Spamerkennung mit der Naïve-Bayes-Methode

Laden Sie das *Enron-Spammail-Dataset* (Enron1-6) von der Adresse `http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/index.html` herunter und packen Sie die Dateien aus. Die Verzeichnisse enthalten Unterordner mit den Namen *Spam* und *Ham*, die jeweils Emails mit Spam- und nicht-Spam-Nachrichten enthalten. Jede Datei enthält genau eine Mail, die auf einfache Art tokenisiert wurde. Der Zeichencode ist ISO-8859-1. Benutzen Sie Enron1-4 als Trainingsdaten, Enron5 als Development-Daten und Enron6 als Testdaten. (Die Developmentdaten werden Sie erst bei der nächsten Aufgabe benötigen.) Erzeugen Sie drei Ordner *train, dev* und *test* mit je zwei Unterordnern *spam* und *ham* und verschieben Sie die Dateien aus den Enron-Verzeichnissen dorthin.

Implementieren Sie dann einen Naïve-Bayes-Klassifikator, den Sie auf den Trainingsdaten trainieren. Der Klassifikator besteht aus

- einem Trainingsprogramm, welches die Modell-Parameter mit interpolierter Backoff-Glättung berechnet und dann mit pickle oder JSON in einer Datei speichert, und
- einem Anwendungsprogramm, welches die Parameterdatei einliest und damit Textdateien klassifiziert.

Die Programme sollen folgendermaßen aufgerufen werden:

```
python3 train.py train-dir paramfile
```

```
python3 classify.py paramfile mail-dir
```

train-dir ist ein Verzeichnis mit einem Unterverzeichnis für jede Klasse, das Dokumente dieser Klasse anthält. *mail-dir* ist ein Verzeichnis mit Email-Dokumenten, bspw. *data/test/spam*. Das Programm *classify.py* soll für jede Datei ihren Namen und die zugewiesene Klasse mit einem Tabulator als Trennzeichen auf einer Zeile ausgeben. Den Inhalt eines Verzeichnisses erhalten Sie bspw. mit der Python-Methode `os.listdir`.

Aus der Programmausgabe berechnen Sie dann die Genauigkeit. Dazu können Sie bspw. mit dem Befehl

```
python3 classify.py ... | cut -f2 | sort | uniq -c
```

berechnen, wie häufig die beiden Klassen ausgegeben wurden. (Das Programm classify.py soll nur annotieren und nicht die Genauigkeit berechnen.)

Ihr Programm sollte so allgemein implementiert werden, dass es auch für Text-Klassifikations-Probleme mit mehr/anderen Klassen anwendbar ist.

Vorüberlegungen

- Welche Teilaufgaben umfasst das Training?
- Was speichern Sie in der Parameterdatei?

- Welche Teilaufgaben umfasst das Anwendungsprogramm?
- Wie verhindern Sie Underflow bei der Berechnung der besten Klasse?

Schicken Sie das fertige Programm, die Liste der ausgegebenen Klassen und die erzielte Genauigkeit (Accuracy) an `schmid@cis.lmu.de`. Die Genauigkeit können Sie von Hand ausrechnen.