IR-Grundlagen - Worksheet 4

Dieses Arbeitsblatt behandelt Language Models, im speziellen n-Gramme und smoothing. Ziel des Arbeitsblattes soll sein für Dokumente und Queries Language-models erstellen zu können, anhand der errechneten Warscheinlichkeiten eine ranked-Retrieval Liste zurückgeben zu können und Smoothing zu integrieren.

1. Zuerst erstelle wir ein einfaches Unigramm-modell. Dabei soll für jedes Dokument ein Language-Model erstellt werden. Dazu müssen folgende Funktionen implementiert werden. Erstellen sie für die Bearbeitung dieser Aufgaben wenn nötig weitere Funktionen und Klassen.

    a. Für jedes Dokument muss eine eigenes Language-Modell erstellt werden. Implementieren sie den Konstruktor der Klasse `Index` und legen Sie dort für jedes Dokument der Collection (gespeichert im `collection` Objekt - analog zu den letzten Arbeitsblättern) ein Objekt der `Unigramm` Klasse an. Sie können alle UniGramm-Modelle in der ArrayList `unigramms` speichern.

    b. Der Konstruktor der `Unigramm` Klasse ruft die abstrakte Methode

    ```
    float calculateProbabilities(Document doc)
    ```

    der Super-Klasse `NGramm` auf, in der für alle Wörter in diesem Dokument eine Warscheinlichkeit errechnet wird. Die Warscheinlichkeiten können in der Hashmap `probabilities` der Super-Klasse `NGramm` gespeichert werden. Die Formel hierfür lautet:

    $$P(wort_{doc}) = \frac{count\ (wort)}{length\ (doc)}$$

    Implementieren sie diese Methode.

    c. Die `Index` Klasse enthält die Methode

    ```
    ArrayList<Integer> searchUnigramm(String query)
    ```

    Diese Methode soll für eine gegebene Query für jedes Language-Modell der Colletion die Warscheinlichkeit berechnen, diese Query von dem jeweiligen Language-Modell erzeugt wurde (Denken sie daran dass in diesem Fall das Ranking nach $P(d|q)$ und $P(q|d)$ als gleichwertig behandelt werden können).

    d. Welche Probleme kann ein Uni-Gramm-Modell haben?

2. Als nächstes wollen wir unser Modell von einem Uni-Gramm in ein Bi-Gramm Modell umbauen. Implementieren sie dazu, analog zu Teilaufgabe 1, folgende Funktionen:

    a. Den Kunstruktor der `Index` Klasse. Erstellen sie nun für jedes Dokument ein Objekt der `Bigramm` Klasse. Diese können in der ArrayList `bigramms` gespeichert werden.

    b. Auch hier wird im Kunstruktor der `Bigramm` Klasse wieder die Funktion zur Berechnung der Warscheinlichkeiten aufgerufen. Implementieren sie diese Methode in der `Bigramm` Klasse. Die Formel lautet:

$$P(wort_i) = \frac{count\ (wort_{i-1}, wort_i)}{count\ (wort_{i-1})}$$

    c. Welcher Grenzfall stellt ein Problem für ein Bigramm Modell dar? Warum?

3. Jetzt versuchen wir diesen Grenzfall abzufangen. Wenn ein Queryterm nicht im Dokument vorkommt ist der gesamten Score gleich 0. Das versuchen wir durch Smoothing auszugleichen.
   Wir verwenden hierfür wie in den Vorlesungsfolien ein Languagemodel für die gesamte Collection. Die Formel lautet:

$$P(wort_{col}) = \frac{count\ (wort)}{length\ (col)}$$

   Diese Warscheinlichkeit wird folgendermaßen verwendet um die Ergebnisse zu smoothen:

$$P(wort \mid doc) = \lambda\, P(wort_i) + (1 - \lambda)\, P(wort_{col})$$

   Benutzen sie diese Formel (vorgeschlagener Wert $\lambda = 0.5$) um Smoothing in ihr System einzubauen.
