Übung 4

Aufgabe 4

Schreiben Sie ein Programm zur Erstellung, Analyse und Anwendung von n-Gram-Modellen mit folgender Funktionalität:

Einlesen einer Menge von Text-Dateien

Der Trainingskorpus sollen aus einer Menge von Textdateien bestehen, die vom Benutzer vorgegeben werden. Verwenden Sie in den nachfolgenden Experimenten den Märchen-Korpus aus Übung 2.

Tokenisierung, Normalisierung und Segmentierung der Texte in Sätze

Der Korpus soll nach folgenden Regeln vorverarbeitet werden:

- Wörter werden durch Whitespace und Interpunktionszeichen getrennt und in lower-case umgewandelt. Alle Whitepace und Interpunktionszeichen werden mit Ausnahme von **. ? ! ;** verworfen.
- Für die Satzsegmentierung werden **. ? ! ;** als Token behalten und definieren das Ende eines Satzes. Bei Texten, die ohne eines dieser Zeichen enden, wird der Satz bei Erreichen von EOF ebenfalls als beendet betrachtet (dies ist zum Beispiel bei "Hänsel und Gretel.txt" der Fall).
- Jeder Satz wird mit den Token <s> und </s> eingeschlossen.

Beispiel: der folgende Ausschnitt aus "Hänsel und Gretel.txt" wird wie folgt in Token und Sätze segmentiert:

> *Hu! da fing sie an zu heulen, ganz grauselig; aber Gretel lief fort, und die gottlose Hexe musste elendig verbrennen.\r\n Gretel aber lief schnurstracks zum Hänsel, öffnete sein Ställchen und rief:\r\n "Hänsel, wir sind erlöst, die alte Hexe ist tot!"\r\n*
>
> ```
> <s> hu ! </s>
> ```
>
> ```
> <s> da fing sie an zu heulen ganz grauselig ; </s>
> ```
>
> ```
> <s> aber gretel lief fort und die gottlose hexe musste
> elendig verbrennen . </s>
> ```
>
> ```
> <s> gretel aber lief schnurstracks zum hänsel öffnete sein
> ställchen und rief hänsel wir sind erlöst die alte hexe
> ist tot ! </s>
> ```

Um bei der späteren Anwendung der Language Models mit dem Out-of-Vocabulary-Problem umzugehen, ersetzen Sie von den Token, die im gesamten Korpus nur einmal vorkommen, 10 zufällig ausgewählte durch das Token <UNK> (siehe auch [Jurasky 2020, Abschnitt 3.3.1]).

Trainieren der Language Models

Trainieren Sie folgende Language Models, indem Sie die Wahrscheinlichkeiten $P(w_n)$, $P(w_n|w_{n-1})$ bzw. $P(w_n|w_{n-2}w_{n-1})$ berechnen:

- LM1: Unigram-Modell
- LM2a: Bigram-Modell ohne Smoothing

Übung 4

- LM2b: Bigram-Modell mit Laplace-Smoothing
- LM3a: Trigram-Model ohne Smoothing
- LM3b: Trigram-Modell mit Laplace-Smoothing

Analyse der Language Models

- Wie groß ist das Vokabular V?
- Aus wie vielen Unigrammen besteht das LM1?
- Aus wie vielen Bigrammen bestehen LM2a und LM2b prinzipiell? Wie hoch ist der Anteil der Bigramme von LM2a mit $P(w_n|w_{n-1}) \neq 0$?
- Aus wie vielen Trigrammen bestehen LM3a und LM3b prinzipiell? Wie hoch ist der Anteil der Trigramme von LM3a mit $P(w_n|w_{n-2}w_{n-1}) \neq 0$?
- Geben Sie für jedes Modell jeweils die 20 k-Gramme mit den höchsten (bedingten) Wahrscheinlichkeiten aus.
- Analysieren Sie für LM2a und LM2b die Verschiebung der Counts und Wahrscheinlichkeiten, die sich durch das Add-one-Smoothing ergeben. Orientieren Sie sich hierbei an den Abbildungen 3.5 -3.7 aus [Jurafksy 2020, Abschnitt 3.4.1]. Geben Sie die 10 Bigramme an, die durch das Smoothing den prozentual stärksten Verlust an Wahrscheinlichkeitsmasse erfahren.

Text Generierung

Generieren Sie mithilfe der Language Models LM1 – LM3b wie in [Jurafsky 2020, Abschnitt 3.3] beschrieben zufällige Texte basierend auf folgendem Textanfang:
`<s> es war`

Wählen Sie dabei das jeweils nächste Token w_n proportional zur Wahrscheinlichkeit

- $P(w_n)$ beim Unigram-Modell LM1,
- $P(w_n|w_{n-1})$ bei den Bigram-Modellen LM2a und LM2b
- $P(w_n|w_{n-2}w_{n-1})$ bei den Trigram-Modellen LM3a und LM3b.

Tipp: Um Elemente zufällig aus einer mit einer Wahrscheinlichkeitsverteilung gewichteten Liste zu ziehen, können Sie die Funktion random.choices verwenden (siehe auch: https://pynative.com/python-weighted-random-choices-with-probability/#h-random-choices)

Klassifikation unbekannter Texte

Welcher der beiden nachfolgenden Texte[1] entspricht eher einem Märchen?

Text 1: *Es war einmal eine Königin. Anspruchsvoll war sie und nicht sehr schön, doch weil sie ihren Untertanen Süße und Glück versprach, schenkten sie ihr die besten Böden. Das Reich der Königin wuchs, aber bald wurde sie krank, und ihre Feinde nutzten ihre Schwäche. Unruhe regte sich im Volk, die Zweifel nahmen zu. Wie ist die Königin zu retten? Oder wäre es besser, sie sterben zu lassen?*

Text 2: *So ließe sie sich erzählen, die Geschichte der runzligen Zuckerrübe, der "Königin der Feldfrüchte", wie Bauern sie nennen. Märchenhaft war sowohl ihr Aufstieg in Deutschland*

[1] Diese beiden Texte sind die ersten beiden Abschnitte aus folgendem Artikel: https://www.zeit.de/2021/15/zuckerrueben-geschichte-anbau-umweltschutz-gift-bienen/komplettansicht

und anderen Ländern Europas als auch der Wohlstand, den sie manchem Landwirt bescherte. Der aus ihr gewonnene Zucker verfeinerte eine Fülle von Lebensmitteln, kaum etwas kam noch ohne ihn aus. Inzwischen jedoch hat Zucker in unserer Nahrung ein solches Übermaß erreicht, dass er mehr Schaden als Beglückung verspricht. Zu viel Zucker kann zu Übergewicht führen, zu Diabetes, Herz-Kreislauf-Erkrankungen, Bluthochdruck. Selbst zwischen Zuckerkonsum und Krebs sehen einige Wissenschaftler Zusammenhänge.

Beantworten Sie die Frage, indem Sie die Texte zunächst wie den Trainingskorpus tokenisieren, normalisieren und in Sätze segmentieren. Ersetzen Sie dabei Wörter, die nicht im Vokabular vorhanden sind, durch das <UNK>-Token.

Berechnen Sie dann auf Basis der Language Models LM1, LM2b und LM3b jeweils die Perplexitäten. Welcher der Texte hat dabei jeweils die kleinere Perplexität (=höhere Wahrscheinlichkeit). Entspricht das Ihrer Intuition?

Weitere Experimente (optional)

Experimentieren Sie mit Varianten Ihrer Language Models. Beispielsweise könnten Sie folgendes ausprobieren:

- Verwendung anderer Smoothing-Techniken wie in [Jurafsky 2020, Abschnitt 3.4.2 – 3.5] beschrieben.
- Wenden Sie beim Text Preprocessing Stemmer oder Lemmatisierer für die deutsche Sprache an. Recherchieren Sie dazu nach Bibliotheken mit entsprechender Funktionalität, z.B. NLTK und spaCy.