

Künstliche Intelligenz (SoSe 2024)

Aufgabenblatt 1

zu bearbeiten bis: 24.04.2024

Bearbeiten Sie die Implementierungsaufgaben generell in Zweier-Teams!

Aufgabe 1.1 (Python und Numpy)

Wir werden im Rahmen des Praktikums ausgiebig mit Python und numpy (Pythons mächtiger Bibliothek für Numerik und lineare Algebra) entwickeln. Während die MI-ler zumindest mit Python vertraut sein sollten, haben viele AI-ler, WI-ler und ITS-ler bisher allenfalls kleine Programme geschrieben. Lesen Sie sich deshalb zur Auffrischung das sehr gute Python- und Numpy-Tutorial der Stanford University durch:

<http://cs231n.github.io/python-numpy-tutorial/>

Es reicht, bis zum Beginn des Themas "Broadcasting" reinzuschauen.

Aufgabe 1.2 (Datenanalyse)

Wir werden in den ersten Wochen der Veranstaltung einen **News-Classifer** implementieren, der Artikel der New York Times in ihre jeweilige Kategorie (*sports, politics, ...*) einsortiert. Als ersten Schritt hierzu schauen wir uns die zugehörigen Trainings- und Testdaten an:

- Laden Sie sich aus dem Read.MI die Dateien `train.json` und `test.json` herunter. Wir werden unseren Klassifikator auf `train.json` trainieren, und auf `test.json` testen wie gut das gelernte Modell funktioniert. Sie können sich die JSON-Files im Browser anschauen (firefox `train.json`):

▼ docs:	
▶ data/train/arts/-mary-hinkson-a-star-for-martha-graham-dies-at-89.html:	{...}
▶ data/train/arts/10-million-gift-for-madison-estate.html:	{...}
▼ data/train/arts/100-things-to-do-before-high-school-on-nickelodeon.html:	
▼ body:	"Is there a minimum age requirement for having a bucket list? Apparently not, because on a cute new offering from Nickelodeon, three seventh-grade friends have one. Sort of. They aren't expecting to kick the bucket, but they are expecting life as they know it to end once they hit high school. And thus It is CJ (Isabela Moner) who first realizes, thanks to a funny "Glee"-parody dream, that the overscheduling and new interests looming in high school will cost her her two best friends, Fenwick (Jaheem Toombs) and Crispo (Owen Joyner). In truth, they are already drifting away with the start of seventh grade. So she concocts a scheme to renew their bonds that she knows they won't be able to resist: rescuing a raccoon that has been imprisoned by the school science teacher. There follows a fast-paced story of mayhem and poor educational theory that will make you glad your child doesn't attend their school but that has a decent number of amusing moments, especially for the middle-school-and-younger audience aimed for here. \n"
▶ intro:	"In "100 Things to Do Before_Hool," on Nickelodeon.\n"
label:	"arts"
▶ title:	"100 Things to Do Before_hool," on Nickelodeon\n"

- Mit den folgenden Kommandos können Sie die Daten in Python laden:

```
1 import json
2
3 with open("train.json") as file:
4     data = json.load(file)
5
6 print(data["docs"])
```

`data` ist nun ein geschachteltes Python-Dictionary. Hierbei enthält `data['vocabulary']` den Grundvokabular, d.h. alle Terme die in der Trainingsmenge vorkommen. Besonders häufige Terme (wie *the*) wurden gefiltert, und alle Terme wurden gestemmt, d.h. auf ihren Wortstamm reduziert (deshalb steht im Vokabular z.B. *audienc* statt *audience*).

`data['docs']` enthält die Dokumente selbst, jedes mit einem Titel, Einleitungssatz (`intro`) und Nachrichtentext (`body`). **Wichtig für Sie** sind vor allem die Felder `label` und `tokens`. `label` gibt an, zu welcher News-Kategorie ein Artikel gehört. In `tokens` finden Sie sämtliche Einzeltermine, die im jeweiligen Artikel auftauchen.

- Laden Sie `train.json` und analysieren Sie mit einem kleinen Python-Skript die Daten. Schauen Sie sich das erste Dokument an (könnten Sie anhand der Tokens die Klasse erraten?).

[Deliverable] Ermitteln Sie außerdem, welche Kategorien/Klassen es gibt.

Aufgabe 1.3 (Naive Bayes: Implementierung 1 – Training)

Implementieren Sie das **Training** eines Naive-Bayes-Klassifikators. Dieser soll automatisch Artikel der *New York Times* ihren Kategorien (wie *arts*, *technology*, *sports*, ...) zuordnen.

- In `classifier.py` (siehe Read.MI) finden Sie ein Programmgerüst, das `train.json` einliest und den Backbone einer Klasse `NaiveBayesDocumentClassifier` enthält.

- Sie starten das Skript mit folgendem Befehl:

```
python classifier.py --train
```

Sie erhalten einen `NotImplementedError`, weil die Methode `train()` noch nicht implementiert ist.

- Implementieren Sie nun `train()`:
 - Verwenden Sie – siehe Vorlesung – boolesche Bag-of-Words Features: Kommt ein Term in einem Dokument vor, ist das entsprechende Merkmal 1, ansonsten 0. Die *Häufigkeit* des Terms innerhalb des Dokuments ist irrelevant.
 - `train()` soll dementsprechend die Wahrscheinlichkeiten $P(x_i=1|y)$ für jeden Term x_i und jede Klasse y lernen, sowie den Prior $P(y)$. Diese Wahrscheinlichkeitstabellen können Sie z.B. als Dictionaries speichern:

```
P['sports']['superbowl']=0.32,
```

oder in Numpy-Arrays, z.B. (für die Wahrscheinlichkeit, dass das i -te Wort in einem beliebigen Dokument der j -ten Klasse auftritt):

```
P[i][j]=0.32
```

[Deliverable] Wahrscheinlichkeitstabellen für $P(y)$ und $P(x_i=1|y)$. Sammeln Sie Beispieleinträge als Plausibilitätschecks.

- Schreiben Sie außerdem etwas Code, um den trainierten Klassifikator zu speichern und später wieder zu verwenden. Hier bietet sich z.B. das Python-Modul `pickle` an.