Aufgabenblatt 3 — 7.11.2019

Einführung in die Computerlinguistik (WS19/20) — Abgabe bis 16.11.2019

Aufgabe 1) Minimum-Edit Distance Simulation **Punkte: 5**

Nehmen Sie die folgenden Kosten für die verschiedenen Operationen an:

- Insert: 1
- Delete: 1
- Substitute: 2
- Match: 0

Berechnen Sie die Editierdistanz für das Wortpaar LISTEN - SILENT mit Hilfe des Minimum-Edit-Distance-Algorithmus. Bitte benutzen Sie das gleiche Format wie in den Folien, um den Tutoren das Korrigieren zu erleichtern.[1] Indizieren Sie für jedes Feld der Tabelle, welche Operation Sie für die Berechnung des jeweiligen Wertes zugrunde gelegt haben.

Extrahieren Sie aus der Tabelle eine mögliche Sequenz mit minimalen Kosten (der Pfad darf nicht dem highlight Pfad aus `http://www.let.rug.nl/kleiweg/lev/` entsprechen). Stellen Sie die entsprechende Alignierung der beiden Wörter dar.

[1] Man kann theoretisch die Tabelle natürlich auch drehen, und so das Endergebnis links oben anstatt rechts unten zu haben etc. Bitte tun Sie das nicht.

Aufgabe 2) Minimum-Edit Distance Fragen **Punkte: 4**

1. In der Vorlesung haben wir uns hauptsächlich auf die folgende Minimum-Edit Distance konzentriert: Operationen substitute, delete, insert über dem Sprachalphabet mit fixen Kosten pro Operation unabhängig davon, was ersetzt, gelöscht, eingefügt wird.

   Gibt es Sprachen, in denen diese Art der Minimum-Edit-Distanz, keinen Sinn macht? Begründen Sie Ihre Antwort. (1 Punkt)

2. Wir haben in der Vorlesung gesagt, dass alle Pfade mit absteigenden Kosten (oder gleichbleibenden auf der Diagonale) eine minimale Folge von Operationen sind, die das Anfangswort in das Zielwort überführen. Gehen Sie zu `http://www.let.rug.nl/kleiweg/lev/` und geben Sie als Anfangswort *inse* und als Zielwort *exe* ein, ohne die Standardeinstellungen zu verändern. Der Algorithmus gibt Ihnen 6 Alignments aus, obwohl es in der Tabelle viel mehr absteigende Pfade nach unserer Definition gibt. Woran liegt dies? Wie müsste man die Pfaddefinition einschränken, um nur die 6 alignments als Pfade abzubilden? Welche Bedingung muss gelten, damit man die Pfaddefinition so einschränken darf? (3 Punkte)

Aufgabe 3) Tokenisierung in der Praxis **Punkte: 6**

Sie sollen nun mit regelbasierten Tokenisierern experimentieren. Das Toolkit NLTK stellt uns ein kleines Twittercorpus twitter_samples bereit, das aus 5.000 positiven, 5.000 negativen und 10.000 zusätzlichen Tweets besteht. Nun wollen wir einen der Tweets aus twitter_samples Korpus tokenisieren. Dazu können Sie die folgenden Befehle in der interaktiven Python-Shell aufrufen. Sie können diese natürlich auch in einer `*.py`-Datei speichern und von der Kommandozeile mit `python dateiname.py` aufrufen.[2]

```
import re #Wir laden auch regulaere Ausdruecke
import nltk
nltk.download('twitter_samples') # das korpus muss nur einmal heruntergeladen werden
from nltk.corpus import twitter_samples
all_tweets = twitter_samples.strings('tweets.20150430-223406.json')
tweet=all_tweets[3227]
print(tweet)
# 'RT @TheKouk: UK election betting. Next PM\nMiliband hot fav at $1.63\nCameron $2.25'
```

1. Nun splitten wir den Tweet an `whitespaces`, indem wir die `split()`-Methode des Strings aufrufen:

   ```
   print(tweet.split())
   ```

   Was wird richtig tokenisiert, wo gibt es Probleme? (1 Punkt)

2. Nun tokenisieren wir den Tweet mit den Regex-Bordmitteln von `nltk`. Dazu laden wir:

   ```
   from nltk import tokenize
   ```

   Jetzt definieren wir ein pattern, das **gültige** Zeichen in einem Wort festlegt:

   ```
   valid_pattern = "\w+|[^\w\s]+"
   ```

   Wie lautet dieses Muster natürlichsprachlich? (1 Punkt)

3. Wenden Sie nun das Pattern an.

   ```
   print(nltk.regexp_tokenize(tweet, valid_pattern))
   ```

   Was wurde verbessert? Wo sind noch Probleme? (1 Punkt)

4. Erweitern Sie `valid_pattern`, so dass der Tweet möglichst korrekt tokenisiert wird. Vergleichen Sie Ihre Tokenisierung auch mit der Tokenisierung, die von NLTK erstellt wurde:

   ```
   tokenized_tweet = twitter_samples.tokenized('tweets.20150430-223406.json')[3227]
   print(tokenized_tweet)
   ```

[2] NLTK ist im Pool installiert. Sollten Sie es bei sich zu Hause durchführen wollen, folgen Sie bitte den Instruktionen zur NLTK Tokenisierung unter `http://www.nltk.org/install.html`.

(3 Punkte)

Aufgabe 4) Der Porterstemmer **Punkte: 5**

Hierzu müssen Sie den Originalartikel unter `https://tartarus.org/martin/PorterStemmer/def.txt` lesen, insbesondere Section 2.

1. Was ist der Sinn einer Regel wie $SS \longrightarrow SS$? (1 Punkt)

2. Warum werden die beiden Worte *reader* und *read* nicht zum gleichen Stem reduziert? Was müssten Sie ändern, wenn Sie die beiden auf den gleichen Stem reduzieren wollten? (2 Punkte)

3. Nehmen Sie zwei beliebige Regeln des Porter Stemmers und finden Sie jeweils mindestens ein fehlerhaftes Stemming für die beiden Regeln. Hierbei nennen wir fehlerhaftes Stemming ein Stemming, das zu einer falschen Äquivalenzklasse führt. Ein Beispiel, das Sie nun natürlich nicht mehr verwenden können, ist das gleiche Stemming von *politic* und *polite* durch das Verwenden von Regel Step 4 auf *politic*. Angabe der Regeln ist notwendig. (2 Punkte)

Aufgabe 5) Bonusaufgabe zu Hyphenization **Punkte: 4**

In der untigen Tabelle finden Sie einige englische Ausdrücke aus einem Korpus (manchmal mit etwas Kontext), die mit Bindestrich auftreten. Manche von diesen sollte man bei einer Tokenisierung als ein Wort behandeln, manche eher trennen. Welche würden Sie wie behandeln und warum? Schlagen Sie Methoden vor, wie man Bindestrichworte, die man trennen sollte, identifizieren könnte.

write an e-mail
a final take-it-or-leave-it offer
anti-social
the salt-export ban
29-member alliance
He as a non-lawyer won't know
cross-border migration
aint-it-great-to-be-a-student Saturday night