Programmierkurs Python II – SS 2013

Übung 9

---

1 Korpus-Sprachmodell (4 + 1 Punkte)

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

- `readCorpus(self, 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(self)` gibt einen Namen für das Modell zurück, z.b. *Englisch* oder *Spam*. Der Name soll in `__init__` übergeben werden.
- `getLikelihood(self,sequence)` soll die Wahrscheinlichkeit zurückgeben, dass eine Wortsequenz (als Wortliste übergeben) vom Sprachmodell generiert wird. Implementiere hierfür ein (möglichst einfaches) Smoothing, so dass jede Sequenz eine Wahrscheinlichkeit > 0 erhält.
- `getBigramFrequency(self,bigram)` und `getWordFrequency(word)` soll für ein Wortpaar bzw. ein Wort seine absolute Häufigkeit im Korpus zurückgeben.
- `generateFollowUp(self,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(self,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 Sequenzen *the little boy sleeps* und the old boy snores. *Bonusaufgabe:* Bedenke, dass längere Sequenzen automatisch unwahrscheinlicher sind - versuche, das in die Implementierung mit einzubeziehen.

In der `Bigrams`-Klasse stellen wir eine Methode zur Verfügung, die ein Trainings-Korpus von `www.gutenberg.org` einliest und abspeichert (siehe Test-Methode im Template).

2 WSD mit Bayes (4 Punkte)

Schreibe einen einfachen Bayes-Klassifizierer in einer Klasse `BayesClassifier`, 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). Für die Infrastruktur implementiere folgende Methoden:

- in `__init__(self, word, categories)` sollen das zu disambiguierende Wort und unterschiedliche Senses (= eine Liste von Strings) übergeben werden (z.B. `BayesClassifier('Schimmel', ['Pferd','Brot','Klavier'])`)
- `add_training_set(self,category,file)`, die die Datei `file` einliest und als Kontext der Klasse bzw. Wordbedeutung `category` betrachtet.
- `classify(self,file)`, die einen Text mit einem ambigen Wort liest und ein Paar aus wahrscheinlichste Kateogrie für die Bedeutung des Wortes ausgibt, sowie die berechnete Wahrscheinlichkeit. Du kannst diese Methode den `Schimmel_wiki[1-3]`-Dateien testen. (Tipp: Bei unseren Trainings-Korpora wird die Klassifikation wahrscheinlich nur funktionieren, wenn unbekannte Wörter *ignoriert* werden - also kein Smoothing, sondern die Wörter überspringen.)

Du kannst als A-Priori-Wahrscheinlichkeit eine Gleichverteilung über alle drei Bedeutungen annehmen. Um die Tests im Template zu benutzen, müssen die Schimmel-Dateien im gleichen Ordner wie die Python-Datei liegen, oder die Pfade angepasst werden.

Abgabe bis Donnerstag, 04.07.2013, 11.00 Uhr