Übung 2

Aufgabe 1.1

a) Erläutern Sie den Unterschied zwischen *Wordform* und *Lemma*.

b) Erläutern Sie den Unterschied zwischen *Word Type* und *Token*.

Aufgabe 1.2

Schreiben Sie ein Programm, das für einen gegebenen Korpus aus txt-Dateien folgendes tut:

a) Geben Sie die Gesamtzahl der Token und der Word Types des Korpus aus.

b) Iterieren Sie über alle Token des Korpus und bestimmen Sie pro Iteration die bis zum diesem Schritt gefundenen Word Types. Plotten Sie dann die Anzahl der pro Iteration gefundenen Word Types v auf der y-Achse gegen die Anzahl der gelesenen Token n auf der x-Achse.

Beispiel:
Korpus: "bla blubb bla blubber bla blubb"

Gelesenes Wort	bla	blubb	bla	blubber	bla	blubb
n	1	2	3	4	5	6
v	1	2	2	3	3	3

c) Optional:
Gemäß Heaps' Law besteht zwischen v und n folgender empirischer Zusammenhang: $v \approx k \cdot n^{\beta}$
Bestimmen Sie optimale Werte für k und β.
Tipp: logarithmieren Sie die n und v-Werte, so dass Sie einen linearen Zusammenhang zwischen $\log v$ und $\log n$ erhalten:

$$\log v \approx \log k \cdot n^{\beta} = \log k + \beta \log n.$$

Die Best-Fit-Werte für $\log k$ und β können Sie mithilfe linearer Regression bestimmen (in Python z.B. mithilfe der Klasse *LinearRegression* aus dem Paket *sklearn.linear_model*, siehe auch https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html)

Hinweise:

- Verwenden Sie den im Moodle-Kurs bereitgestellten Märchen-Korpus.
- Die Texte sind UTF8-kodiert mit Windows-Zeilenende (CR/LF bzw. \r\n).

Übung 2

Aufgabe 1.3

Implementieren Sie den BPE-Algorithmus, indem Sie Funktionen für den Token Learner und den Token Segmenter programmieren.

Trainieren Sie den Token Learner mit dem Text "Ali Baba und die 40 Räuber.txt". Wählen Sie dabei die Vokabulargröße k so, dass das vom Learner erzeugte Vokabular alle Worttypen des Eingabetextes vollständig umfasst.

Wenden Sie dann den Token Segmenter mit dem gelernten Vokabular auf den Text "Aschenputtel.txt" an. Welche Worttypen aus "Aschenputtel.txt" werden nicht vollständig, sondern nur auf Subword-Ebene segmentiert.

Hinweise:

- Ersetzen Sie vor dem Token Learning/Segmentierung alle Punktionszeichen durch ein Leerzeichen und nutzen Sie dann Leerzeichen als Wort-Separator.
- Benutzen Sie den Unterstrich _ zur Markierung von Wortenden.

Aufgabe 1.4

a) Spielen Sie den Minimum-Edit-Distance-Algorithmus aus [Jurasky 2020, Fig 2.17] für folgendes Beispiel durch: source = 'Wort', target = 'Satz'.
Zeigen Sie die ausgefüllte Distanz-Matrix und kennzeichnen Sie einen Alignment-Pfad (analog zu Fig 2.19). Die Kosten für Löschen und Einfügen seien = 1, die Kosten für Ersetzung seien = 1, wenn die zu ersetzenden Zeichen ungleich sind, und 0 für identische Zeichen.

b) Die Kosten für die elementaren Operationen seien wie in a) festgelegt.
   - Geben Sie eine untere Schranke für die Minimum-Edit-Distanz zwischen den Strings s1 und s2 an in Abhängigkeit der String-Längen m=|s1| und n=|s2| an.
   - Geben Sie eine obere Schranke für die Minimum-Edit-Distanz zwischen den Strings s1 und s2 an in Abhängigkeit der String-Längen m=|s1| und n=|s2| an.
   - Welche Laufzeit-Komplexität hat der Minimum-Edit-Distance-Algorithmus?
   - Welche Speicherplatz-Komplexität hat der Minimum-Edit-Distance-Algorithmus?
   - Zu welcher Algorithmen-Klasse gehört der Minimum-Edit-Distance-Algorithmus?

a) Für gewisse Anwendungen ist es sinnvoll, die Ersetzungen von Zeichen individuell zu gewichten:
   - Unter welchen Umständen wäre es sinnvoll, die Ersetzung von "n" durch "m" anders zu gewichten als die Ersetzung von "n" durch "q"? Was sollte mehr kosten?
   - Unter welchen Umständen wäre es sinnvoll, die Ersetzung von "o" durch "q" anders zu gewichten als die Ersetzung von "o" durch "s"? Was sollte mehr kosten?