Vorlesung "Computerlinguistische Techniken"

4. Übung (5.12.2023)

Wintersemester 2023/2024 – Prof. Dr. David Schlangen

1 Das Zipfsche Gesetz – 30

Überzeugen Sie sich davon, dass das Zipfsche Gesetz gilt. Beschaffen Sie sich dafür die folgenden, sehr unterschiedlichen Korpora:

- das Brown-Korpus (über NLTK)
- den Text von „The Jungle Book" (siehe moodle)
- den vollständigen deutschen Text der Bibel (siehe moodle)

Bestimmen Sie die absoluten Häufigkeiten der einzelnen Wörter und sortieren Sie die Wörter nach absteigender Häufigkeit. Für Dschungelbuch und Bibel wird die Methode `split` der String-Klasse nützlich sein.

Installieren Sie schließlich, falls noch nicht vorhanden, auf Ihrem Computer die Python-Bibliotheken `numpy` und `matplotlib`. Informieren Sie sich über logarithmische Skalen. Verwenden Sie dann die Funktionen `matplotlib.pyplot.plot` und `matplotlib.pyplot.loglog`, um die Häufigkeitsverteilungen graphisch darzustellen, und reichen Sie die drei Kurven mit einer kurzen Diskussion ein.

Geben Sie Ihren Programmcode in einer Datei `zipf.py` ab, die eine Funktion plot() mit den notwendigen Parametern enthält, die die jeweiligen Kurven produziert. Achten Sie darauf, dass Ihre Abbildungen gut lesbar sind, d.h. sinnvolle Beschriftungen der Achsen, und ggf. Titel und Legende enthalten.

2 Statistische Abhängigkeit – 40

Unigramm-Modelle nehmen an, dass für alle t die Ereignisse $X_t = w_1$ und $X_{t+1} = w_2$ statistisch unabhängig sind, d.h. die Wahrscheinlichkeit $P(X_{t+1} = w_2)$ hängt nicht davon ab, welches Wort an der Position t stand. Überzeugen Sie sich davon, dass diese Annahme falsch ist.

Sie können dazu folgendermaßen vorgehen. Wenn die beiden Ereignisse statistisch unabhängig wären, dann müsste nach Definition $P(X_t = w_1, X_{t+1} = w_2) = P(w_1) \cdot P(w_2)$ sein. Die *pointwise mutual information*

$$\text{pmi}(w_1, w_2) = \frac{P(w_1, w_2)}{P(w_1) \cdot P(w_2)} \approx \frac{C(w_1 w_2) \cdot N}{C(w_1) \cdot C(w_2)}$$

wäre in diesem Fall 1; die Abweichung der pmi von 1 ist also ein Maß für die statistische Abhängigkeit von w_1 und w_2 (dabei ist N die Korpusgröße). (Üblicherweise wird hier noch der Logarithmus genommen, aber das lassen wir hier aus.)

Berechnen Sie die absoluten Häufigkeiten aller einzelnen Wörter und aller aufeinanderfolgenden Paare $w_1 w_2$ von Wörtern im Brown-Korpus und bestimmen Sie daraus das angegebene Verhältnis. Wörter (nicht Bigramme), die weniger als zehnmal im Korpus vorkommen, sollen dabei ignoriert werden. Geben Sie schließlich die 20 Wortpaare mit dem höchsten pmi-Wert sowie die 20 Wortpaare mit dem niedrigsten pmi-Wert an. Was fällt Ihnen dabei auf?

Geben Sie Ihren Programmcode in einer Datei `pmi.py` ab.

3 Zufällige Texte generieren – 30

Das „Dissociated-Press“-System, das von MIT-Studenten in den 1970er Jahren entwickelt wurde, generiert aus einem Korpus zufällige neue Texte (siehe den Wikipedia-Artikel dazu). Dazu wählt es eine zufällige Stelle im Korpus aus; die ersten n Wörter des Ausgabetextes sind die n Wörter ab dieser Stelle. Um dann ein nächstes Wort zu generieren, wählt das System eine zufällige Stelle im Korpus aus, an der die letzten n ausgegebenen Wörter stehen, und gibt dann das nächste Wort aus dem Korpus aus. Dies wird so lange wiederholt, bis der Ausgabetext die gewünschte Länge erreicht hat.

Implementieren Sie eine Variante von „Dissociated Press“ mit n-Gramm-Modellen. Trainieren Sie dazu ein n-Gramm-Modell auf einem Korpus Ihrer Wahl (z.B. einem aus `nltk.corpus.gutenberg.words(austen-sense.txt')` or `nltk.corpus.gutenberg.words('shakespeare-hamlet.txt')`), z.B. mit Hilfe der Klasse `NGrams` aus dem moodle. Wenn Sie einen Zufallstext generieren, können Sie dann jeweils auf der Grundlage der $n-1$ letzten Wörter des Textes ein zufälliges n-tes Wort erzeugen.

Verwenden Sie Ihr System, um einige Texte der Länge 100 zu erzeugen. Geben Sie eine Auswahl der interessantesten Texte mit der Übung ab. Variieren Sie dabei n von 2 bis 4. Was beobachten Sie hinsichtlich der Qualität der erzeugten Ausgaben und der „Kreativität“ Ihres Systems?

Geben Sie Ihren Programmcode in einer Datei `generator.py` ab, die eine Klasse `Generator` mit einer Methode `generate()` enthält. Ein Gerüst dafür befindet sich auf Moodle.

Abgabe bis 19.12.2023, 13:00 Uhr, via Moodle