Programmierkurs Python II

Michaela Regneri
FR 4.7 Allgemeine Linguistik (Computerlinguistik)
Universität des Saarlandes

Sommersemester 2013

Übersicht

- Einführung zu maschinellem Lernen
- Sprachmodelle
- Textklassifizierung
- Word Sense Disambiguation, Teil 1
- Ein paar praktische Hinweise

Danke an
Dietrich Klakow, für die
Folien von SNLP.

Maschinelles Lernen

- Prinzip: lerne aus gesehenen Mustern von linguistischen Informationen -- irgendwas --
- Unterschiedliche Korpora für unterschiedliche Lernziele:
  - *Sprachmodelle*
  - Parsebäume
  - Übersetzungs-Modelle
  - Textkategorien
  - ...
- Grundsätzlich: je zahlreicher und je informativer die Daten, desto besser

Korpusanalyse

- Einfache Beispiele für Informationen (Konkordanzen)

Burschenschaft »Arminia« auch	**Studenten**	der Technischen Hochschule in
die Karlsruher Händel-Festspiele. Als	**Student**	in Halle und Göttingen von den
Girokonto für alle Azubis, Schüler,	**Studenten**	usw. für 0 DM Gebühren. Außerdem
und richtet sich an	**Studenten**	in den ersten Semestern.
Seminar richtet sich vorzugsweise an	**Studenten**	des Grundstudiums im 1. oder 3.
2000 (Programme for International	**Student**	Assessment) ergeben, dass die

der deutsch spricht und die	**Kinder antworten meistens**	auf schwedisch - aus
ausgeschaltet, die	**Kinder verschwinden meistens**	in die Häuser hinein, es
und Mineralstoffen kommen.	**Kinder reagieren meistens**	noch instinktiv auf
hieß es. « Die	**Kinder gehen meistens**	nur einmal die Woche in
Entwicklung hinzu. Die	**Kinder wachsen meistens**	in einem armen soziale
Schulalter. Top Symptome	**Kinder fallen meistens**	durch ein verändertes

Formales

- Wahrscheinlichkeit eines Wortes:

  $p(w = blah)$ — *die Wahrscheinlichkeit das Wort „blah“ anzutreffen*

- Wahrscheinlichkeit einer Wortfolge:

  $p(w1=der, w2=Hund, w3=bellt)$

  $p(der,Hund,bellt)$

  Wahrscheinlichkeit von „der Hund bellt“

- (durch seine „Geschichte“) bedingte Wahrscheinlichkeit eines Wortes:

  $p(w2 \mid w1)$ — *die Wahrscheinlichkeit für w_2, wenn davor w_1 gesehen wurde*

Eigenschaften von Wahrscheinlichkeiten

- $p(w) \geq 0$
- $\sum_{i=1}^{N} p(w_i) = 1$ (N = Anzahl aller Worte)
- $p(w_2|w_1) = \frac{p(w_1, w_2)}{p(w_1)}$ (Definition von bedingter Wahrscheinlichkeit)

Einfache Korpus-Analyse

- ist eine „Spaghetti Napoli" eine Kollokation?
- Prinzip: überprüfe, ob *Spaghetti* und *Napoli* öfter zusammen auftreten, als per Zufall zu erwarten wäre
- Ein numerisches Maß, das Korrelationen misst, ist z.B. „Pointwise Mutual Information" (*PMI*)
- auch eingesetzt für Pattern-Suche etc.

$$PMI = \log \frac{p(X,Y)}{p(X) \times p(Y)}$$

- *PMI = 0*: X ist **unabhängig** von Y
- *PMI > 0*: X tritt **häufiger** mit Y auf als man bei Unabhängigkeit erwartet

Einfache Korpus-Analyse - Beispiel

DeWac-Korpus:	1.4 x 109	Wörter
Spaghetti:	2448	Vorkommen
Napoli:	310	Vorkommen
Spaghetti Napoli:	12	Vorkommen

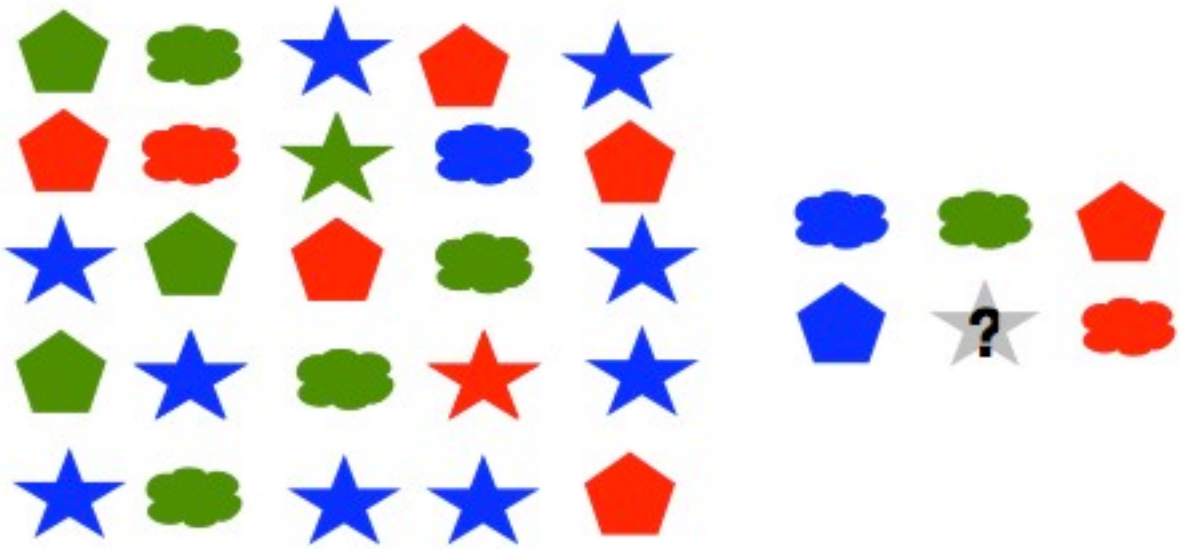
$$PMI = \log \frac{p(X,Y)}{p(X) \times p(Y)}$$

$p(Spaghetti, Napoli) \approx 12 / (1.4\ 10^9)$
$p(Napoli) \approx 310 / (1.4\ 10^9)$
$p(Spaghetti) \approx 2448\ (1.4\ 10^9)$

Wahrscheinlichkeiten abschätzen durch relative Häufigkeiten

$$PMI = log \frac{p(Spaghetti, Napoli)}{p(Spaghetti) \times p(Napoli)} = 4.35$$

Datenmangel (Sparse Data Problem), schematisch



Begriffe zu Sprachmodellen

- Sprache (Deutsch, Pascal,...)
- Wörter
  - z.B. deutsche Wörter
  - könnten auch Buchstaben, Silben etc. sein
- Die Sprache ist hier die Zusammenfassung gültiger Wort*folgen*
  - vgl. Automaten: elementare Einheiten = Alphabet
  - hier: elementare Einheiten = Wörter

Der Satz von Bayes

Theorem: $p(w_2|w_1) \times p(w_1) = p(w_1|w_2) \times p(w_2)$

Beweis:

$$p(w_2|w_1) \times p(w_1) = \frac{p(w_1, w_2)}{p(w_1)} \times p(w_1)$$

$$= p(w_1, w_2)$$

$$p(w_2|w_1) = \frac{p(w_1, w_2)}{p(w_1)}$$

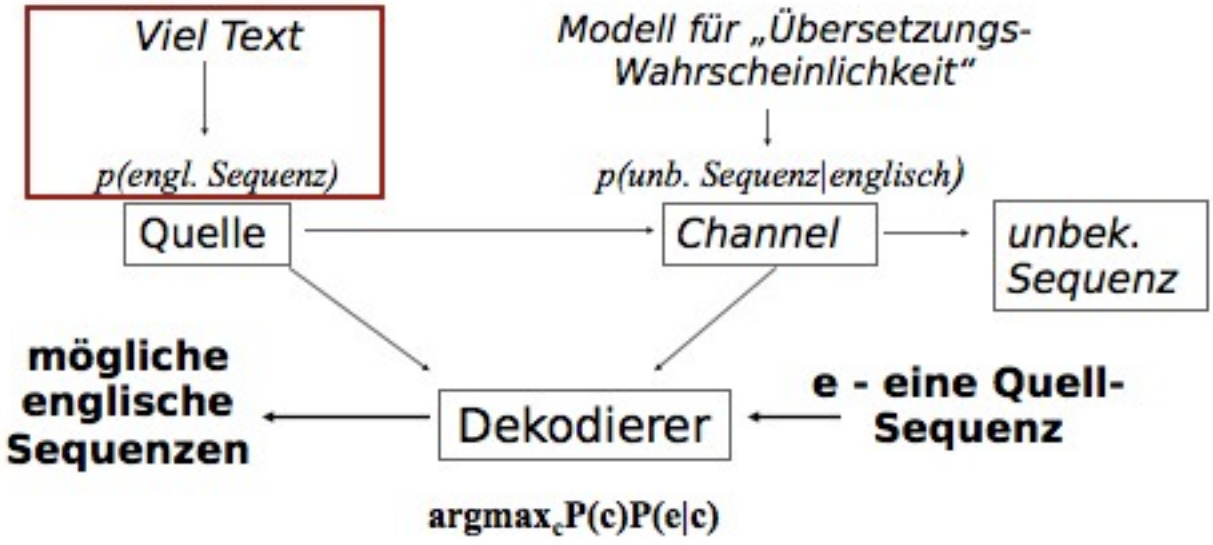
$$= \frac{p(w_1, 2_2)}{p(w_2)} \times p(w_2)$$

$$= p(w_1|w_2) \times p(w_2)$$

11

Das „Noisy Channel Model“

Ziel: ein Englischer Satz aus einer unbekannten Sequenz (deutsch, Sprachsignal,...)



Sprachmodelle

- Problem: woher bekommt man $p(w = wort)$ oder $p(w_1, w_2, w_3, ..?)$
- Sprachmodelle geben Auskunft darüber, mit welcher Wahrscheinlichkeit eine Wort-Sequenz in einer Sprache als gültige Sequenz auftaucht
  - damit kann man z.B. auch das wahrscheinlichste „nächste Wort“ bestimmen
  - klassische Anwendung:

`Der Schimmel galoppiert.`

```
The white horse galops.
The mold galops.
```


Sprachmodelle

$$p(w_1, w_2, w_3, ..., w_N)$$

Wahrscheinlichkeit einer Sequenz = ?

$$= p(w_N | w_1, ..., w_{N-1}) \times p(w_1, ..., w_{N-1})$$

$$p(w_{N-1} | w_1, ..., w_{N-2}) \times p(w_1, ..., w_{N-2})$$

$$\dots$$

Produkt aus allen Wahrscheinlichkeiten der einzelnen Worte im Kontext ihrer Vorgänger

$$\prod_{i=1}^{N} p(w_i | w_1, w_2, ...w_{i-1})$$

Sprachmodelle

$$\prod_{i=1}^{N} p(w_i|w_1, w_2, ...w_{i-1})$$

- Problem: Dieses Produkt wird fast immer 0
- Tentative Lösung: Die Markov-Annahme

  $$p(w_i|w_1, w_2, ..., w_{i-1}) = p(w_i|w_{i-M+1}, ..., w_{i-1})$$

- Unterschiedliche Sprachmodelle = unterschiedliches M:
  - Unigram: $p(w_i|w_1, w_2, ...w_{i-1}) = p(w_i)$
  - Bigram: $p(w_i|w_1, w_2, ...w_{i-1}) = p(w_i|w_{i-1})$
  - Trigram: $p(w_i|w_1, w_2, ...w_{i-1}) = p(w_i|w_{i-1}, w_{i-2})$
  - ...
- Das impliziert eine *Unabhängigkeitsannahme*

Unabhängigkeitsannahme

- Für viele Zwecke eingeführte Annahme / Definition:
  - ein Wort ist *unabhängig* von seinem Vorgänger, bzw. nur von seinen n-1 Vorgängern (n-Gramme)
  - dann: $p(w_1|w_2) = p(w_1)$
- Direkte Folge:
  - $p(w_2|w_1) = \frac{p(w_1, w_2)}{p(w_1)}$
  - $p(w_1, w_2) = p(w_1) * p(w_2)$
- Das kann man direkt auf n-Gramme übertragen (Wahrscheinlichkeit einer Sequenz = Produkt aller n-Gramm- Wahrscheinlichkeiten)

Maximum Likelihood Estimation (MLE)

- ein Sprachmodell basiert immer auf einem Korpus, das die Sprache repräsentiert
- ein Korpus ist immer nur ein Ausschnitt einer Sprache
- eine Möglichkeit, Wahrscheinlichkeiten anzunähern, ist MLE
- die Wahrscheinlichkeit eines Wortes (n-Gramms) ist seine relative Häufigkeit im Korpus $$p(w) = \frac{count(w)}{all\ words}$$
- Für N-Gramme: $p(w_1, w_2, ..., w_n) = \frac{count(w_1, w_1, w_2, ..., w_n)}{count(w_1, w_1, w_2, ..., w_{n-1})}$

Sprachmodell-Szenario

- Gegeben sei eine Liste von Bigrammen aus einem Korpus (Wahrscheinlichkeit für jedes Wortpaar)
- Die Wahrscheinlichkeit $p(w_1, w_2, w_3, ..., w_N)$ ist dann

$$\prod_{i=1}^{N} p(w_i|w_i - 1) = \prod_{i=1}^{N} \frac{p(w_{i-1}, w_i)}{p(w_{i-1})}$$

$$= \prod_{i=1}^{N} \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Die wahrscheinlichste Fortsetzung

- Gegeben ein Sprachmodell und eine Wortsequenz $(w_1, ..., w_i)$, generiere die wahrscheinlichste Fortsetzung

$$w_{i+1} = \max_w [p(w|w_1, ..., w_i)]$$

Wahrscheinlichkeit der gegebenen Sequenz * neues n-Gramm mit Wort aus der Sprache

- Für ein n-Gramm-Modell:

$$w_{i+1} = \max_w [p(w|w_{i+1-n}, ..., w_i) * \prod_{j=1}^{i} p(w_j|w_{j+1-n}, ..., w_{j-1})]$$

verändert sich mit der n-Gramm-Länge

- Das ist trivial äquivalent mit

Das wahrscheinlichste n-Gramm beginnend mit den n-1 letzten Worten der Sequenz

$$w_{i+1} = \max_w [p(w|w_{i+1-n}, ..., w_i)]$$

MLE und Datenmangel

- Trotz Markov-Annahme wird man immer ungesehene n-Gramme antreffen
- wenn ein n-Gramm in der Kette Wahrscheinlichkeit 0 hat, wird die Wahrscheinlichkeit der Sequenz 0
- das ist das grundsätzliche Problem bei maschinellem Lernen: Datenmangel (*Sparse Data Problem*)
  - Einordnungen passieren nur aufgrund gesehener Muster
  - ungesehene Muster kann man nicht adäquat einordnen

Smoothing

- Jede Wortfolge bekommt eine Wahrscheinlichkeit > 0
- Verschiedene grundsätzliche Taktiken:
  - Frequenzen manipulieren: Addiere eine kleine Zahl zu den Wortanzahlen (und verändere das Sprachmodell)
  - Wahrscheinlichkeiten manipulieren: schätze eine Wahrscheinlichkeit für unbekanntes, normalisiere die übrigen Wahrscheinlichkeiten (Summe =1)
  - Backing-Off: Ermittele n-Gramm-Wahrscheinlichkeit aus n-1-Grammen, ..., Unigrammen und „Zerogrammen" (alle Worte haben die gleiche Wahrscheinlichkeit)

Smoothing

- Einfaches Beispiel (n-Gramme, n=i): Häufigkeit + x

$$p(w_1, ..., w_i) = \frac{count(w_1, ..., w_i) + \lambda}{N + \beta * \lambda}$$

  - λ = kleine positive Zahl (z.B. 1 oder 0,5)
  - β = Anzahl der Bigramme im Test-Korpus

- Nachteil: Alle ungesehenen Sequenzen haben die gleiche Wahrscheinlichkeit

Text-Klassifizierung

- Gegeben:
  - eine Menge von Text-Kategorien (Spam/Nicht-Spam, Nachrichten-Themen, ...)
  - ein Text (oder mehrere)
- Aufgabe: Ordne den Text einer Kategorie zu
- Prinzip:
  - wähle verschiedene Kriterien (*features*), die bei der Texteinordnung helfen
  - trainiere ein Modell mit annotierten Texten (Features und Kategorie)
  - bestimme die wahrscheinlichste Kategorie

Text-Klassifizierung mit Naïve Bayes

$$c = \max_{c_i}[p(\chi|c_i) * p(c_i)]$$

- c: Kategorie, p(ci) = „A-Priori-Wahrscheinlichkeit“
- χ: sämtliche (!) features
- Auch hier Unabhängigkeitsannahme, dieses Mal zwischen den Features (N = Anzahl der Features)

$$p(\chi_1, \cdots \chi_N|c) = \prod_{i=i}^{N} p(\chi_i|c)$$

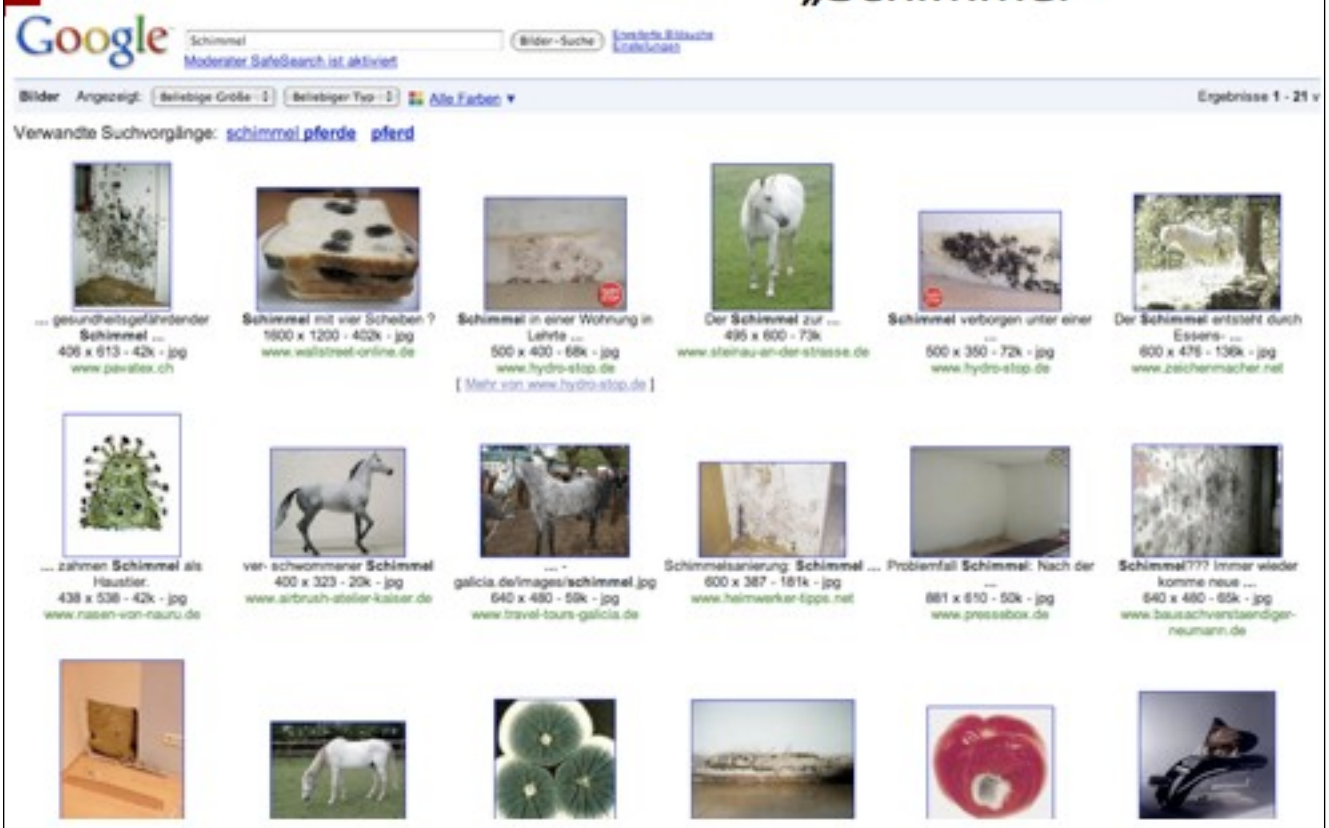
- Features könnten auch Wörter sein
- Bsp.: Sprachmodelle für Spam oder Nicht-Spam
- Klasse = die Kategorie, für die der Text wahrscheinlicher ist

Word Sense Disambiguation

- Aufgabe: entscheide für ein mehrdeutiges Wort, welche Bedeutung es im Kontext hat
- *Bedeutung* kann unterschiedliches bedeuten
  - Wörterbuch-Eintrag
  - WordNet-Nummer
  - Übersetzung
  - ...
- Wichtig für maschinelle Übersetzung, aber ggf. auch Suchmaschinen uvm.
- zu einem gewissen Grad in Sprachmodellen implizit gegeben, aber funktioniert oft nicht...

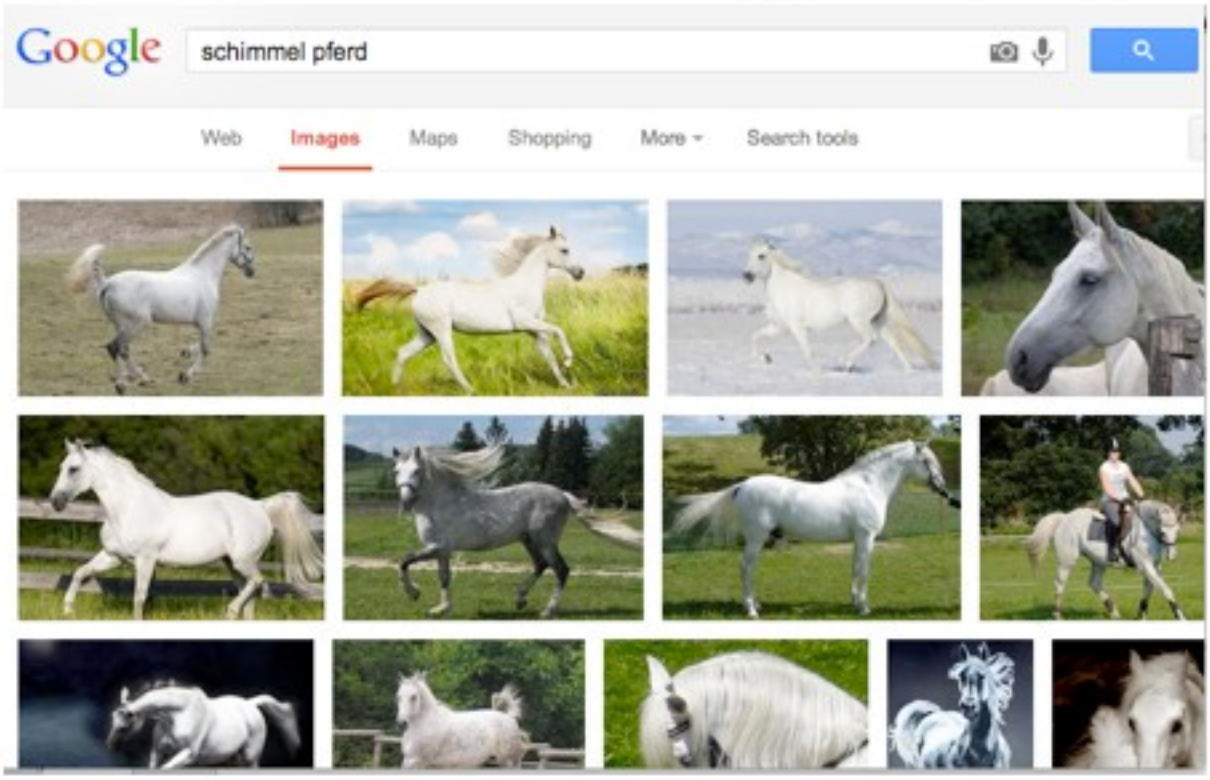
(Eine) Anwendung für WSD

„Schimmel“



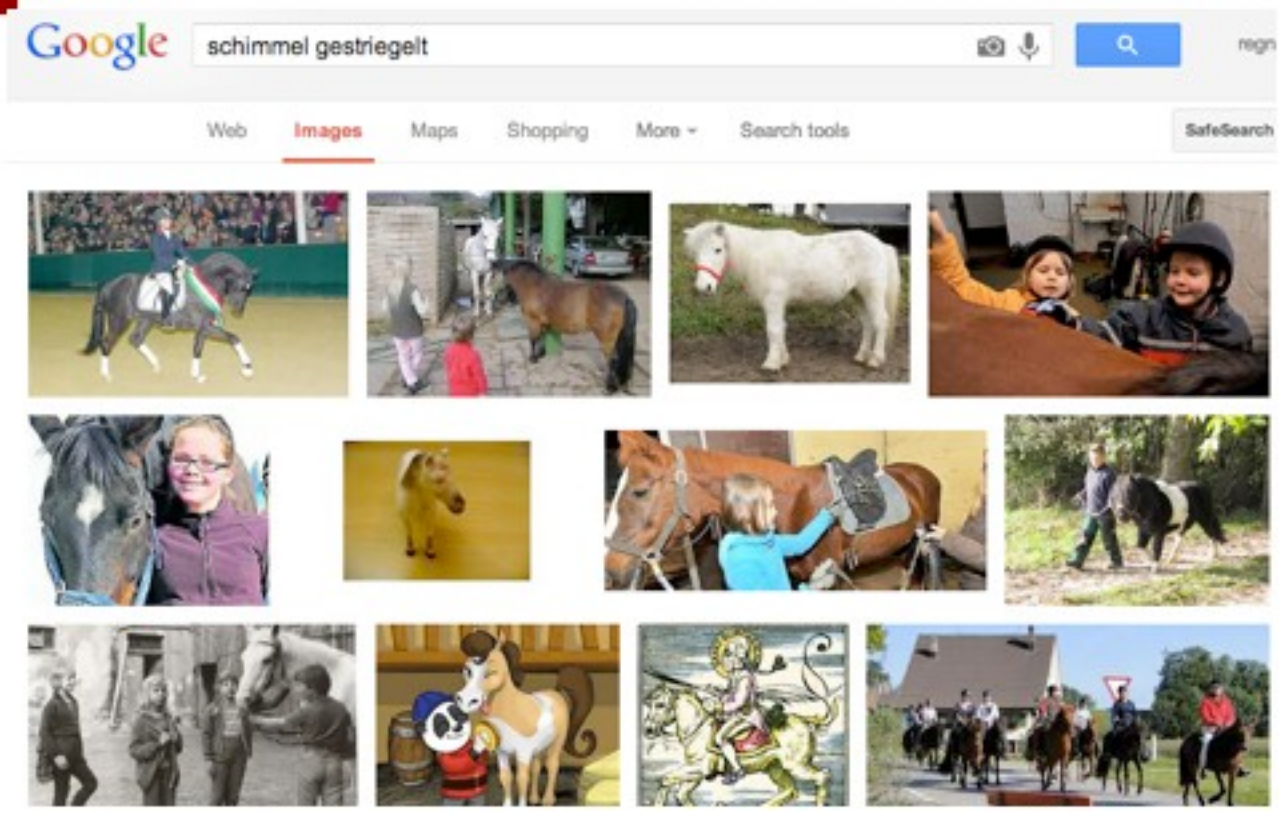
27

„Schimmel Pferd“

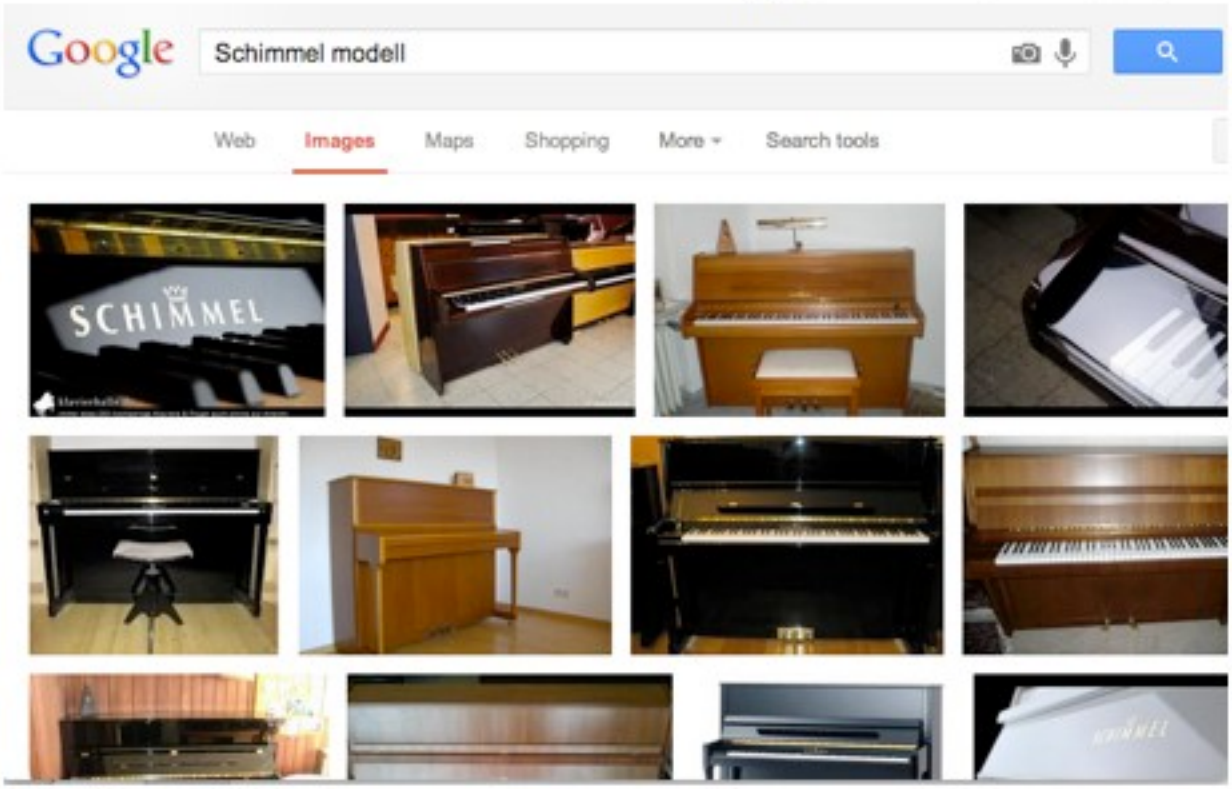


28

„Schimmel gestriegelt“



„Schimmel Modell“



Word Sense Disambiguation (WSD)

- Annahme für alle Algorithmen:
  - Ein Wort hat eine endliche Anzahl von klar abgegrenzten Bedeutungen
  - anhand des Kontextes ist de Bedeutung entscheidbar
- Das ist in der Realität nur bedingt so
  - Menschen haben für schwierige Fälle (viele Bedeutungen, die sich überlappen) auch nur ca. 65% Präzision
  - Die Aufgabe ist artifiziell; Menschen können Texte verstehen, ohne Worten Nummern zuzuordnen

Word Sense Disambiguation

- Annahmen von Yarowsky (1992,1993):
  - *One sense per discourse*
    Mehrere Vorkommen des gleichen Wortes in einem Text haben die gleiche Bedeutung
  - *One sense per collocation*
    Der Sinn eines Wortes kann durch syntaktische Konstellationen mit den benachbarten Wörtern bestimmt werden (≈ im wesentlichen disrtibutionelle Hypothese)

WSD als Klassifizierung

- Angenommene Trainingsdaten: Ein Korpus, in dem das Wort in Frage mit seinem korrekten Sinn annotiert ist
- WSD kann so als Klassifizierungs-Aufgabe betrachtet werden
  - gegeben eine Menge von Wort-Auftreten
  - ordne jedem Wort-Auftreten den richtigen Wortsinn zu
- Dafür kann man z.B. auch den Naïve Bayes-Algorithmus verwenden

Naïve Bayes

- Die Features könnten im einfachsten Fall Worte im Kontext sein (Variable: Kontextgröße)
- Die einzelnen Worte im Kontext würden dann als unabhängig voneinander betrachtet (bag of words)
- Aber auch beliebige andere Features sind denkbar, je nach vorhandenen Trainingsdaten (POS in der Umgebung, Syntax, Wordsinne in der Umgebung...)

Naïve Bayes: Vorgehen

D1
```
Klavier... Pianist...
Flügel... spielen...
Konzert... Kinder...
```

D2
```
Lebensmittel...
Pilz... weiß...
Kühlschrank... Brot...
schwarz...
```

D3
```
galoppiert... Hof...
Pferde... spielen...
Kinder... weiß...
```

Wort	*Pferd*	*Pilz*	*Flügel*	Σ
Klavier	0	0	10	10
spielen	1	0	3	4
Pilz	0	10	0	10
weiß	4	7	4	15
Pferde	10	0	0	10
Flügel	2	0	9	11
Brot	4	9	0	13
Hof	9	0	1	10
schwarz	1	8	4	13

Naïve Bayes: Vorgehen

$$p(\chi_1, \cdots \chi_N | c) = \prod_{i=i}^{N} p(\chi_i | c)$$

Wort	*Pferd*	*Pilz*	*Flügel*	Σ
Klavier	0	0	10	10
spielen	1	0	3	4
Pilz	0	10	0	10
weiß	4	7	4	15
Pferde	10	0	0	10
Flügel	2	0	9	11
Brot	4	9	0	13
Hof	9	0	1	10
schwarz	1	8	4	13
Σ	31	34	31	96

$$p(weiß|Pf) = \frac{p(weiß \wedge Pf)}{p(Pf)}$$

$$p(weiß|Pf) = \frac{4/96}{apriori(Pf)}$$

apriori(x) ändert sich mit der Häufigkeit der Wortbedeutung

Bigramme in Python, Variante 1

```
import re


def makeBigrams(file):
    store = dict()
    with open(file) as f:
       r = [w for w in re.split('[\W_]', f.read()) if w != '']
       for bigram in [r[i] + ' ' + r[i+1] for i in range(len(r)-1)]:
          clean = [word for word in re.split('\W', bigram) if word != '']
          bi = clean[0] + ' ' + clean[1]
          if bi not in store:
             store[bi] = 1
          else:
             store[bi] = store[bi] + 1
    return store
```


Bigramme in Python, Variante 2

Für größere Sprachmodelle wird der Speicher überlaufen -> shelve-Objekte! (Siehe Kurs I, Projekt)

```
def makeBigrams2(file):
    store = dict()
    with open(file) as f:
       r = [w for w in re.split('[\W_]', f.read()) if w != '']
       for bigram in [r[i] + ' ' + r[i+1] for i in range(len(r)-1)]:
           clean = [word for word in re.split('[\W_]', bigram) if word!='']
           first, second = clean[0] , clean[1]
           if first not in store:
               store[first] = {second:1}
           elif second in store[first]:
               store[first][second] = store[first][second] + 1
           else:
               store[first][second] = 1
    return store
```

Bigramme ohne Python

```
bsh$ tr -sc 'a-zA-Z' '\012' < dieTextdatei > tokens1
bsh$ tail +2  tokens1 > tokens2
bsh$ paste tokens1 tokens2 | sort | uniq -c | sort
```

- **tr** ersetzt Zeichen
  - c = „complement"
  - s = „squeeze"
- **tail** zeigt das Endstück einer Datei (ab Position x)
- **paste** positioniert 2 Texte als Spalten nebeneinander
- **uniq** löscht aufeinander folgende Dopplungen
  - c fügt deren Anzahl hinzu
- **sort** sortiert (für Optionen siehe *man sort*)

Project Gutenberg

```
(Copyright etc.)
Language: English

Character set encoding: ASCII

*** START OF THE PROJECT GUTENBERG EBOOK NOTHING TO EAT ***

Produced by Charles Aldarondo, Charles Franks
and the Online Distributed Proofreading Team


[Illustration: "PROTESTING, EXCUSING, AND SWEARING A VOW,
SHE'D NOTHING WORTH EATING TO GIVE US FOR DINNER."]


NOTHING TO EAT.
Illustrated.
NOT
By the Author of "Nothing to Wear"

"I'll nibble a little at what I have got."
```

Zusammenfassung

- Kurze Einführung zu maschinellem Lernen
- Sprachmodelle
- einfacher Naive-Bayes-Klassifizierer
- Praktisches (zur Übung und sonst)