1. Ziel von Wortbedeutungsdesambiguierung ? Welche Input bekommt der Desambigierer und was gibt er als Output/Ergebnis aus?

2. Nenne ein Bespiel von Input and Output.

1. Ziel von Wortbedeutungsdesambiguierung ?

> Für ein mehrdeutiges Wort in einem Satz, die richtige Wortbedeutung zu bestimmen.

Welche Input bekommt der Desambigierer und was gibt er als Output/Ergebnis aus?

> Input: ein Text in dem der zu desambiguierendem Wort vorkommt.
Output: die wahrscheinlichste Wortbedeutung

2. Nenne ein Bespiel von Input and Output.
> Input : “Ich setze mich auf die Bank” , das zu desambiguierende Wort = “Bank”
> Ouput: Bank = Bank zum Sitzen

Desambiguierung

Die Anwendung des Naïve Bayes-Klassifikators ist einfach.

Für ein gegebenes mehrdeutiges Wort w im Kontext der Wörter $w_1 ... w_n$

- berechnet man das Produkt $p(s) \prod_{i=1}^{n} p(w_i|s)$ für jede Bedeutung s
- und wählt die Bedeutung mit dem höchsten Wert.

3. Wortbedeutungsdesambiguierung ist eine Klassifikation-Aufgabe. Was wollen wir klassifizieren?

4. Was ist die Grundidee von einem statatischen Model für Wortbedeutungsdesambiguierung (in unserer Vorlesung)? (Welche Informationen über das zu disambigierende Wort werden verwendet um das Ergebnis des Models zu berechnen?)

Antwort : context wörter, Inhaltswörter

keine Antwort: Bag of Words model, alle substantive Wörter ohne Konjunktionen

Welche Annahme macht Naive Bayes Model : Bag of Words model im Kontext. aber die Kontextwörter (Positionen) sind nicht abhängig voneinander

3. Wortbedeutungsdesambiguierung ist eine Klassifikation-Aufgabe. Was wollen wir klassifizieren?

- Classify the meaning of a word from many meanings (multi class classification).
- Given a word with many possible senses, the classifier the word to the correct sense.

4. Was ist die Grundidee von einem statatischen Model für Wortbedeutungsdesambiguierung (in unserer Vorlesung)? (= welche Annahme macht man über die Bedeutung eines Wortes? / Welche Informationen über das zu disambigierende Wort werden verwendet um das Ergebnis des Models zu berechnen?)

- Context words (neighbouring words of ambigue word) can tells about the word meaning. E.g. if the ambigue word mostly tagged with meaning_1 when it occur with a particular group of context words, then the model should classify this word to meaning_1, when it occurs with these context words.

Formel von einem statistischen Wortbedeutungsdesambiguierer?

s^ = argmax_s p(s| context words)

s = best sense
s = all senses of the word that we want to disambiguate
context words = for example, n previous and next n words of the word that we want to disambiguate

Formel von einem statistischen Wortartdesambiguierer

Wir wollen die wahrscheinlichste Bedeutung $\hat{s}$ des Wortes w im Kontext C bestimmen:

$$\begin{aligned} \hat{s} &= \arg\max_s p(s|C) \\ &= \arg\max_s \frac{p(s,C)}{p(C)} \\ &= \arg\max_s p(s,C) \end{aligned}$$

p(C) ist eine Konstante, die keinen Einfluss auf das Ergebnis der argmax-Operation hat.

$$= \arg\max_{s} p(s, C)$$

Wie wird p(s, C) definiert und welche Annahme wurde gemacht?

Wie wird p(s, C) definiert und welche Annahme wurde gemacht?

Der Kontext C wird durch eine Menge von Kontextwörtern $= w_1 ... w_n$ (z.B. die 100 nächsten Inhaltswörter) repräsentiert.

$$\begin{aligned} p(s, C) &= p(s, w_1 ... w_n) \\ &= p(s) \prod_{i=1}^{n} p(w_i | s, w_1 ... w_{i-1}) \end{aligned}$$

Zur Vereinfachung des Modelles wird angenommen, dass die Kontextwörter statistisch unabhängig sind, wenn die Bedeutung s gegeben ist ("Bag of Words"-Modell):

$$p(s, C) = p(s) \prod_{i=1}^{n} p(w_i | s)$$

Formel für Naïve Bayes-Klassifikator ? (Erklären auch was die Variablen bedeute`

$$\hat{s} = \arg\max_s p(s) \prod_{i=1}^{n} p(w_i|s)$$

Welcher Teil heißt Apriori-Wk?

> p(s)

Ein Beispiel
p(s) = p(Bank= Sitzplatz) , p(Bank = Geldinstitut)
p(s | w1,w2,w3) = p(s) p(w1 |s) p(w2 | s) p(w3|s)

Ich setze mich auf die **Bank** und trinke Cola

p(sense1|w1,w2,w3)=p(sense1) **Π p(w1,w2,w3|sense1) – nicht richtig**

Product_i=1 bis n von p(wi |s) = p(w1|s) * p(w2| s) * p(w3 | s)

Text: Der Hahn am Waschbecken im Badezimmer tropft.

Wir wollen “Hahn” disambigieren zwischen “Hahn=Tier” und “Hahn=Wasserhahn”

Was muss das Model konkret berechnen um s^ zu bekommen? (Was wäre s, und w_i ?)

$$\hat{s} = \arg\max_{s} p(s) \prod_{i=1}^{n} p(w_i|s)$$

Text: Der Hahn am Waschbecken im Badezimmer tropft.

Wir wollen “Hahn” disambigieren zwischen “Hahn=Tier” und “Hahn=Wasserhahn”

Was müssen wir berechnen um s^ zu bekommen?

$$p(s, C) = p(s) \prod_{i=1}^{n} p(w_i|s)$$

Text: Der Hahn am Waschbecken im Badezimmer tropft.

Klasse: Hahn1 (=Wasserhahn)

Wahrscheinlichkeit: p(Hahn1, Waschbecken, Badezimmer, tropft) =

p(Hahn1) p(Waschbecken|Hahn1) p(Badezimmer|Hahn1) p(tropft|Hahn1)

$$\hat{s} = \arg\max_{s} p(s) \prod_{i=1}^{n} p(w_i|s)$$

Wie werden p(s) und p(wi | s) geschätzt? (Schrieb Formel und erklären wofür stehen die Variablen s^, s, wi, n)

p(s) = …..
p(wi | s) = …..

Wie sieht die Training-Daten aus ? Welche Annotationen brauchen wir?

Parameterschätzung

Um einen Wortbedeutungsdesambiguierer für das Wort w zu erstellen, braucht man ein Korpus, in dem jedes Vorkommen des Wortes w manuell mit seiner korrekten Bedeutung annotiert wurde.

Man zählt die Bedeutungen und berechnet die Apriori-Wahrscheinlichkeit:

$$p(s) = \frac{f(s)}{\sum_{s'} f(s')}$$

Dann zählt man alle Wörter w', die im Kontext der Bedeutung s des Wortes w auftauchen und berechnet:

$$p(w'|s) = \frac{f(w', s)}{\sum_{w''} f(w'', s)}$$

Training data with annotation.

Ich gehe in die **Bank**(Bank= Geldinstitut) und Geld abzuheben.
Ich setze mich auf eine **Bank** (Bank = Sitzplatz) unter einem Baum.
Die Bank ist nicht mehr offen, deswegen kann ich nicht zur **Bank** (Bank= Geldinstitut) gehen.
Eine Frau setzte sich auf die **Bank** (Bank = Sitzplatz) und liest ein Buch.

Training data with annotation.

Ich gehe in die **Bank**(Geldinstitut) und Geld abzuheben.
Ich setze mich auf eine **Bank** (Sitzplatz) unter einem Baum.
Die **Bank** (Geldinstitut) ist nicht mehr offen, deswegen kann ich nicht zur **Bank** (Geldinstitut) gehen.
Eine Frau setzte sich auf die **Bank** (Sitzplatz) und liest ein Buch.
Die **Bank**(Geldinstitut) ist zu.

$$p(s) = \frac{f(s)}{\sum_{s'} f(s')}$$

We want to disambigutate “Bank”. Apply the formula to both senses of Bank and calculate f(s) and all f(s’)

Training data with annotation.

Ich gehe in die **Bank**(Geldinstitut) und Geld abzuheben.
Ich setze mich auf eine **Bank** (Sitzplatz) unter einem Baum.
Die **Bank** (Geldinstitut) ist nicht mehr offen, deswegen kann ich nicht zur **Bank** (Geldinstitut) gehen.
Eine Frau setzte sich auf die **Bank** (Sitzplatz) und liest ein Buch.
Die **Bank**(Geldinstitut) ist zu.

$$p(s) = \frac{f(s)}{\sum_{s'} f(s')}$$

We want to disambigutate “Bank”. Apply the formula to both senses of Bank and calculate f(s) and all f(s’)

p(Geldinstitut) = f(Geldinstitute) / f(Geldinstitute) + f(Sitzplatz)
p(Sitzplatz) = f(Sitzplatz) / f(Geldinstitute) + f(Sitzplatz)
f(Geldinstitut) = 4, f(Sitzplatz) = 2

Training data with annotation.

Ich gehe in die **Bank**(Geldinstitut) und Geld abzuheben.
Ich setze mich auf eine **Bank** (Sitzplatz) unter einem Baum.
Die **Bank** (Geldinstitut) ist nicht mehr offen, deswegen kann ich nicht zur **Bank** (Geldinstitut) gehen.
Eine Frau setzte sich auf die **Bank** (Sitzplatz) und liest ein Buch.
Die **Bank**(Geldinstitut) ist zu.

$$p(w'|s) = \frac{f(w', s)}{\sum_{w''} f(w'', s)}$$

New text: “Ich will ein Buch kaufen und zur **Bank** gehen, aber sie ist vermutlich nicht mehr offen”

(context words are all content words in text)

What are w’ ? How many of them do we have here?
Use the formula on this text for all w’ (with s = Geldinstitute)

Training data with annotation.

Ich gehe in die **Bank**(Geldinstitut) und Geld abzuheben.
Ich setze mich auf eine **Bank** (Sitzplatz) unter einem Baum.
Die **Bank** (Geldinstitut) ist nicht mehr offen, deswegen kann ich nicht zur **Bank** (Geldinstitut) gehen.
Eine Frau setzte sich auf die **Bank** (Sitzplatz) und liest ein Buch.
Die **Bank**(Geldinstitut) ist zu, deswegen kann ich kein Geld abheben.

$$p(w'|s) = \frac{f(w', s)}{\sum_{w''} f(w'', s)}$$

New text: New text: “Ich will ein Buch kaufen und zur **Bank** gehen, aber sie ist vermutlich nicht mehr offen”

(context words are all content words in text)

What are w’ ? How many of them do we have here?
Use the formula on this text for all w’ (with s = Geldinstitute)

w’ = context words = gehen, offen, Buch, kaufen
Bank = Geldinstitut : s1
p(w’, s) = p(gehen, s1) = f(gehen, s1) / f(gehen, s1) + f(offen, s1) + f(Buch, s1) + f(kaufen, s1)
p(w’, s) = p(offen, s1) = f(offen, s1) / f(gehen, s1) + f(offen, s1) + f(Buch, s1) + f(kaufen, s1)
p(w’, s) = p(Buch,s1) = ...
p(w’, s) = p(kaufen, s1) = ...

Training data with annotation.

Ich gehe in die **Bank**(Geldinstitut) und Geld abzuheben.
Ich setze mich auf eine **Bank** (Sitzplatz) unter einem Baum.
Die **Bank** (Geldinstitut) ist nicht mehr offen, deswegen kann ich nicht zur **Bank** (Geldinstitut) gehen.
Eine Frau setzte sich auf die **Bank** (Sitzplatz) und liest ein Buch.
Die **Bank**(Geldinstitut) ist zu, deswegen kann ich kein Geld abheben.

$$p(w'|s) = \frac{f(w', s)}{\sum_{w''} f(w'', s)}$$

New text: “Ich will ein Buch kaufen und zur **Bank** gehen, aber sie ist vermutlich nicht mehr offen”

f(gehen, Bank=Geldinstitut) = 2
f(offen, Bank=Geldinstitut) = 1
f(Buch, Bank=Geldinstitut) = 0
f(Kaufen, Bank=Geldinstitut) = 0

Assuming we have trained a model. How can we use it to compute the sense of “Schwester” in “Seine einzige Verwandte, die Schwester, liegt im Krankenhaus” ?

- What parameters are stored in the model?
- And which one do we use?
- What probabilities does the model compute? And how to determine the best meaning using this information?

Assuming we have trained a model. How can we use it to compute the sense of “Schwester” in “Seine einzige Verwandte, die Schwester, liegt im Krankenhaus” ?

- What parameters are stored in the model?
> all possible senses of the word
> p(sense1) , p(sense2), …
> p(context_word, sense) for all possible context found in the training corpus
- And which one do we use?

Seine einzige Verwandte, die **Schwester**, liegt im Krankenhaus

lemmatisierte Kontextwörter:
einzig, Verwandte, Krankenhaus, liegen

Wahrscheinlichkeiten:

Wort w	p(w|SchwesterV)	p(w|SchwesterK)
einzig	0.001	0.00015
Verwandte	0.02	0.0001
liegen	0.0005	0.002
Krankenhaus	0.0001	0.03

$p(\text{SchwesterV}) = p(\text{SchwesterK}) = 0.5$

Beispiel

Seine einzige Verwandte, die **Schwester**, liegt im Krankenhaus

lemmatisierte Kontextwörter:
einzig, Verwandte, Krankenhaus, liegen

Wahrscheinlichkeiten:

Wort w	p(w|SchwesterV)	p(w|SchwesterK)
einzig	0.001	0.00015
Verwandte	0.02	0.0001
liegen	0.0005	0.002
Krankenhaus	0.0001	0.03

$p(SchwesterV) = p(SchwesterK) = 0.5$

$p(SchwesterV, einzig, Verwandte, Krankenhaus, liegen) = 5e\text{-}13$
$p(SchwesterK, einzig, Verwandte, Krankenhaus, liegen) = 4.5e\text{-}13$

$\Rightarrow$ Wir wählen hier die Lesart *SchwesterV*.

How does a simple baseline predict word sense?

Baseline und Obergrenze

Wenn kein Vergleichssystem zur Verfügung steht, bietet sich die folgende Baseline-Methode für Vergleichszwecke an:

Baseline: Wähle immer die Lesart, die in den Trainingsdaten am häufigsten war.

Wenn diese Baseline nicht übertroffen wird, (was oft gar nicht so einfach ist), ist das Verfahren nutzlos.

Wenn man wissen möchte, wie viel "Luft" noch für Verbesserungen vorhanden ist, kann man die menschliche Leistung bei dieser Aufgabe als **Obergrenze** zum Vergleich heranziehen.

Aufgabe 6) Erklären Sie, wie eine Pseudowort-Evaluierung funktioniert und wofür man Sie verwenden kann. (3 Punkte)

Pseudowort-Evaluierung

- Wozu benutzen wir Pseudowort-Evaluierung?
- Schritte ?

Pseudowort-Evaluierung

- Wozu benutzen wir Pseudowort-Evaluierung?

    - Um den Ansatz (Naive Bayes Model) zu evaluieren
    - Vorteile: keine annotieretes Korpus nötig
    - Es kann das wirkliche Model nicht evaluieren
- Schritte ?
    - Model ist trainiert, um das wort “w1-w2” mit zwei Bedeutungen w1, w2 zu disambiguieren.

Pseudowort-Evaluierung

Die manuelle Annotation von Daten ist teuer.

Zur Evaluierung einer Desambiguierungsmethode kann man jedoch ein großes perfekt annotiertes Trainingskorpus erstellen, indem man zwei Wörter zu einem neuen mehrdeutigen "Pseudo"-Wort zusammenfasst:

Pseudowort-Evaluierung:

- Nimm ein großes Textkorpus.
- Wähle zwei Wörter w_1 und w_2 und ersetze alle Vorkommen von w_1 und w_2 durch das Pseudowort w_1-w_2.
- Behalte das ursprüngliche Wort als ,,Bedeutungs"-Annotation.
- Teile das Korpus in zwei Teile.
- Trainiere den Desambiguierer auf dem ersten Teil, zwischen den Bedeutungen w_1 und w_2 des Pseudowortes w_1-w_2 zu unterscheiden.
- Evaluiere den Desambiguierer auf dem zweiten Teil durch Vergleich der Klassifikatorausgabe mit dem Originalwort.

```
Die       ART      die
Wächter   NN       Wächter
ihrerseits         ADV      ihrerseits
sind      VAFIN    sein
nicht     PTKNEG   nicht
hart      ADJD     hart
,         $,       ,
sie       PPER     sie
haben     VAINF    haben
ein       ART      eine
Herz      NN       Herz
und       KON      und
sind      VAFIN    sein
nicht     PTKNEG   nicht
ohne      APPR     ohne
Mitgefühl          NN       Mitgefühl
.         $.       .
```

We want to extract the lemma and the POS-tag of each word.

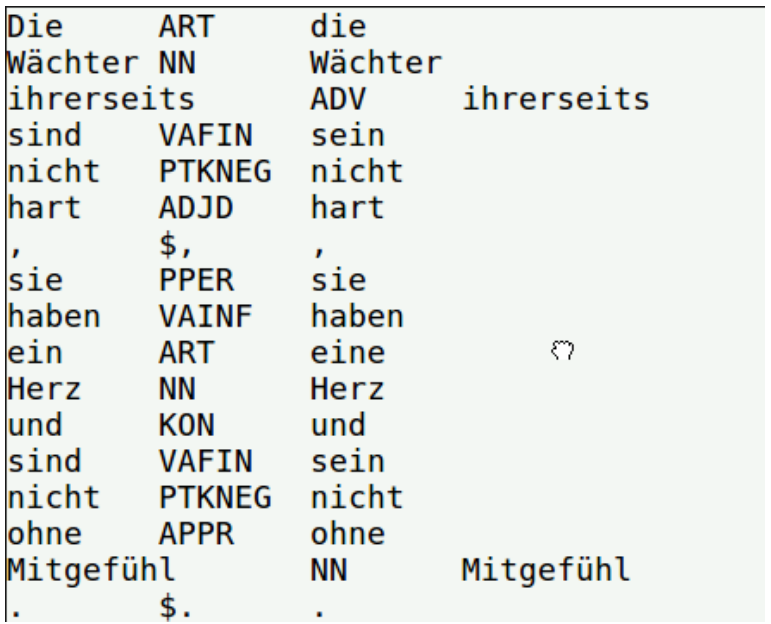
We want a list of the lemma and a list of its tag.

Read file and generate these lists.

```
Die     ART     die
Wächter NN      Wächter
ihrerseits      ADV     ihrerseits
sind    VAFIN   sein
nicht   PTKNEG  nicht
hart    ADJD    hart
,       $,      ,
sie     PPER    sie
haben   VAINF   haben
ein     ART     eine
Herz    NN      Herz
und     KON     und
sind    VAFIN   sein
nicht   PTKNEG  nicht
ohne    APPR    ohne
Mitgefühl       NN      Mitgefühl
.       $.      .
```

```python
# Trainingsdaten einlesen
print("reading data...", file=sys.stderr)
with open(sys.argv[1]) as file:
    lemmas = []
    tags = []
    for line in file:
        word, tag, lemma = line.strip().split('\t')
        lemmas.append(lemma)
        tags.append(tag)
```

We define the pseudoword: merged_words = ['Staat', 'Kind']
It means we generate a new word “Statt-Kind” with sense1 = Staat, sense2 = Kind
Now we want to calculate
f(sense1= Staat, context word) and f(sense2= Kind, context word) for all context words we found in the corpus.



```
Die       ART      die
Wächter   NN       Wächter
ihrerseits         ADV      ihrerseits
sind      VAFIN    sein
nicht     PTKNEG   nicht
hart      ADJD     hart
,         $,       ,
sie       PPER     sie
haben     VAINF    haben
ein       ART      eine
Herz      NN       Herz
und       KON      und
sind      VAFIN    sein
nicht     PTKNEG   nicht
ohne      APPR     ohne
Mitgefühl          NN       Mitgefühl
.         $.       .
```

How to get the context words?
- e.g. for “sense1 = Staat”
- iterate each word in the corpus, look for “Staat” (representing the word Staat-Kind with sense1) in the corpus. The context words are 50 previous and next content words. We count their occurences and store them in a dictionary.

Write code that computes
count[“Staat”] [context] = f(“Staat”, context)
count [“Kind”] [context] = f(“Kind”, context)

from lemmas and tags list

$$p(w|s) = \frac{f(s,w) + \alpha(s)\, p(w)}{f_1(s) + \alpha(s)}$$

:ind" oder "Staat" und $\alpha(s)$ ist die Zahl d
ler Nachbarschaft von s aufgetaucht sind.

$$p(w) = \frac{f_2(w)}{N}$$

/orthäufigkeit $f_2(w)$ wie folgt definiert ist:

$$f_2(w) = f(Staat, w) + f(Kind, w)$$

eit N sich so ergibt:

$$N = f_1(Staat) + f_1(Kind)$$

$$f_1(s) = \sum_w f(s,w)$$

```
# Häufigkeiten zählen
count  = defaultdict(lambda: defaultdict(int))
for i in range( len(tags) ):
    # Ist das aktuelle Wort im Pseudowort enthalten?
    if lemmas[i] in merged_words:
        # Kontextwörter extrahieren
        start = max(0, i-max_dist)
        end = min(len(tags), i+max_dist+1)
        # über alle Wörter bis Maximalabstand max_dist iterieren
        for k in range(start, end):
            # falls das Kontextwort ein relevantes Tag hat und
            # nicht mit dem aktuellem Wort identisch ist
            if tags[k] in relevant_tags and k != i:
                # Häufigkeit des Paares erhöhen
                count[lemmas[i]][lemmas[k]] += 1
```

How to create a dictionary ?

- my_dict = {}
- my_dict = defaultdict( )
- my_dict = defaultdict(int)

How to add a new element into the dict?

- my_dict[key] = value

How to access the element?

- for key, value in my_dict.items():
    print(key, value)

-for keys in my_dict: # or mydict.keys()
    print(keys) # list of keys

- for values in my_dict.values():
    print(values)

How to create a dict in dict?

- my_dict = defaultdict(**lambda:** defaultdict(int))
- my_dict = defaultdict(dict)
- my_dict = {key1: {key1_1: value, key1_2: value, ..},
  key2: {key2_1: value, key2_2: value,…}, …}

Assign new value

- my_dict[sense][context_word1] = 3
- my_dict[sense][context_word2] = 2

- my_dict[sense] = {context_word1: 3, context_word2: 2}

Access elements

- for sense, context_freq in my_dict.items():
  for context, freq in context_freq.items():
  print(sense, context, freq)

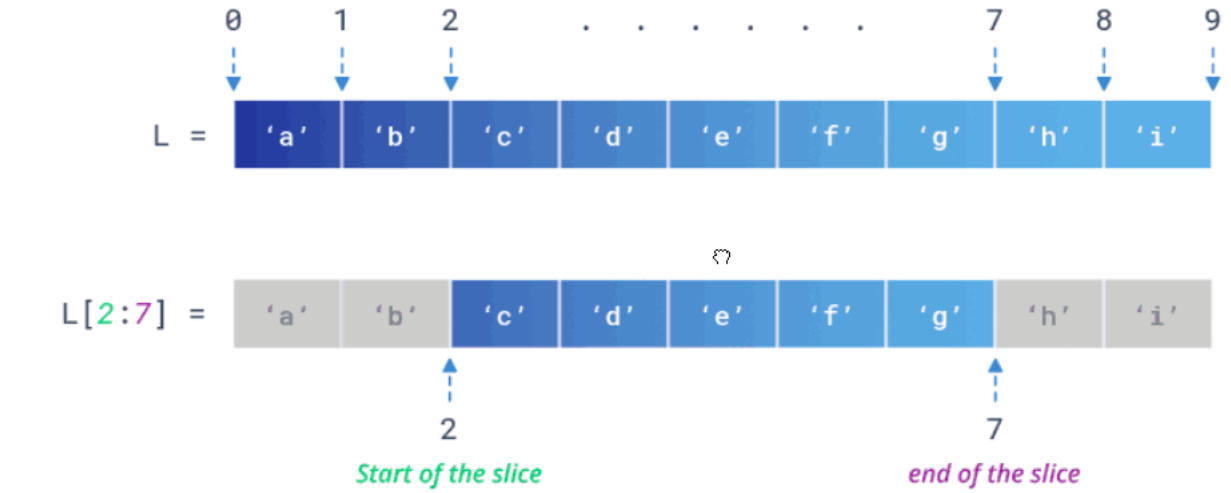
- for sense in my_dict:
  for context, freq in context_freq[sense].items():
  print(sense, context, freq)

```
from collections import defaultdict

target_dict = defaultdict(dict)
target_dict[key1][key2] = val
```

```
L[start:stop:step]
```

Start position *End position* *The increment*



https://www.learnbyexample.org/python-list-slicing/

Now we want to compute f(sense) and f(context)

Output: sense_count[sense] = f(sense)
cword_count[context] = f(context)
Using the dict “count” to compute this.

$$p(w|s) = \frac{f(s,w) + \alpha(s)\, p(w)}{f_1(s) + \alpha(s)}$$

ind” oder “Staat” und $\alpha(s)$ ist die Zahl d
ler Nachbarschaft von s aufgetaucht sind.

$$p(w) = \frac{f_2(w)}{N}$$

orthäufigkeit $f_2(w)$ wie folgt definiert ist:

$$f_2(w) = f(Staat, w) + f(Kind, w)$$

eit N sich so ergibt:

$$N = f_1(Staat) + f_1(Kind)$$

$$f_1(s) = \sum_w f(s,w)$$

Now we want to compute f(sense) and f(context)

Output: sense_count[sense] = f(sense)
cword_count[context] = f(context)
Using the dict “count” to compute this.

```
# Gesamthäufigkeiten der Bedeutungen und Kontextwörter berechnen
sense_count = defaultdict(int)
cword_count = defaultdict(int)
for sense in count:
    for word, freq in count[sense].items():
        sense_count[sense] += freq
        cword_count[word] += freq
```

$$p(w|s) = \frac{f(s,w) + \alpha(s)\, p(w)}{f_1(s) + \alpha(s)}$$

Compute this prob and store it in logprob = { }

Hier ist s entweder "Kind" oder "Staat" und $\alpha(s)$ ist die Zahl der *unterschiedlichen* Inhaltswörter, die in der Nachbarschaft von s aufgetaucht sind. $p(w)$ wird wie folgt geschätzt:

$$p(w) = \frac{f_2(w)}{N}$$

wobei die allgemeine Worthäufigkeit $f_2(w)$ wie folgt definiert ist:

$$f_2(w) = f(Staat, w) + f(Kind, w)$$

und die Gesamthäufigkeit N sich so ergibt:

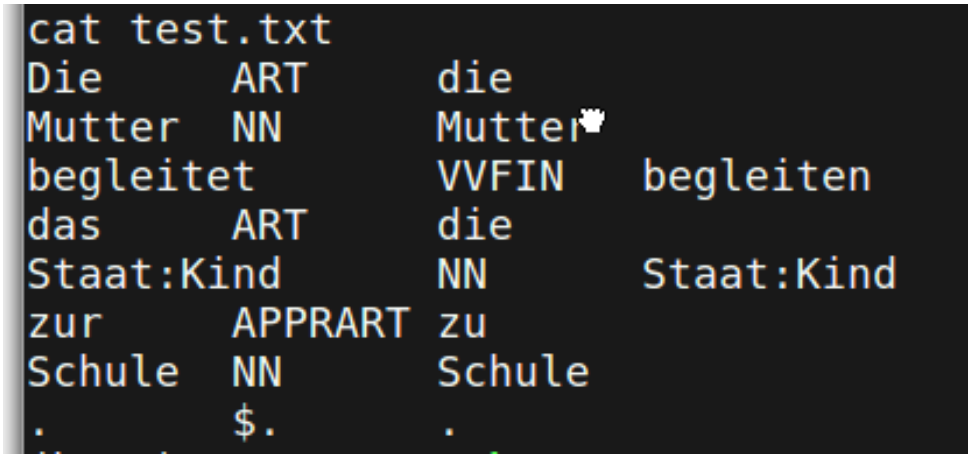
$$N = f_1(Staat) + f_1(Kind)$$

mit

$$f_1(s) = \sum_w f(s,w)$$

```
# Wahrscheinlichkeiten schätzen (Witten-Bell-Methode)
N = sum(sense_count.values())  # Gesamtzahl der Kontextwörter
logprob = {}  # logarithmierte Wahrscheinlichkeiten der Kontextwörter
for sense in count:  # für alle Lesarten
    logprob[sense] = {}
    alpha = len(count[sense])  # Zahl der Types bei den Kontextwörtern von Lesart s
    for word, freq in cword_count.items():  # für alle Kontextwörter (insgesamt)
        # Apriori-Wortwahrscheinlichkeit des Wortes berechnen
        pp = freq / N
        # mit Witten-Bell geglättete Wahrscheinlichkeit berechnen
        p = (count[sense].get(word, 0.0) + alpha * pp)/(sense_count[sense] + alpha)
        logprob[sense][word] = math.log(p)
print("done", file=sys.stderr)
```

```
    # Modell in einer Datei speichern
    with open(sys.argv[2],'wt') as file:
        json.dump(logprob, file)
```



Evaluate the model using this sentence

Write code that reads this file and generate lemmas and tags list

```
cat test.txt
Die     ART     die
Mutter  NN      Mutter
begleitet       VVFIN   begleiten
das     ART     die
Staat:Kind      NN      Staat:Kind
zur     APPRART zu
Schule  NN      Schule
.       $.      .
```

Evaluate the model using this sentence

Write code that reads this file and generate lemmas and tags list

```
# zu desambiguierende Daten einlesen
with open(sys.argv[2]) as file:
    data = [line.strip().split('\t') for line in file]
    _, tags, lemmas = zip(*data)
```

Write code that opens the logprob from the trained model

Write code that opens the logprob from the trained model

```
# Modell aus Datei lesen
with open(sys.argv[1]) as file:
    logprob = json.load( file )
```

Write code to fine the best sense for “Staat:Kind” and print the output

Given the following variables
lemmas = list of all tokens in the input text
tags = list of the corresponding tags of each lemma
ambiguous_word = “Staat:Kind”
senses = ['Staat', 'Kind']
max_dist = max distance of the context words from the ambiguous word
relevant_tags = set(['ADJA', 'ADJD', 'ADV', 'NE', 'NN', 'VVFIN', 'VVINF', 'VVPP', 'VVIZU'])
logprob[sense][word] = p(sense,context word)

```
# Vorkommen ambiger Wörter bestimmen
for i in range(len(lemmas)):
    if lemmas[i] == ambiguous_word:
        # Kontextwörter extrahieren
        start = max(0, i-max_dist)
        end = min(len(tags), i+max_dist+1)
        logp = {sense:0.0 for sense in senses}
        for k in range(start, end):
            # logarithmierte Wahrscheinlichkeiten aller
            # Kontextwörter aufsummieren
            if tags[k] in relevant_tags and k != i:
                for s in senses:
                    if lemmas[k] in logprob[s]:
                        logp[s] += logprob[s][lemmas[k]]

        # argmax-Operation berechnen
        best_sense = max(logp.keys(), key=logp.get)

        # Position und beste Lesart ausgeben
        print('Zeile %d: %s' % (i, best_sense))
```

Aufgabe 1) Welche Wahrscheinlichkeitsformel verwendet ein Naive-Bayes-Klassifikator? Welche Unabhängigkeitsannahmen macht der Klassifikator? Wie wählt der Klassifikator die Ergebnisklasse aus? (5 Punkte)

$\hat{s} = \operatorname{argmax}_s \; p(s) \prod_{i=1}^{n} p(w_i \mid s)$

Unabhängigkeitsannaheme: Bag of Words
Das die Reihenfolge der Wörter egal ist
solange sie im definierten Abstand (Fenster) auftauchen

contextwort wi hängt nur von der Bedeutung s ab und nicht andere Kontext wörter

die Bedeutungsklasse mit dem höchstenwert
Die Bedeutung mit höchsten p(s| C)

Aufgabe 2) Wie definiert ein Naive-Bayes-Modell die gemeinsame Wahrscheinlichkeit einer Klasse (bzw. Wortlesart) c und eines Textes (bzw. Wortkontextes) $w_1, .., w_n$? (2 Punkte)

Wie berechnen Sie konkret die Wahrscheinlichkeit der Klasse *Spam* für den Text: *"Diese Mail ist kein Spam !"* (2 Punkte)

$$p(s, C) = p(s) \prod_{i=1}^{n} p(w_i|s)$$

p(c, w1,…,wn) = p(c,w1^n) = p(c) p(w1|c) **p(w2|c,w1) p(w3|c, w1, w2)... – nicht korrekt**

= p(c) p(w1|c) p(w2|c) p(w3|c)

Muss es bei 2) laut Formel nicht heißen: p(Spam, Diese, Mail, ist, kein, Spam) = p()… ?
p(Spam, diese, Mail, ist,kein, Spam) = p(Spam) p(diese|Spam) p(Mail|Spam) p(ist|Spam) p(kein|Spam) p(Spam|Spam)

2) p(Spam| diese Mail ist kein Spam) = p(Spam) p(diese|Spam) p(Mail|Spam) p(ist|Spam) p(kein|Spam) p(Spam|Spam) --- nicht richtig

Aufgabe 8) Ein Naive-Bayes-Modell soll benutzt werden, um einen Zeitungsartikel einem Themengebiet (Politik, Wirtschaft etc.) zuzuweisen. Wie trainieren Sie das Modell? Welche Art von Daten benötigen Sie dazu? Wie berechnen Sie die wahrscheinlichste Kategorie eines Artikels (mit Formel)?

Anmerkung: Wir haben diese Anwendung nicht in der Vorlesung besprochen. Sie müssen hier Ihr Wissen auf eine neue Anwendung übertragen. (4 Punkte)

Wie trainieren wir das Modell?
berechnen beste_Thema = argmax_themen von p(thema| Artikel)

Daten? Korpus von Artikeln mit Themen

Wk von Kategorie eines Artikels =
p(Kategorie, Artikel) = p(Kat) Product_i=1,..i=n von p(wi|Kat)

Artikel = w1,…,wn

Aufgabe 5) Ein Naive-Bayes-Modell kann verwendet werden, um die wahrscheinlichste Bedeutung $\hat{s}$ eines ambigen Wortes gegeben die Folge seiner Nachbarwörter $w_1 ... w_n$ zu bestimmen. Erklären Sie für jeden Schritt der folgenden Herleitung des Modelles, wie er begründet werden kann. Welche Theoreme und Definitionen werden angewandt? Welche Unabhängigkeitsannahmen werden gemacht?

$$
\begin{aligned}
\hat{s} &= \arg\max_s p(s|w_1, ..., w_n) && (1)\\
&= \arg\max_s \frac{p(s)\ p(w_1, ..., w_n|s)}{p(w_1, ..., w_n)} && (2)\\
&= \arg\max_s p(s)\ p(w_1, ..., w_n|s) && (3)\\
&= \arg\max_s p(s) \prod_{i=1}^{n} p(w_i|s, w_1, ..., w_{i-1}) && (4)\\
&= \arg\max_s p(s) \prod_{i=1}^{n} p(w_i|s) && (5)\\
&= \arg\max_s \prod_{i=1}^{n} p(w_i|s) && (6)
\end{aligned}
$$

(4 Punkte)

Inhaltliche Wiederholung aus der Vorlesung

Naive Bayes-Modelle können zur Desambiguierung von Wortbedeutungen auf Basis der Kontextwörter verwendet werden. Die beste Bedeutung $\hat{s}$ eines Wortes w im Kontext der Wörter $context(w)$ wird nach der folgenden Formel berechnet:

$$\hat{s} = \arg \max_{s \in senses(w)} p(s|w) \prod_{c \in context(w)} p(c|s) \quad (1)$$

$senses(w)$: Menge der Bedeutungen des Wort-Tokens w
$context(w)$: Liste der Kontext-Wörter

Anmerkung: Dieses Modell soll nicht nur ein ambiges Wort w sondern mehrere ambige Wörter desambiguieren.

Aufgabe 1: Naive-Bayes-Parameterschätzung

Schreiben Sie ein Programm, welches die Wahrscheinlichkeiten $p(s|w)$ und $p(c|s)$ aus Trainingsdaten schätzt.

Es existiert bereits eine Funktion *extract_data*, welche das Trainingskorpus einliest und als eine Liste von Listen zurückliefert. Ihr Programm beginnt daher mit der Zeile:

```
my @data = extract_data();
```

`@data` enthält eine Liste von Wortvorkommen mit der korrekten Bedeutung und den Kontextwörtern.

`$w = $data[1][0]` liefert Ihnen das 2. ambige Wort (z.B. "Bank"), das aus dem Korpus extrahiert wurde
`$s = $data[3][1]` liefert Ihnen die Bedeutung des 4. ambigen Wortes des Korpus (z.B. "Sitzbank")
`$context = $data[3][2]` liefert Ihnen eine Referenz auf eine Liste mit den Wörtern im Kontext des 4. ambigen Wortes des Korpus (z.B. die Liste "(sitzen, Mann, vor, Baum)")

Sie sollen die folgenden Wahrscheinlichkeiten schätzen:

- die Wahrscheinlichkeit der Bedeutung s gegeben das Wort w gemäß der Formel

$$p(s|w) = \frac{freq(w,s)}{\sum_{s'} freq(w,s')} \tag{2}$$

- die Wahrscheinlichkeit des Kontextwortes c gegeben die Wortbedeutung s gemäß der Formel

$$p(c|s) = \frac{freq(s,c)}{\sum_{c'} freq(s,c')} \tag{3}$$

Schritte:

- Berechnen Sie, wie oft das ambige Wort w mit der Bedeutung s auftaucht (= $freq(w,s)$)
- Berechnen Sie, wie oft das Kontextwort c bei der Bedeutung s auftaucht (= $freq(s,c)$)
- Schätzen Sie $p(s|w)$ nach der Formel (2)
- Schätzen Sie $p(c|s)$ nach der Formel (3)

(6 Punkte)

Aufgabe 2: Naive-Bayes-Anwendung

Schreiben Sie nun ein Programm, welches das Naive Bayes-Modell aus einer Datei liest und auf Daten anwendet.

Ihr Programm beginnt mit

```
my(%wsprob,%scprob);
read_params(\%wsprob,\%scprob);
my @data = extract_data();
```

Danach sind die Parameter und die Eingabedaten eingelesen. (Die Funktionen `read_params` und `extract_data` müssen Sie nicht programmieren.)

`$wsprob{Schwester}{SCHWESTER1}` enthält die Wahrscheinlichkeit der Bedeutung „SCHWESTER1" des Wortes „Schwester" und entspricht $p(s|w)$

`$scprob{SCHWESTER1}{Krankenhaus}` enthält die Wahrscheinlichkeit des Wortes „Krankenhaus" im Kontext der Bedeutung „SCHWESTER1" und entspricht $\Rightarrow p(c|s)$

`@data` enthält die Eingabedaten, wobei dieses Mal die Bedeutung (z.B. `$data[3][1]`) undefiniert ist, da Sie diese ja berechnen sollen.

2. Wortbedeutungsdesambiguierung

Implementieren Sie einen Wortbedeutungsdesambiguierer auf Basis eines Naive-Bayes-Modelles. Auch hier schreiben Sie eine Funktion **desamb(word, tokens)**, welche das zu desambiguierende Wort und einen Text in Form einer Liste von Tokens als Argumente nimmt. Folgende Funktionen sind gegeben:

- Die Funktion **senses(word)** liefert die Liste der Bedeutungen des in "word" gespeicherten Wortes. (Bspw. könnte für das Wort *Bank* die Liste {*Bank1, Bank2, Bank3*} zurückgegeben werden.
- **wordProb(word, sense)** liefert die Wahrscheinlichkeit des Wortes "word" im Kontext der Bedeutung "sense" des zu desambiguierenden Wortes. Der Rückgabewert der Funktion ist immer größer als Null.
- **priorProb(sense)** liefert die Apriori-Wahrscheinlichkeit der Bedeutung "sense".

Sie sollen in der Tokenliste nach Vorkommen des zu desambiguierenden Wortes suchen und diese auf Basis der Kontextwörter mit maximalem Abstand 50 desambiguieren. Ausgegeben werden soll jeweils die Position des desambiguierten Wortes und die zugewiesene Bedeutung. Sie können die Ergebnisse entweder auf die Konsole ausgeben oder in einer Liste von Paaren (Position, Bedeutung) zurückgeben.