der Wort-Tokens B = Anzahl der Wort-Types

Addiere-λ Glättung

reduziert das Ausmaß der Glättung

Addiere-λ Glättung

$$p(w) = \frac{f(w)+\lambda}{N+B\lambda} \quad \text{mit } 0 < \lambda < 1$$

Absolute Discounting

Linear Discounting (Add-λ Glättung) führt zu schlechten Ergebnissen, wenn die Daten eine Zipf'sche Verteilung aufweisen.

Absolute Discounting

- subtrahiert einen festen Betrag (Discount) von den Häufigkeiten der beobachteten Ereignisse
- verteilt die Discounts über die unbeobachteten Ereignisse.

$$p(w) = \begin{cases} \frac{f(w)-\delta}{N} & \text{falls } f(w) > 0 \\ \frac{(B-N_0)\delta}{N_0 N} & \text{sonst} \end{cases}$$

For our Markov language model, we will use this

Katz'sche Backoff-Glättung

Backoff-Glättung wird für bedingte Wahrscheinlichkeiten verwendet.

Statt die Wahrscheinlichkeitsmasse gleichmäßig über alle unbeobachteten Wörter zu verteilen, werden Sie gemäß einer Backoff-Verteilung mit kleinerem Kontext verteilt.

$$p(w_i|w_{i-k}^{i-1}) = \begin{cases} \frac{f(w_{i-k}^{i})-\delta_k}{f(w_{i-k}^{i-1})} & \text{falls } f(w_{i-k}^{i}) > 0 \\ \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1}) & \text{sonst} \end{cases}$$

Backoff-Glättung mit Interpolation

Hier wird jeweils die mit α gewichtete Backoff-Wahrscheinlichkeit hinzuaddiert.

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

Auch hier gilt: $f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1} w)$

Die max-Operation verhindert, dass der vordere Term negativ wird.

$\Rightarrow$ Interpolierte Modelle sind meistens etwas besser und die Berechnung des Backoff-Faktors ist einfacher.

Addiere-1 Glättung (Laplace-Glättung)

Bei der **Addiere-1** Glättung wird 1 zu allen Häufigkeiten (auch Nullhäufigkeiten) addiert:

$$p(w) = \frac{f(w) + 1}{N + B}$$

N = Zahl der Wort-Tokens B = Anzahl der Wort-Types

In Laplace smoothing, 1 (one) is added to all the counts and thereafter, the probability is calculated. This is one of the most trivial smoothing techniques out of all the techniques.

Maximum likelihood estimate (MLE) of a word w_i occuring in a corpus can be calculated as the following. N is total number of words, and $count(w_i)$ is count of words for whose probability is required to be calculated.

compare MLE and add 1 smoothing

MLE: $P(w_i) = \frac{count(w_i)}{N}$

After applying Laplace smoothing, the following happens. Adding 1 leads to extra V observations.

MLE: $P_{Laplace}(w_i) = \frac{count(w_i)+1}{N+V}$

V is B in our slide

unigram	freq
the	2
man	3
eats	1

freq from training data

intput text

„the man eats rice and veggy“

NOT CORRECT

unigram	freq
the	2
man	3
eats	1
rice	0
veggie	0
and	0

unigram	f_add1
the	2 +1
man	3 +1
eats	1 +1
rice	0 +1
veggie	0 +1
and	0 +1

unigram	f_add1
the	3
man	4
eats	2
rice	1
veggie	1
and	1

we want to compute p(the man eats rice and veggie) with a unigram model.

p(the man eats rice and veggie) = p(the) p(man) p(eats) .....

As an example, will compute p(the) and p(rice) using the **add-1 smoothing**

$$p(w) = \frac{f(w)+1}{N+B}$$

N=3+4+2+1+1+1 = 12
B= 6

p(the) = f_add1(the) / N + B
= 3 / (12 / 6)

p(rice) = f_add1(rice) / N + B
= 1 / (12 / 6)

* note about how to compute the number of types B (or V)

- Typically, we assume $V = \{w : c(w) > 0\} \cup \{\text{UNK}\}$

https://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf (page 9)

CORRECT

Addiere-1 Glättung (Laplace-Glättung)

Bei der **Addiere-1** Glättung wird 1 zu allen Häufigkeiten (auch Nullhäufigkeiten) addiert:

$$p(w) = \frac{f(w) + 1}{N + B}$$

N = Zahl der Wort-Tokens B = Anzahl der Wort-Types

We can only use this method **when we know the final value of N and B.** Meaning, we would give the model the words from a training corpus (these are words whose frequency > 0) and also words whose frequency = 0 (for example, valid words that are defined in lexica/dictionaries but are not present in the training corpus). Then we would have the frequency table that looks like this

unigram	freq
the	2
man	3
eats	1
rice	0
veggie	0
and	0

Then we can use the add-1 smoothing to smooth p(w) for any word w.

unigram	freq
the	2
man	3
eats	1
rice	0
veggie	0
and	0

unigram	f_add1
the	2 +1
man	3 +1
eats	1 +1
rice	0 +1
veggie	0 +1
and	0 +1

$$p(w) = \frac{f(w)+1}{N+B}$$

unigram	f_add1
the	3
man	4
eats	2
rice	1
veggie	1
and	1

N=3+4+2+1+1+1 = 12
B= 6

p(the) = f_add1(the) / N + B
= 3 / (12 / 6)

p(rice) = f_add1(rice) / N + B
= 1 / (12 / 6)

....

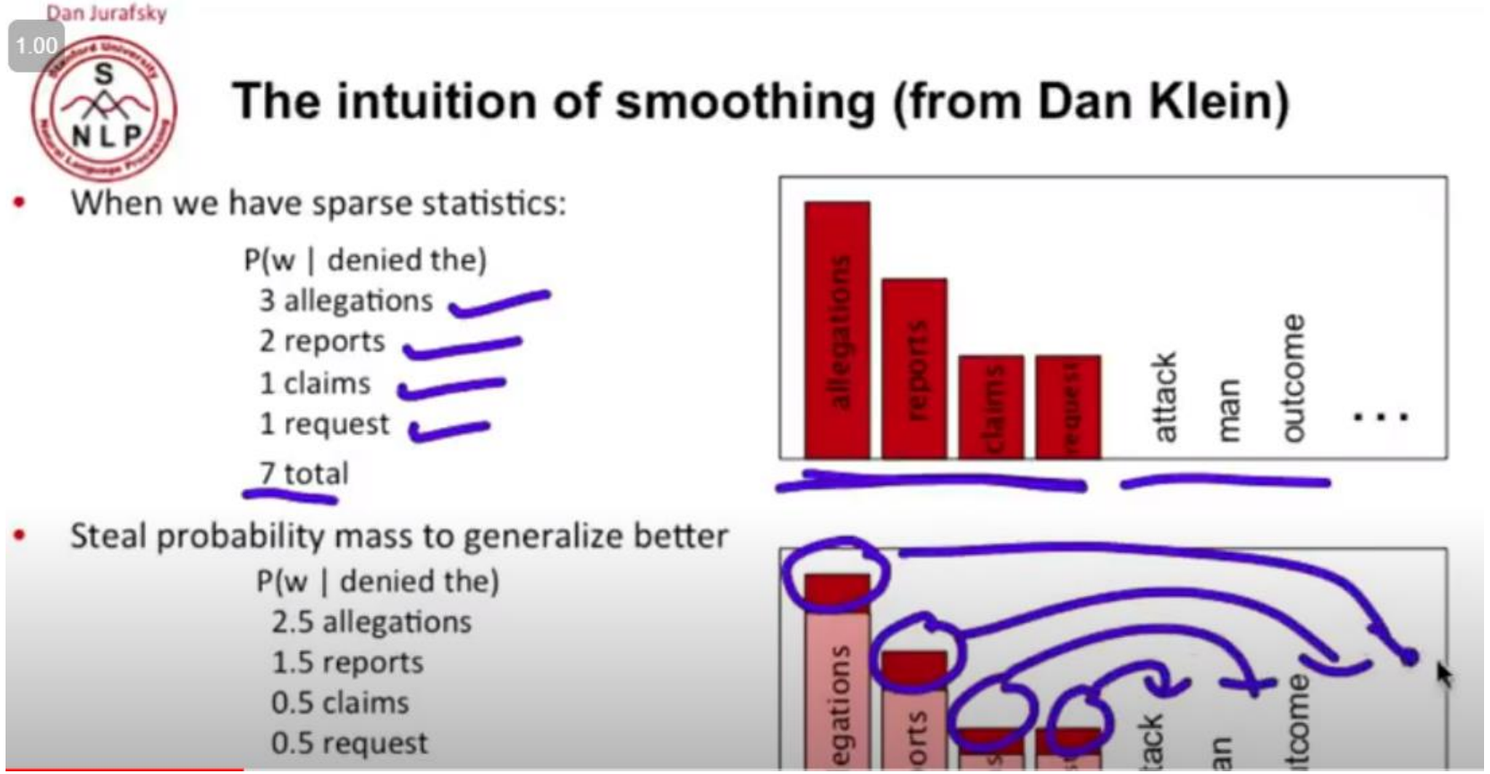
The model should know from the beginning that *rice, veggie, and* has a frequency of 0. Meaning they are not found in the training corpus, but they are valid words that we feed to the model using knowledge from lexica/dictionaries.

´Then we can use p(w) for the following example.

p(the man eats rice and veggie) = p(the) p(man) p(eats) .....

CORRECT

- This tutorial shows how to compute add-1 smoothing for conditional probability (e.g. bigram, trigram).
- It also explains the meaning of MLE.



https://www.youtube.com/watch?v=gYBtv91tvr0&ab_channel=ArtificialIntelligence-AllinOne

Add-one smoothing

- Add one to all of the counts before normalizing into probabilities
- MLE unigram probabilities

$$P(w_x) = \frac{count(w_x)}{N}$$

corpus length in word tokens

- Smoothed unigram probabilities

$$P(w_x) = \frac{count(w_x)+1}{N+V}$$

vocab size (# word types)

Add-one smoothing: bigrams

- MLE bigram probabilities

$$P(w_n \mid w_{n-1}) = \frac{count(w_{n-1}w_n)}{count(w_{n-1})}$$

- Laplacian bigram probabilities

$$P(w_n \mid w_{n-1}) = \frac{count(w_{n-1}w_n)+1}{count(w_{n-1})+V}$$

http://www.cs.cornell.edu/courses/cs4740/2014sp/lectures/smoothing+backoff.pdf

Addiere-1 Glättung (Laplace-Glättung)

Bei der **Addiere-1** Glättung wird 1 zu allen Häufigkeiten (auch Nullhäufigkeiten) addiert:

$$p(w) = \frac{f(w) + 1}{N + B}$$

N = Zahl der Wort-Tokens B = Anzahl der Wort-Types

Die Addiere-1 Glättung ist **optimal**, falls die uniforme Verteilung am wahrscheinlichsten ist, was in der Sprachverarbeitung selten der Fall ist, wo Zipf'sche Verteilungen dominieren.

Die Addiere-1 Glättung **überschätzt** daher die Wahrscheinlichkeit ungesehener Wörter.

Addiere-λ Glättung

reduziert das Ausmaß der Glättung

Addiere-λ Glättung

$$p(w) = \frac{f(w) + \lambda}{N + B\lambda} \quad \text{mit } 0 < \lambda < 1$$

Die Addiere-λ Glättung ist äquivalent zu einer Interpolation der relativen Häufigkeit f(w)/N mit einer uniformen Verteilung 1/B.

$$p(w) = \mu\frac{f(w)}{N} + (1-\mu)\frac{1}{B} \quad \text{wobei } \mu = \frac{N}{N + B\lambda}$$

$\Rightarrow$ Addiere-λ Glättung reduziert die Wahrscheinlichkeit von gesehenen Wörtern um einen Betrag, der linear mit der Wahrscheinlichkeit steigt.

$\Rightarrow$ Linear Discounting

Übung: Zeigen Sie die Äquivalenz zur ersten Formel, indem Sie die Definition von μ in die zweite Formel einsetzen und umformen.

here we do not add 1 but add a number less than 1 (but bigger than 0)

- If a word has a non-zero freq, an amount of a prob of that word will be taken and add to other unseen words.
- If a word has a high prob (high freq), a high amount of prob will be subtract from it.

Absolute Discounting

Linear Discounting (Add-λ Glättung) führt zu schlechten Ergebnissen, wenn die Daten eine Zipf'sche Verteilung aufweisen.

Absolute Discounting

- subtrahiert einen festen Betrag (Discount) von den Häufigkeiten der beobachteten Ereignisse
- verteilt die Discounts über die unbeobachteten Ereignisse.

$$p(w) = \begin{cases} \frac{f(w)-\delta}{N} & \text{falls } f(w) > 0 \\ \frac{(B-N_0)\delta}{N_0 N} & \text{sonst} \end{cases}$$

B: Zahl der Types
N: Zahl der Tokens
N_i: Zahl der Types mit Häufigkeit i

Absolute Discounting

Kneser, Essen, Ney leiten die folgende Formel für den Discountbetrag δ her:

$$\delta = \frac{N_1}{N_1 + 2N_2}$$

N0 = the number of types that has freq = 0
N1 = the number of types that has freq = 1
N2 = the number of types that has freq = 2

unigram	freq
the	2
man	3
eats	1

freq from training data

intput text

„the man eats rice and veggy“

NOT CORRECT

unigram	freq
the	2
man	3
eats	1
rice	0
veggy	0
and	0

we want to compute p(the man eats rice and veggy) with a unigram model.

p(the man eats rice and veggy) = p(the) p(man) p(eats) .....

Here, we will compute p(the) and p(rice) using **Absolute Discounting.**

$$p(w) = \begin{cases} \frac{f(w)-\delta}{N} & \text{falls } f(w) > 0 \\ \frac{(B-N_0)\delta}{N_0 N} & \text{sonst} \end{cases}$$

B: Zahl der Types
N: Zahl der Tokens
N_i: Zahl der Types mit Häufigkeit i

δ = discount

$$\delta = \frac{N_1}{N_1 + 2N_2}$$

- p(the)
  - f(the) = 2 which is > 0 so we use the first formula
  - compute the discount
    - N1= 1 (eat)
    - N2 = 1 (the)
    - discount = 1 / (1 + 2*1) = 1 / 3
  - N = 2+3+1 = 6

  p(the) = 2 – (1/3) / 6

- p(rice)
  - f(rice) = 0, so we use the second formula
  - B = 6
  - N0 = 3 (rice, veggy, and)
  - discount = 1/3
  - N= 6

  p(rice) = (6 – 3) * (1/3) / (3 * 6)

CORRECT

Here we will compute p(the) and p(rice) using **Absolute Discounting**

$$p(w) = \begin{cases} \frac{f(w)-\delta}{N} & \text{falls } f(w) > 0 \\ \frac{(B-N_0)\delta}{N_0 N} & \text{sonst} \end{cases}$$

B: Zahl der Types
N: Zahl der Tokens
N_i: Zahl der Types mit Häufigkeit i

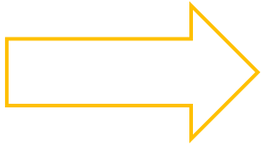
δ = discount

$$\delta = \frac{N_1}{N_1 + 2N_2}$$

from training corpus

words from dictionaries

unigram	freq
the	2
man	3
eats	1
rice	0
veggie	0
and	0



- p(the)
  - f(the) = 2 which is > 0 so we use the first formula
  - compute the discount
    - N1= 1 (eat)
    - N2 = 1 (the)
    - discount = 1 / (1 + 2*1) = 1 / 3
  - N = 2+3+1 = 6

  p(the) = 2 – (1/3) / 6

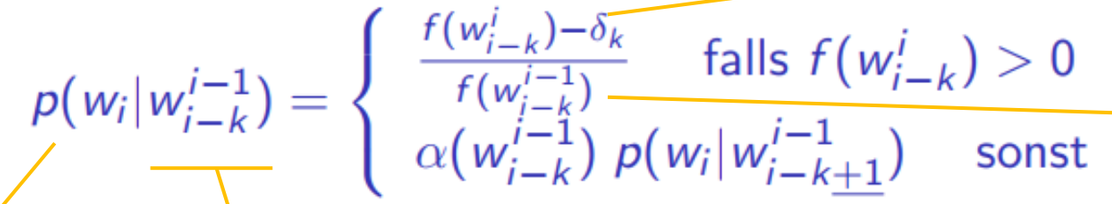
- p(rice)
  - f(rice) = 0, so we use the second formula
  - B = 6
  - N0 = 3 (rice, veggy, and)
  - discount = 1/3
  - N= 6

  p(rice) = (6 – 3) * (1/3) / (3 * 6)

Katz'sche Backoff-Glättung

Backoff-Glättung wird für bedingte Wahrscheinlichkeiten verwendet.

Statt die Wahrscheinlichkeitsmasse gleichmäßig über alle unbeobachteten Wörter zu verteilen, werden Sie gemäß einer Backoff-Verteilung mit kleinerem Kontext verteilt.



$$p(w_i|w_{i-k}^{i-1}) = \begin{cases} \frac{f(w_{i-k}^{i})-\delta_k}{f(w_{i-k}^{i-1})} & \text{falls } f(w_{i-k}^{i}) > 0 \\ \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1}) & \text{sonst} \end{cases}$$

freq of 3-gram „a1,a2,a3)

discount for order k

- if k =2, we compute the discount by looking at the 3-gram frequency table.
- k is the number of contexts in the n-gram

freq of (context)
This is f(a1,a2) in our example

p(a3| a1, a2)

- the probability of a3 given a1,a2

this part is called „context“

explain the first formula

explain the second formula

$$p(w_i|w_{i-k}^{i-1}) = \begin{cases} \frac{f(w_{i-k}^{i})-\delta_k}{f(w_{i-k}^{i-1})} & \text{falls } f(w_{i-k}^{i}) > 0 \\ \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1}) & \text{sonst} \end{cases}$$

p(a3| a1, a2)

backoff-factor of the context (a1,a2)
* this is **not** alpha multiplies with the context (a1,a2)

$$\alpha(C) = \frac{1 - \sum_{w:f(C,w)>0} \frac{f(C,w)-\delta}{f(C)}}{1 - \sum_{w:f(C,w)>0} p(w|C')}$$

This is how we compute the backoff-factor. This formula is, however, **not relevant for us** because we will later use another formula for computing the prob (the interpolation version)

- this part is **the backoff distribution (Backoff-Verteilung)**
- It is the prob of the smaller n-gram
- It is smaller because we reduce the context by removing the first item in the context out of the 3-gram (back off step)

E.g. we start with p(a3| a1, a2) , then we reduce the context by 1 item
= p(a3| ~~a1~~, a2)
= p(a3| a2)

To compute p(a3| a2), we will use the main backoff formula recursively until we have only p(unigram), meaning p(a3).
At p(a3) we will compute the formula the last time.

* We will mostly focus on this method

Backoff-Glättung mit Interpolation

Hier wird jeweils die mit α gewichtete Backoff-Wahrscheinlichkeit hinzuaddiert.

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\ p(w_i|w_{i-k+1}^{i-1})$$

Auch hier gilt: $f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1}w)$

Die max-Operation verhindert, dass der vordere Term negativ wird.

$\Rightarrow$ Interpolierte Modelle sind meistens etwas besser und die Berechnung des Backoff-Faktors ist einfacher.

The original Backoff formula (not interpolation)

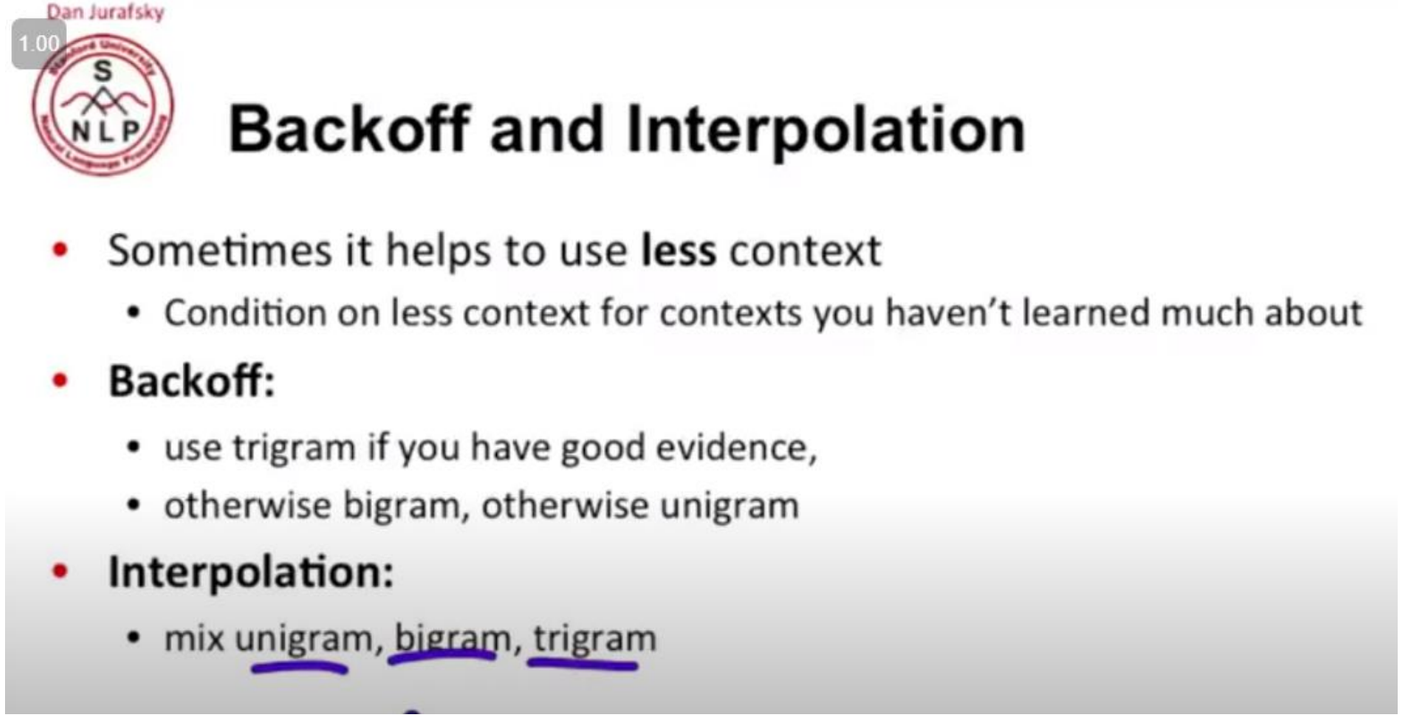
$$p(w_i|w_{i-k}^{i-1}) = \begin{cases} \frac{f(w_{i-k}^{i}) - \delta_k}{f(w_{i-k}^{i-1})} & \text{falls } f(w_{i-k}^{i}) > 0 \\ \alpha(w_{i-k}^{i-1})\ p(w_i|w_{i-k+1}^{i-1}) & \text{sonst} \end{cases}$$

With the new formula, any kind of ngram will use the same formula regardless of its freq (unlike the original backoff formula)

This means, for ngram that has non zero freq, we also get to compute the backoff steps (reduce the context of it).

https://www.youtube.com/watch?v=PC0nIk4-HoI&ab_channel=ArtificialIntelligence-AllinOne

interpolation + backoff explanation



Explain the formula

$$\delta = \frac{N_1}{N_1 + 2N_2}$$

discount_k=2

the backoff prob of the ngram whose context has been reduced
E.g. p(a3| ~~a1,~~ a2)

To compute this, we use the main formula recursively

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

f(a1, a2, a3)

p(a3| a1, a2)

here we have
k=2

f(a1, a2)
f of context

the part is often refered to as
„Relative Häufigkeit mit Discount“
(in out lecture)

backoff factor

$$\alpha(C) = 1 - \sum_{w:f(C,w)>0} \frac{f(C,w) - \delta}{f(C)}$$

f(a1,a2,w)

f(a1,a2)

C means context
e.g. alpha(a1,a2)

„Relative Häufigkeit mit Discount“
of every ngram (a1,a2,w)
that has freq > 0

Example (word level language model):

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

Auch hier gilt: $f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1} w)$

the man eats	2
the man walks	2
the man walked	3
we go to	1
go to school	1
go to the	1
to the park	2
at the park	1

p(eats| the man) = max [0, f(the man eats) – discount_2)] / f(the man) + alpha(the man) * p(eats| ~~the~~ man)

p(eats | man) = max [0, f(man eats) – discount_1)] / f(man) + alpha(man) * p(eats)

p(eats) = max [0, f(eats) – discount_1)] / f( ) + alpha( ) * f(eats) / N

We want to compute „the relative freq with discount“ part (let's call it p*)

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

Auch hier gilt: $f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1}w)$

p*(eats| the man)

p(eats| the man) = max [0, f(the man eats) – discount_2)] / f(the man) + alpha(the man) * p(eats| ~~the~~ man)

calculate f(the man eats), discount_2, and f(the man) by looking at the 3-gram table

- f(the man eats) = **2**
- compute discount_2
  - N1 = 4, N2 = 3
  - discount = 4 / (4 + 2*3) = 4 / 10 = **2 /5**

$$\delta = \frac{N_1}{N_1 + 2N_2}$$

- compute f(the man)

$$f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1}w)$$

f(the man) = f(the man eats) + f(the man walks) + f(the man walked)
= 2 + 2 +3
= **7**

the man eats	2
the man walks	2
the man walked	3
we go to	1
go to school	1
go to the	1
to the park	2
at the park	1

p*(eats| the man) = 2 – (2/5) / 7

Compute the backoff factor

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

p(eats| the man) = max [0, f(the man eats) – discount_2)] / f(the man) + alpha(the man) * p(eats| ~~the~~ man)

To calculate alpha(the man) , we first have to compute „the relative freq with discount“ p* of every 3-gram

$$\alpha(C) = 1 - \sum_{w:f(C,w)>0} \frac{f(C,w) - \delta}{f(C)}$$

	freq	p*: relative freq with discount
the man eats	2	0.3
the man walks	2	0.4
the man walked	3	0.5
we go to	1	0.1
go to school	1	0.2
go to the	1	0.3
to the park	2	0.22
at the park	1	0.6

alpha(the man) = 1 – [p*(eats| the man) + p*(walks| the man) + p*(walked| the man)]

e.g. f(the man eats)

$$\alpha(C) = 1 - \sum_{w:f(C,w)>0} \frac{f(C,w) - \delta}{f(C)}$$

In this example, k=2, so compute the discount using 3-gram freq table

we iterate over every word w that has f(C,w) > 0

f(the man)

Compute the backoff part

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

Auch hier gilt: $f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1} w)$

p(eats| the man) = max [0, f(the man eats) – discount_2)] / f(the man)+ alpha(the man) * p(eats| ~~the~~ man)

p(eats| man) = max [0, f(man eats) – discount_1)] / f(man) + alpha(man) * p(eats)

Create a bigram freq table

the man	5
man walks	3
man walked	4
the dog	3
dog walks	2
the bird	1
to the	2
the park	1

- max [0, f(man eats) – discount_1)] / f(man) --- compute this using bigram table (this is p*(eats| man))
  - do the same as the slide before, just use bigram table instead of 3-gram
- alpha(the man) --- compute this by computing p* of all bigrams (same as the previous slide)
  - then use the formula for computing a backoff factor
- p(eats)
  - there are a few ways to compute the last p(unigram)
  - see next slide

		p*
the man	5	0.3
man walks	3	0.4
man walked	4	0.5
the dog	3	0.1
dog walks	2	0.2
the bird	1	0.3
to the	2	0.22
the park	1	0.6

- p(eats) : there are a few ways to compute the last p(unigram)

1) according to Übung4:

- Es berechnet für jede Sprache L eine logarithmierte Wahrscheinlichkeit gemäß der Formel:

$$lp_L(a_1, ..., a_n) = \sum_{i=1}^{n} log\, p_L(a_i|a_{i-k}, ..., a_{i-1})$$

wobei $p_L(a_i|a_{i-k}, ..., a_{i-1})$ jeweils rekursiv wie folgt berechnet wird:

$$p_L(a_i|a_1^{i-1}) = p_L^*(a_i|a_1^{i-1}) + \alpha(a_1^{i-1}) p_L(a_i|a_2^{i-1}) \quad \text{für } 1 < i \leq n$$

$$p(a_1) = p_L^*(a_1) + \alpha() \frac{1}{1000}$$

p(eats) = p*(eats) + backoff ( ) * 1/ N

p(eats) = max [0, f(eats) – discount_0)] / f( )+ backoff ( ) * 1/ N

- f(eats) is computed using the unigram table
- N = freq of all unigrams (in Ü4, N=1000)

the	85
man	23
walked	14
dog	13
walks	32
bird	21
to	12
park	12

2) according to the example in slide 91

Beispiel:

p(Buch | das, rote) = p*(Buch | das, rote) + α(das, rote) (
p*(Buch | rote) + α(rote) (
p*(Buch)))

p*(Buch | das, rote) = (f(das, rote, Buch) - δ_2) / f(das, rote)

p*(Buch | rote) = (f(rote, Buch) - δ_1) / f(rote)

p*(Buch) = f(Buch) / N

It says, the last p(unigram) is simply p*(unigram) without the part „.... + backoff( ) * 1/ N „

As we reach p(unigram) version Übung4, the context will be empty.
What does the computation with empty context look like?

the	85
man	23
walked	14
dog	13
walks	32
bird	21
to	12
park	12

- p(eats) = max [0, f(eats) – discount_0)] / f( ) + backoff ( ) * 1 / N

- f(eats) : look at the unigram freq table
- discount_0 : look at the unigram freq table

$$\delta = \frac{N_1}{N_1 + 2N_2}$$

  - N1 = 0, N2= 0, so the discount will be 0, but we normally set it to some value that is not zero
  - In Übung4, we set the default value of discount to be 0.5
- f( ) -- > this is f (context)

$$f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1} w)$$

this is empy, so we have only w left

  - f( ) = f(the) + f(man) + f(walked) +....
  - it is simply the sum of the freq of every unigram
- backoff( )

every unigram

backoff( ) = 1 – [f(the)-discount / f( ) + f(man) –discount / f( ) + f(walked) –discount / f( ) + ...]

$$\alpha(C) = 1 - \sum_{w:f(C,w)>0} \frac{f(C,w) - \delta}{f(C)}$$

Interpolierte Backoff-Glättung

Beispiel:

$$p(\text{Buch} \mid \text{das, rote}) = p^*(\text{Buch} \mid \text{das, rote}) + \alpha(\text{das, rote}) \; (\\ p^*(\text{Buch} \mid \text{rote}) + \alpha(\text{rote}) \; (\\ p^*(\text{Buch})))$$

$$p^*(\text{Buch} \mid \text{das, rote}) = (f(\text{das, rote, Buch}) - \delta_2) \; / \; f(\text{das, rote})$$

$$p^*(\text{Buch} \mid \text{rote}) = (f(\text{rote, Buch}) - \delta_1) \; / \; f(\text{rote})$$

$$p^*(\text{Buch}) = f(\text{Buch}) \; / \; N$$

Example (word level language model):

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\, p(w_i|w_{i-k+1}^{i-1})$$

Auch hier gilt: $f(w_{i-k}^{i-1}) = \sum_w f(w_{i-k}^{i-1} w)$

the man eats	2
the man walks	2
the man walked	3
we go to	1
go to school	1
go to the	1
to the park	2
at the park	1

p(eats| the man) = max [0, f(the man eats) – discount_2)] / f(the man) + alpha(the man) * p(eats| ~~the~~ man)

p(eats | man) = max [0, f(man eats) – discount_1)] / f(man) + alpha(man) * p(eats)

p(eats) = max [0, f(eats) – discount_1)] / f( ) + alpha( ) * f(eats) / N

Summary: Sprachidentifizierung

- We want to identify the language of the text „Sprach“
- The language guesser (here: bigram Markov model) has to compute the prob p(L | T) for each language

p_eng(Sprach) = p(S|<s>,<s>) p(p|<s>,S) p(r|S,p) p(a|p,r) p(c|r,a) p(h|a,c) p(</s>|c,h)
p_de(Sprach) = p(S|<s>,<s>) p(p|<s>,S) p(r|S,p) p(a|p,r) p(c|r,a) p(h|a,c) p(</s>|c,h)
p_fr(Sprach) = p(S|<s>,<s>) p(p|<s>,S) p(r|S,p) p(a|p,r) p(c|r,a) p(h|a,c) p(</s>|c,h)

German

- We have learned several methods for computing the conditional probability, e.g. p(S|<s>,<s>) and p(r|S,p).
  - Maximum-likelihood-estimation (relative frequency)
    - p(w3 | w1, w2) = f(w1,w2,w3) / f(w1,w2)
  - Backoff smoothing

$$p(w_i|w_{i-k}^{i-1}) = \begin{cases} \frac{f(w_{i-k}^{i})-\delta_k}{f(w_{i-k}^{i-1})} & \text{falls } f(w_{i-k}^{i}) > 0 \\ \alpha(w_{i-k}^{i-1})\ p(w_i|w_{i-k+1}^{i-1}) & \text{sonst} \end{cases}$$

  - Backoff smoothing with interpolation (this works best and is used in Übung4)

$$p(w_i|w_{i-k}^{i-1}) = \frac{\max(0, f(w_{i-k}^{i}) - \delta_k)}{f(w_{i-k}^{i-1})} + \alpha(w_{i-k}^{i-1})\ p(w_i|w_{i-k+1}^{i-1})$$