Report

Montag, 31. August 2020 18:45

Angelina-Sophia Hauswald
Matrikel-Nr.: 785803

Aufgabe

Ich habe als Aufgabe die Sentiment Analysis gewählt mit dem Large Movie Review Dataset (https://ai.stanford.edu/~amaas/data/sentiment/).

Ich habe die Module

- re (um in den Daten nach den punctuation-based und character-based Features zu suchen)
- nltk (um in den Daten nach den syntactic-based features zu suchen)
- pandas (um mit den CSV-Dateien arbeiten zu können)
- pytest (um unit-Tests anzufertigen)
- os (um Ordner anlegen zu können)

verwendet.

Ich hatte mich dazu entschlossen einfach so viel Features, wie mir möglich waren hinzuzufügen und dann unter diesen die Features herauszufinden, die wirklich ausschlaggebend für meine Fragestellung waren. Die Features, die in der Aufgabenstellung angegeben waren, hatten nur leider keine guten Ergebnisse gebracht. Die besten Ergebnisse erhielt ich nachdem ich den overall sentiment score hinzugefügt habe, davor hatte ich mich erst gesträubt, da es sich nicht richtig anfühlte dieses Modul zu verwenden, wenn das doch im groben eigentlich meine Aufgabenstellung gewesen ist, dadurch wurden aber meine Ergebnisse deutlich besser.

Programmteile

Das Programm unterteilt sich in zwei Klassen, eine ist zuständig für das Training, die andere Klasse ist zuständig für das Klassifizieren von unbekannten Movie Reviews. Im folgenden erkläre ich alle Schnittstellen, die der Nutzer verwenden kann.

1. Training
   MSC_feature.py: **FeatureExtractor**

   Zum extrahieren der Features wird zunächst ein Objekt erstellt, dem Konstruktor wird ein statisches Feature übergeben, d.h. die Features, die außer die Feature, die im Code extrahiert werden, noch hinzugefügt werden sollen. Im Fall des Sentiment Movie Classifiers ist es so, dass ich hier festlege, dass es noch ein Feature gibt, welches sich "gold" nennt, für den Goldstandard. Der Goldstandard ist ja im Datensatz durch die Ordnerstruktur erkenntlich, um diesen hinzufügen zu können wird dieser gold Wert als statisches Feature festgelegt und von den anderen nötigen Funktionen als Argument erwartet.

```
fe = FeatureExtractor(static_features=["gold"])
```

   Mit der folgenden Funktion werden alle Features aus den pos/neg Trainingsdaten extrahiert und in data/train.csv gespeichert:

```
list_file_to_feature_csv(list_file="lists/train_pos.txt",
                         csv_file="data/train.csv",
                         append=False,
                         static_features={"gold": "pos"})
```

   Beschreibung Argumente der Funktion:

list_file	Dateipfad, in denen sich Daten befinden, aus denen Features extrahiert werden sollen
csv_file	Datei, in der Ergebnisse der Feature Extraktion gespeichert werden
append	False, wenn neue CSV erstellt werden soll, true, wenn CSV bereits existiert und nur noch in die bereits vorhandene geschrieben werden soll
static_features	Wert für den Goldstandard, ob es sich um pos o. neg Daten handelt, die bearbeitet werden sollen

Die folgende Funktion schreibt in eine CSV-Datei für alle Klassen, die bekannt sind (in dem Fall pos/neg), die durchschnittlichen Werte der errechneten Features (diese Funktion befindet sich in der MSC_Classifier.py über der Classifier Klasse):

```
document_feature_csv_to_class_feature_csv(
          document_feature_csv_path="data/train.csv",
          class_label_column_name="gold",
          label_feature_csv_path="data/averages.csv")
```

Beschreibung Argumente der Funktion:

Document_feature_csv_path	Dateipfad, der CSV, in der die Features zu den Trainingsdaten enthalten sind
Class_label_column_name	Standardmäßig auf gold, für den Goldstandard
Label_feature_csv_path	Dateiname, der CSV, in der die durchschnittlichen Werte der Features eingeschrieben werden sollen

2. Klassifizieren
   MSC_Classifier.py: **Classifier**

Zunächst wird wieder ein Objekt erstellt, dem Konstruktor wird die CSV übergeben, die durch das Training erstellt wurden, in der die durchschnittlichen Werte aller errechneten Features stehen. Darüber hinaus kann man im Konstruktor noch diejenigen Features festlegen, mit denen man klassifizieren möchte.

```
c = Classifier("data/averages.csv",
               only_use_features=["vader_compound","ratio_char_questionmark"]
               )
```

Folgende Funktion erstellt eine neue CSV-Datei, in der steht welche Datei der Movie Reviews welche Prediction, also welches Label bekommt (pos/neg) und was der eigentliche erwartete Goldstandard gewesen wäre.

```
classify_documents_in_file_and_save_to_csv(
         documents_csv_file="data/validation.csv",
         target_csv_file="data/validation_prediction.csv")
```

Beschreibung der Argumente der Funktion:

Documents_csv_file	CSV-Datei, die alle Dateinamen und die zugehörig errechneten Features zu jeder Datei enthält
Target_csv_file	Dateiname, der CSV-Datei, die erstellt werden soll

Diese Funktion überprüft wie gut die klassifizierten predictions sind:

```
read_confusion_matrix_from_prediction_csv(
         "data/validation_prediction.csv")
```

Reflektion

Dadurch, dass ich zusätzlich zu dem Projekt noch eine Klausur schreiben musste, um das Semester abzuschließen, blieben mir letztendlich für die Bearbeitung des Projekts nur noch ca. zweieinhalb Wochen (Klausur war am 12.08.), dementsprechend hatte ich mich nicht wirklich an irgendein Zeitmanagement gehalten. Das fällt mir bei Abschlussprojekten generell schwer, weil ich sehr schnell dazu neige den ganzen Tag nichts anderes zu machen, immerhin liegt der Laptop ja auch immer nur einen Griff weit entfernt, was auch dazu führt, dass sich selbst Pausen dann nicht immer nach Pausen anfühlen, wenn man am gleichen Ort arbeitet, an dem man auch seine Freizeit verbringt. Bei diesem Projekt habe

ich es also genauso gemacht, ich habe einfach Tag für Tag dran gearbeitet, ohne groß darauf zu achten, wieviel Zeit ich letztendlich damit verbringe. Bei Schwierigkeiten habe ich entweder im Internet nach einer passenden Antwort gesucht oder andere Studenten um Rat gefragt, git/github richtig zu benutzen fiel mir bspw. noch etwas schwer, da ich darin noch nicht so geübt war und dort hatte ich auch einen Freund ab und an um Hilfe gebeten (ich hatte ihn auch als Collaborator eingeladen).

Meine Erwartungen zu dem Projekt waren vor allem, dass ich es als noch viel umfangreicher eingeschätzt habe, als es letztendlich gewesen ist. Bisher war ich es gewöhnt, dass man in den Abschlussprojekte nie andere Bibliotheken verwenden darf, dadurch, dass das dieses Mal erlaubt war, ging vieles schneller und man musste sich viel mehr damit auseinandersetzen, wie man die gewünschten Funktionen verwendet als alles durchweg selbst implementieren zu müssen.

Beim nächsten Mal würde ich mir definitiv mehr Zeit einrechnen, um eine bessere Recherche zu den Features zu machen, um rauszufinden, welche Features sinnvoll sind hinzuzufügen und nicht alles ausprobieren zu müssen und aus den Ergebnissen Rückschlüsse zu ziehen.

Die Review, die ich bekommen habe, hat mir durchaus weitergeholfen, die meisten Sachen konnte ich noch gut umsetzen, wie bspw. Kommentare hinzufügen, jedoch habe ich allein aus zeitlichen Gründen es nicht mehr geschafft etwas im Code zu verändern, für mehr Lesbarkeit hatte Lukas vorgeschlagen, dass ich das befüllen von ratio_dict in der Funktion __ratio_features() in eine weitere Funktion hätte auslagern sollen, sehr viel sinnvoller, da stimme ich ihm zu, ich habe es nur leider nicht geschafft. Er merkte auch an, dass ich in der MSC_Classifier.py hätte die Funktionen, die über der Klasse stehen noch einmal in eine extra Datei hätte packen können, allerdings fand ich, dass sie thematisch zum Classifier gehören und deshalb ließ ich sie dort drin. Nichtsdestotrotz fand ich die Review hilfreich, zum ersten Mal hatte man die Möglichkeit Feedback zu bekommen und noch etwas verbessern zu können, bevor das Projekt endgültig bewertet wird.