

Sprachverarbeitung: Übung 23

Statistische Sprachmodellierung

Aufgabe 1: N-Gram-Sprachmodelle einfacher Sprachen

Bestimmen Sie das angemessene N-Gram-Sprachmodell für die folgenden Sprachen:

- Das Vokabular der Sprache L_1 besteht aus den Wörtern “a”, “b”, “c” und “d”. Korrekte Sätze dieser Sprache sind alle möglichen Wortfolgen, bei denen nie zwei gleiche Wörter unmittelbar nacheinander folgen. Bei gegebener Vorgeschichte sind alle erlaubten Folgewörter gleich wahrscheinlich (d.h. abgesehen von der erwähnten Einschränkung).
- Die Sprache L_2 hat das Vokabular “x” und “y”, wobei “x” anderthalb mal so häufig ist wie “y”. Zudem treten in dieser Sprache die Wortpaare mit zwei ungleichen Wörtern dreimal so häufig auf wie Wortpaare mit zwei gleichen Wörtern.

Lösungshinweis: Nehmen Sie an, dass Sie von der Sprache L_2 eine sehr lange Wortsequenz haben, aus der Sie das N-Gram-Sprachmodell ermitteln wollen. Überlegen Sie, was sich daraus für eine Bedingung für die Wortpaare mit zwei ungleichen Wörtern ergibt. Diese Bedingung liefert Ihnen eine weitere Gleichung, die Sie zum Ermitteln der Wahrscheinlichkeiten des Sprachmodells benötigen. Um schliesslich das aufgestellte Gleichungssystem zu lösen, können Sie die Matlab-Funktion `solve` verwenden. Hier ein einfaches Beispiel für zwei Gleichungen mit zwei Unbekannten:

```
clear all;
syms x y;
eqns = [y == x + 1, ...
        y == 1 - x];
S = solve(eqns, [x y]);
disp(['x = ' char(S.x)])
disp(['y = ' char(S.y)])
```

Achtung: Das obige Beispiel benötigt die Symbolic Math Toolbox Version 6.1 oder höher. Sie können mit dem Matlab-Skript `example_with_function_solve.m` feststellen, welche Version dieser Toolbox Sie benutzen. In diesem Matlab-Skript, das im Directory `Uebung23/Gegebenes` zu finden ist, sehen Sie auch, wie das Beispiel für ältere Versionen geschrieben werden muss.

Aufgabe 2: Perplexität einfacher Sprachen

Bestimmen Sie die Perplexität der Sprachen L_1 und L_2 von Aufgabe 1.

Aufgabe 3: Schätzen von N-Gram-Wahrscheinlichkeiten

In dieser Aufgabe geht es darum, N-Gram-Wahrscheinlichkeiten aus gegebenen Trainingsdaten zu schätzen. Das Vokabular V besteht hier aus den Zahlen von 1 bis $|V|$. Die Trainingsdaten bestehen aus einer Menge von endlichen Sequenzen dieser Vokabularelemente, beispielsweise:

```
[1 2 1 1]
[2 1]
```

In diesem einfachen Beispiel umfassen die Trainingsdaten nur 2 Sequenzen und es kommen nur die Vokabularelemente 1 und 2 vor.

Für Unigrams genügt es, die Auftretenswahrscheinlichkeiten für jedes Vokabularelement zu bestimmen. Für N-Grams mit bedingten Wahrscheinlichkeiten müssen zusätzlich Beginn und Ende der Sequenz behandelt werden (siehe Abschnitt 14.2.4.2 im Buch). Wir definieren dafür ein spezielles Element, das im Folgenden mit “BND” bezeichnet wird. Dieses Element soll den Index $|V|+1$ haben, im obigen Beispiel also 3. Das Symbol “BND” ist nicht in den Trainingssequenzen vorhanden, sondern muss während der Berechnung der Bigrams und Trigrams am Anfang und am Ende angefügt werden.¹

Schreiben Sie nun eine Funktion, welche aus derartigen Trainingsdaten die N-Gram-Wahrscheinlichkeiten ermittelt. Die Funktion soll die folgenden Argumente haben:

```
ngram = estim_ngram_probs(tr_data,vocsize,N,smval,dbg);
```

Die Argumente sind im Help der Funktion `estim_ngram_probs_frame` beschrieben. Am besten gehen Sie vom File `estim_ngram_probs_frame.m` aus, um Ihre Funktion zu schreiben. Sie finden es im Directory `Uebung23/Gegebenes/`.

Beginnen Sie mit dem einfachsten Fall, nämlich mit dem Unigram und testen Sie die realisierte Funktion mit der Funktion `ueb23_3(N)`, wobei zum Testen der Unigram-Schätzung das Argument N auf 1 zu setzen ist. Wenn die Tests erfolgreich sind, erweitern Sie die Funktion für Bigram und Trigram. Für die Tests werden in `ueb23_3` nur sehr wenig Trainingsdaten mit einem sehr kleinen Vokabular verwendet, sodass die Resultate gut von Hand nachvollzogen werden können. Wenn ein Test fehlschlägt, dann werden die dabei verwendeten Trainingsdaten und die N-Gram-Sollwerte angezeigt.

Aufgabe 4: Sprachidentifikation mittels N-Gram

Wenn Ihre Funktion `estim_ngram_probs` richtig funktioniert, dann können Sie sie nun mit der Funktion `ueb23_4` zur Sprachidentifikation einsetzen (siehe: `help ueb23_4`). Für einen eingegebenen Text² schätzt diese Funktion, ob dieser Text Deutsch, Englisch, Französisch oder Italienisch ist.

Die Schätzung beruht darauf, dass Buchstaben, Buchstabenpaare und Buchstabentripel in diesen vier Sprachen unterschiedlich häufig vorkommen. Die Funktion `ueb23_4` ermittelt deshalb

¹In der Vorlesung haben wir aus Gründen der Plausibilität die Hilfselemente für ein Bigram mit START und END bezeichnet. Naheliegenderweise können diese gleich bezeichnet werden. Welches gemeint ist, ist aus der Bezeichnung der bedingten Wahrscheinlichkeit ersichtlich. So ist mit $P(3|1)$ bzw. $P(1|3)$ im Falle einer Vokabulargröße von 2 die Wahrscheinlichkeit gemeint, dass das Vokabularelement 1 am Anfang bzw. am Ende einer Sequenz steht.

²Als Text für die Sprachidentifikation kann ein Satz oder auch nur ein Wort verwendet werden, wobei die deutschen Umlaute als “ae”, “oe” bzw. “ue” geschrieben und in französischen und italienischen Wörtern die Akzente weggelassen werden.

zuerst aus vorgegebenen Texten (siehe Directory `Uebung23/Gegebenes/ueb23_data/`) für jede Sprache ein N-Gram und berechnet sodann für den zu identifizierenden Text je die Kreuzentropie (siehe Formel (211) im Buch). Das N-Gram mit dem kleinsten Wert der Kreuzentropie gibt die Sprache an. Überlegen Sie, warum das so ist!

Falls Sie in der Aufgabe 3 die Funktion `estim_ngram_probs` nicht nur für die Schätzung eines Unigrams verwirklicht haben, sondern auch für das Bigram und das Trigram, dann können Sie hier nun deren Leistungsfähigkeit bei der Sprachidentifikation vergleichen.