Pro2-A Modulprojekt: Report

Sara Derakhshani
Matrikelnummer: 792483

31.08.2020

1. Zur Aufgabe

Als Projektaufgabe habe ich Ironie-Erkennung mit dem News Headlines Dataset for Sarcasm Detection gewählt.

Features

Die Features anhand derer klassifiziert wird sind: Durchschnittliche Wortlänge, Anzahl der Adjektive, Verhältnis von Adjektiven zu sonstigen POS-Tags, maximaler Polaritätsscore von Adjektiven und Adverbien und die Differenz zwischen der Summe positiver und negativer Polaritätsscores aller Wörter der Überschrift.
Ich habe unter Anderem auch Features wie durchschnittliche Anzahl aller Stopwörter, Anzahl langer Wörter (Wortlänge > 6), Anzahl der Zeichen, Anzahl verschiedenster POS-Tags, sowie synonymbezogene Features ausprobiert. Da die Accuracy aber immer sehr niedrig war (meist nur knapp über 50%) habe ich mich für die Features entschieden, die in Kombination die höchste Accuracy geliefert haben (ca. 59%).

Erweiterbarkeit

Es ist auf jeden Fall gut möglich zu versuchen die Accuracy zu verbessern. Dafür müssten in der Headline-Klasse nur neue Methoden hinzugefügt bzw. Methoden erweitert werden, um weitere Features hinzuzufügen. Eine weitere Möglichkeit wäre eine sinnvolle Gewichtung der Features zu implementieren, denn aktuell sind alle mit 1 gewichtet.
Dem Klassifizierer könnten weitere Evaluierungs- oder Klassifizierungsmethoden hinzugefügt werden.

2. Programmteile

Programmstruktur/Klassen

Das Programm besteht aus drei Klassen.
Die Headline-Klasse repräsentiert einzelne Überschriften und deren linguistische Features. Bei der Instanziierung eines Headline-Objekts werden die Features extrahiert und als Attribut gespeichert.
Die HeadlineData-Klasse wird zur Repräsentation von Datenkorpora verwendet. Bei Instanziierung werden die Daten eingelesen und jeder Dateneintrag als Headline-Objekt gespeichert. Die Feature-Statistiken des gesamten Korpus sind mithilfe eines Generators abrufbar.
Als Letztes repräsentiert die Classifier-Klasse den binären Klassifizierer. Sie verwendet die HeadlineData-Klasse um Trainingsdaten und die zu klassifizierenden Daten zu repräsentieren und deren Einzelstatistiken zu extrahieren, anhand derer dann aggregierte Statistiken für die beiden Klassen berechnet werden. Diese dienen dann zur Klassifizierung.

Module

Das logging-Modul habe ich während des Entwicklungsprozesses verwendet und behalten, um informative Outputs zu liefern. Die Module json und csv sind nützlich zum Einlesen der Daten, die in diesen Formaten bestehen. Mithilfe des time-Moduls wird die Laufzeit gemessen, das os-Modul dient zur Handhabung der Dateipfade, sowie Existenzprüfungen dieser und argparse ist für das Informieren über die benötigten Argumente sinnvoll. Ich nutze spacy für NLP-Aufgaben und zur Feature-Extraktion. Aus dem nltk.corpus-Paket verwende ich das sentiwordnet-Modul, um Polaritätsscores als Features verwenden zu können.

3. Reflektion

Als Erstes habe ich eine Klasse zum Splitten der Daten geschrieben. Zur Entwicklung der Programmstruktur und Schaffung eines Überblicks habe ich mich an die in der genauen Aufgabenstellung vorgeschlagene Bearbeitungsreihenfolge gehalten. Ich habe also erstmal mit einem Feature gearbeitet.
Ich habe die Klasse zur Repräsentation der einzelnen Überschriften mit ihren Features und die Klasse zur Verarbeitung der Korpora implementiert. Anfangs habe ich NLP in der HeadlineData-Klasse durchgeführt und die Outputs der Headline-Klasse als Attribut übergeben. Später habe ich mich dazu entschlossen NLP und Feature-Extraktion komplett in der Headline-Klasse durchzuführen, um einen klare Abgrenzung zwischen den Zuständigkeiten der Klassen zu haben.
Danach habe ich die Classifier-Klasse geschrieben und die anfängliche Struktur fast komplett im Endprodukt behalten.
Dieser gesamte Prozess ging schneller als ich gedacht hätte und meine größten Schwierigkeiten

hatte ich mit den Features bzw. der Accuracy. Ich habe viel Zeit damit verbracht unterschiedliche Features in verschiedenen Kombinationen auszuprobieren, aber die Accuracy blieb bei unter 60%, was bei binärer Klassifikation nicht besonders gut ist. Da mein Korpus nur aus Überschriften bestand, sind die paragraph- und satzbezogenen Features weggefallen (deswegen habe ich auch kein Sentence-Splitting durchgeführt), sowie Features die auf Satzzeichen bezogen waren. Aus dem Paper 'Automatic Detection of Irony and Humour in Twitter' von Barbieri und Saggion (2014) habe ich ein paar Ideen für Features entnommen, wie z.B. Polaritätsscores oder synonymbezogene Features (wovon ich nur Ersteres schlussendlich übernommen habe). Ich habe auch verschiedene Gewichtungen ausprobiert, aber das hat die Accuracy immer verschlechtert. Letztendlich habe ich die Features auf die oben beschriebenen beschränkt, da diese mir die höchste Accuracy geliefert haben.
Dieser Prozess war auch der Teil an dem Projekt in den ich die meiste Zeit gesteckt habe und den ich nicht erwartet hätte.
Da die Programmstruktur soweit fertig war, habe ich in der restlichen Zeit Variablen- und Methodennamen überarbeitet, zu lange Methoden gekürzt und viel an den Methodenargumenten gearbeitet. Ich fand es schwierig einzuschätzen wie viele und welche Inputs sinnvoll waren, wegen des Arbeitens mit mehreren Dateien und Dateiformaten. Ich wollte nicht so viel hartkodieren, aber auch möglichst wenig Inputs benötigen. Wie gut das gelöst ist, bin ich mir auch nicht sicher. Das wäre eine Sache mit der ich mich beim nächsten Projekt mehr beschäftigen würde. Außerdem habe ich keine Review machen lassen, was ich auch nächstes Mal anders machen würde. Besonders am Ende habe ich gemerkt, dass eine andere Meinung sehr hilfreich sein könnte.