Programmierkurs Python II – SS 2011

Übung 9

---

1 Korpus-Sprachmodell (4 + 1 Punkte)

Implementiere eine Infrastruktur, die aus einer Sammlung von Texten ein Bigramm-Sprachmodell baut. Deine Klasse soll folgende Methoden unterstützen:

- `readCorpus(file)` soll aus einer Datei (`file`) Daten in das Sprachmodell einlesen. Wenn das Modell schon Daten enthält, sollen die gelesenen Daten dazu addiert werden.
- `getLabel()` gibt einen Namen für das Modell zurück, z.b. *Englisch* oder *Spam*. Der Name soll in `__init__` übergeben werden.
- `getLikelihood(sequence)` soll die Wahrscheinlichkeit zurückgeben, dass eine Wortsequenz (als Wortliste übergeben) vom Sprachmodell generiert wird.
- `getBigramFrequency(bigram)` und `getWordFrequency(word)` soll für ein Wortpaar bzw. ein Wort seine absolute Häufigkeit im Korpus zurückgeben.
- `generateFollowUp(word_list)` bekommt als Parameter eine Liste von Strings, die eine Wort-Sequenz darstellt. Sie soll das Wort zurück geben, das diese Frequenz am wahrscheinlichsten vervollständigt.
- `compareSequences(words1, words2)` bekommt als Parameter zwei Wortlisten. Diese Wortsequenzen sollen verglichen werden. Wenn `words1` die wahrscheinlichere Sequenz ist, soll 1 zurückgegeben werden, wenn sie gleich wahrscheinlich sind, 0, und sonst -1. Vergleiche die Google-Übersetzung für *Ein Schimmel galoppiert.* mit der korrekten Übersetzung *A white horse galops.* *Bonusaufgabe:* Bedenke, dass längere Sequenzen automatisch unwahrscheinlicher sind - versuche, das in die Implementierung mit einzubeziehen.

Du kannst Das Sprachmodell z.B. mit folgenden Texten füttern:
http://www.gutenberg.org/files/20795/20795.txt
http://www.gutenberg.org/dirs/etext97/hpaot10.txt
http://www.gutenberg.org/files/12/12.txt
http://www.gutenberg.org/dirs/etext05/pirat10.txt
http://www.gutenberg.org/dirs/etext04/nthtt10.txt

2 WSD mit Bayes (4 + 3 Punkte)

Schreibe einen einfachen Bayes-Klassifizierer, der WSD mit der *Bag of Words*-Methode macht. Als Trainingscorpora stehen auf der Homepage drei Dateien mit Textausschnitten zu den einzelnen Bedeutungen von *Schimmel* (`Schimmel_Brot`, `Schimmel_Pferd` und `Schimmel_Klavier`). Trainiere den Klassifizierer auf diesen Texten, betrachte den gesamten Text als Relevant (keine Beschränkung des Kontext-Fensters). Schreibe eine Methode `readTrainingCorpus(class,file)`, die die Datei `file` einliest und als Kontext der Klasse bzw. Wordbedeutung `class` betrachtet. Schreibe weiterhin eine Methode `classify(file)`, die einen Text mit einem ambigen Wort liest und die wahrscheinlichste Klasse für die Bedeutung des Wortes ausgibt. Du kannst diese Methode mit der Datei `Schimmel_wikipedia` testen, ein Ausschnitt aus einem Wikipedia-Artikel zu einer Bedeutung von Schimmel. Du kannst als A-Priori-Wahrscheinlichkeit eine Gleichverteilung über alle drei bedeutungen annehmen.

Bonusaufgaben: Die Annahme, dass alle Bedeutungen von *Schimmel* gleich häufig vorkommen, ist natürlich falsch. Die Bonuspunkte verteilen sich auf zwei Aufgaben:

2 Bonuspunkte: Versuche eine Möglichkeit zu finden, die A-Priori-Wahrscheinlichkeit besser zu schätzen (z.B. mit einer Suchmaschine und bestimmten Kontext-Merkmalen). Integriere diese Schätzung in Dein Programm (dokumentiere genau, wie Du es gemacht hast, wenn Du nicht auch die 2. Bonusaufgabe implementierst). Suche den Wikipedia-Artikel (oder eine andere längere Webseite) zu den anderen ($\neq$ der aus `Schimmel_wikipedia`) Bedeutungen von Schimmel und versuche, sie mit Deinem modifizierten Algorithmus zu klassifizieren.

1 Bonuspunkt: Implementiere diese Schätzung dynamisch in Dein Programm (der Code soll vor jedem Durchlauf die Verteilung selbst schätzen).

Abgabe bis Donnerstag, 07.07.2011, 08.30 Uhr