

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.

Lösung zu 1)

Notation: i = Einfügen/insert, d = Löschen/delete, s = Ersetzung/substitution, m = keine Operation/match

	#	s	i	l	e	n	t
#	0	1_i	2_i	3_i	4_i	5_i	6_i
l	1_d	$2_{s,i,d}$	$3_{s,i,d}$	2_m	3_i	4_i	5_i
i	2_d	$3_{s,i,d}$	2_m	$3_{i,d}$	$4_{s,i,d}$	$5_{s,i,d}$	$6_{s,i,d}$
s	3_d	2_m	$3_{i,d}$	$4_{s,i,d}$	$5_{s,i,d}$	$6_{s,i,d}$	$7_{s,i,d}$
t	4_d	3_d	$4_{s,i,d}$	$5_{s,i,d}$	$6_{s,i,d}$	$7_{s,i,d}$	6_m
e	5_d	4_d	$5_{s,i,d}$	$6_{s,i,d}$	5_m	6_i	$7_{i,d}$
n	6_d	5_d	$6_{s,i,d}$	$7_{s,i,d}$	6_d	5_m	6_i

Table 1: Minimum Edit Distance für *Listen– Silent*.

Jeder absteigende Pfad ist möglich. Bitte gehen Sie im Tutorium einige Pfade und Alignierungen durch. Wichtig ist zu bemerken, dass man für die möglichen Alignierungen sich wirklich die

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

Pfeile bzw Operationen merken muss: die Zahlen alleine genügen nicht. Sonst könnte man zum Beispiel von rechts unten schräg nach oben gehen (von 6 zu 6), dies ist aber nicht erlaubt, da die entsprechenden Operation nur ein i sein darf.

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)

Lösung zu 2)

1. Logographische Sprachen wie das Chinesische. Hier ist das "Alphabet" zu groß.
2. Dies liegt daran, dass die direkte Kombination von vertikal/horizontal (delete und dann insert) zur gleichen Alignierung mit den gleichen Kosten führt wie diagonale Ersetzung. Dies gilt, solange die Kosten für delete plus insert den Kosten für substitute entsprechen. Will man nur die Zahl der verschiedenen Alignierungen, so sollte man Pfade nicht erlauben, die eine direkt aufeinanderfolgende vertikal/horizontal (bzw horizontal/vertikal) Kombination enthalten.

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
all_tweets = twitter_samples.strings('tweets.20150430-223406.json')
tweet=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]
```

(3 Punkte)

²NLTK 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>.

Lösung zu 3)

1. Probleme Punctuation wie “:”, “.” nicht abgetrennt, “\n” noch vorhanden.
2. Spezifiziert, dass ein valides Wort aus einer Sequenzen von Wortsymbolen (a-z, A-Z, 0-9 sowie _) bestehen kann, oder aus Symbolen, die weder Wortsymbole noch white spaces sind. Hierdurch wird nun Punctuation abgespalten.
3. Punctuation wird nun abgetrennt. Die Zahl “2.25” wird leider aufgetrennt.
4. Da die Veroderung des Regex-Musters von links nach rechts überprüft wird, kann man zum Beispiel Dezimalzahlen hardcoden und der Veroderung voranstellen: `\d+\.\d+|\w+|[\^\w\s]`

Diese Frage ist offener. Man könnte auch @ Zeichen diskutieren etc.

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 $SS \rightarrow SS$? (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)

Lösung zu 4)

1. Diese Regeln existieren, damit eine andere nicht fälschlicherweise greift. Zum Beispiel würde ohne diese Regel *caress* zu *cares* gestemmt werden.
2. Die Regel, die *er* abspaltet, gilt nur für ziemlich lange Stems. Man müsste diese auf kürzere ausweiten (was aber dann auch wieder zur Übergenerierung führen könnte).
3. Beispiele für fehlerhafte Äquivalenzklassen (die Regeln hierzu finden sich leicht:) *wit-ness/wit*, *colonize/colon*, *admiral/admire*, *university/universe*.

Aufgabe 5) Bonusaufgabe zu Hyphenization

Punkte: 4

In der untenen 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

Lösung zu 5)

Hier soll hauptsächlich diskutiert werden. Trennen sollte man wahrscheinlich Beispiele 2,4,5,8.

Möglichkeiten zur automatischen Entscheidung bei einer Hyphenisierung $x - y$. Frequenz der Kombinationen $x - y$ (je häufiger, desto mehr nicht trennen), Häufigkeit von x alleine (je häufiger, desto mehr trennen), Vorkommen von xy ohne Bindestrich (je häufiger, desto eher nicht trennen). Länge von x (je kürzer, desto eher nicht trennen). Je mehr Hyphens, desto eher trennen (betrifft 2 und 8).