Maschinelle Sprachverarbeitung

Übung

Aufgabe 2: Part-of-Speech Tagging mit Hidden-Markov-Modellen

Mario Sänger

Aufgabenstellung

- Implementation eines Hidden-Markov-Modell (HMM) zum Part-of-Speech Tagging
  - Hidden States=POS-Tags und Observable States=Wörter

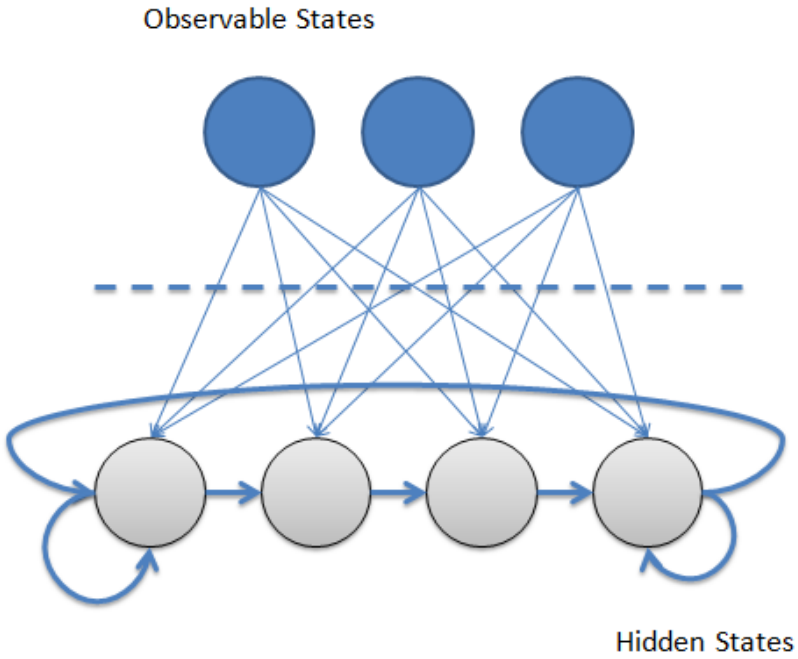


Bild: http://crsouza.com/2010/03/23/hidden-markov-models-in-c/

Aufgabenstellung

- Implementation eines Hidden-Markov-Modell (HMM) zum Part-of-Speech Tagging
  - Lernen eines Modells mittels der Maximum-Likelihood-Methode
  - Bereitstellung eines Trainingskorpus mit annotierten Sätzen
- Implementation eines Taggers basierend auf dem gelernten Modell (Viterbi-Algorithmus)
- Durchführung einer 10-fach Kreuzvalidierung

Trainingskorpus

- Korpus beinhaltet annotierte, englischsprachige Sätze

  SBA/np works/vbz closely/rb with/in the/at principal/jjs property/nn disposal/nn installations/nns of/in the/at Federal/jj

  …

- Verwendung des Brown Tag Sets
  - Trainingsdaten enthalten alle möglichen Tags
  - Keine anderen / weiteren Modifikatoren
  - Einige Wörter beinhalten ein „/" => Letztes „/" bildet Separator

- Korpus ist verfügbar unter:
  - https://hu.berlin/ue_maschsp1718_ue2_korpus

Hinweise und Gestaltungsmöglichkeiten

- Multiplikation sehr kleiner Fließkommazahlen
  - Beachtet die numerische Genauigkeit!
- Übergangswahrscheinlichkeiten (transition probabilities)
  - Verwendung von 1-grams, 2-grams …
- Umgang mit ungesehenen Wörter?
  - Korpus ist “Standard Englisch”
  - Geeignetes Smoothing ist notwendig
- Startwahrscheinlichkeiten nicht vergessen!

Aufgabendetails

- Euer Programm unterstützt zwei Modi: Trainings- und Annotationsmodus

- Trainingsmodus:
  - Das Programm liest alle Dateien in „train_dir/“ und lernt ein Hidden-Markov-Modell mittels der Maximum-Likelihood-Methode
  - Speichern des Modells im Ausführungsverzeichnis („./“)

```
java -jar uebung2-groupX.jar train train_dir/
```

Aufgabendetails

- Annotationsmodus:
  - Programm lädt gelerntes Modell aus dem Ausführungsverzeichnis
  - Programm liest und taggt alle Dateien in „input_dir/"
  - Ausgabe erfolgt nach „output_dir/input-file-name"
  - Ausgabedatei hat das gleiche Format wie die Eingabedatei

```
java -jar uebung2-groupX.jar annotate \
     input_dir/ output_dir/
```

- Implementation des Programms in Java oder Scala!

Annotationsmodus

- Eingabe: input_dir/example1

  SBA/NA works/NA closely/NA with/NA the/NA principal/NA property/NA disposal/NA installations/NA of/NA the/NA Federal/NA.

  …

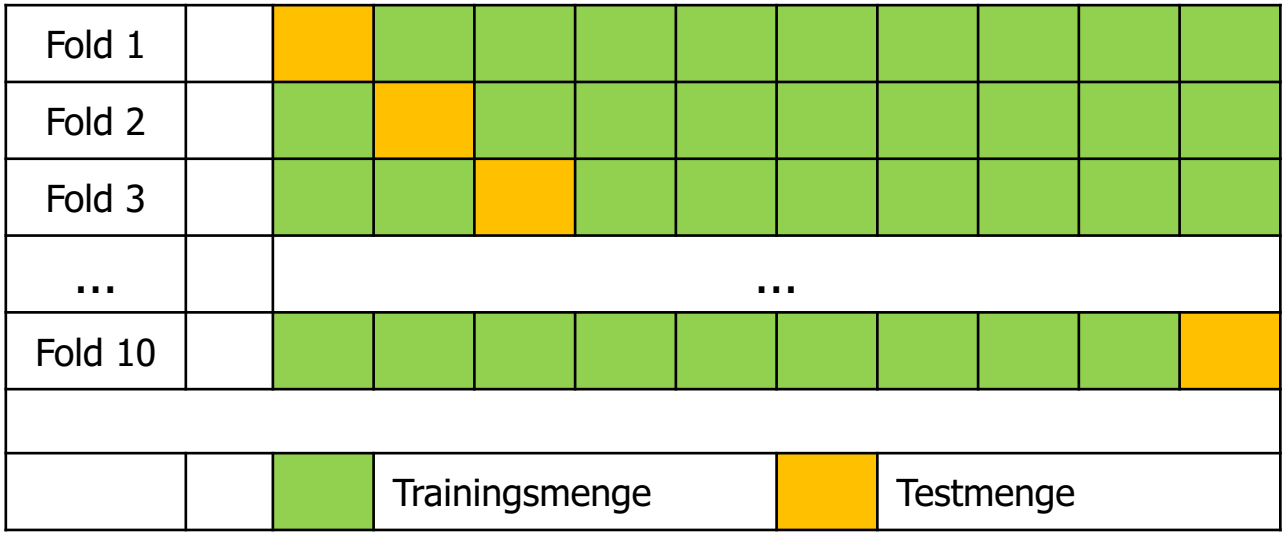
- Ausgabe: output_dir/example1

  SBA/np works/vbz closely/rb with/in the/at principal/jjs property/nn disposal/nn installations/nns of/in the/at Federal/jj

  …

10-fach Kreuzvalidierung

- Test des Modells mittels 10-fach Kreuzvalidierung
  - Aufteilung aller Trainingsinstanzen in 10 disjunkte Teilmengen
  - Nutze jede Teilmenge einmal zum Testen und den Rest jeweils als Trainingsmenge



- Modell muss mindestens 66% Genauigkeit erreichen!

Abgabe

- Abgabe eines ZIP-Archivs uebung2-gruppeX.zip
  - Archiv enthält ausführbare JAR und den Quellcode
  - Ergebnisse der 10-fach Kreuzvalidierung (inkl. Durchschnitt, Standardabweichung und Median)
  - Verwendete Aufteilung der Trainingsdaten (10 Dateien)

- Testet Euer JAR vor der Abgabe!
  - Ein Beispieldatei aus dem Evaluationssatz: https://hu.berlin/ue_maschsp1718_ue2_test

- Abgabe bis spätestens 03.12.2017, 23:59 Uhr über: https://hu.berlin/ue_maschsp1718_ue2

Wettbewerb

- Test der Modelle und Tagger auf ungesehenen Daten
  - Gute Lösungen müssen schnell und genau sein!
- Wettbewerb 1: Es gewinnst die Lösung mit der höchsten Genauigkeit
- Wettbewerb 2: Es gewinnt die schnellste Lösung mit einer Genauigkeit unter den Top 6
- Die drei besten Teams bekommen jeweils 3/2/1 Punkte

Fragen?