Authorship Attribution von Tweets von Hillary Clinton und Donald Trump

Lea Wetzke, 06.09.2020

lwetzke@uni-potsdam.de

1 Zur Aufgabe

Im Rahmen des Projektes habe ich mich für die Methode der Authorship Attribution und das Hillary Clinton and Donald Trump Tweets Korpus entschieden. Die Erkenntnisse, die man durch die automatische Klassifizierung von Tweets beider PolitkerInnen ziehen könnte, könnten Einblicke in die syntaktischen Strukturen beider Kandidaten der Wahl 2016 zeigen: Gibt es verschiedene Eigenheiten, die jemand besonders oft wiederholt? Variiert Satzlänge oder Wortlänge? Benutzt jemand mehr Ausrufezeichen oder Großbuchstaben als die andere Person? Tweeten beide vielleicht weniger unterschiedlich, als man denkt? Für all das könnten die Ergebnisse ggf. nützlich sein. Ich habe mich im Rahmen des Projektes für folgende features entschieden:

- Anzahl an characters pro Tweet
- Anzahl an Großbuchstaben pro Tweet
- Anzahl an Kleinbuchstaben pro Tweet
- Anzahl an Sätzen pro Tweet
- Anzahl an Ausrufezeichen (!) pro Tweet
- Anzahl an Punkten (.) pro Tweet
- Anzahl an Fragezeichen (?) pro Tweet
- Anzahl an Kommas (,) pro Tweet
- Durchschnittliche Wortlänge pro Tweet

Ich habe mich hauptsächlich für Features entschieden, die meiner Meinung nach unterschiedliche Schreibstile erkenntlich machen könnten - mir ist beispielsweise beim Durchgehen des Korpus aufgefallen, dass Donald Trump gerne Ausrufezeichen und Großbuchstaben benutzt, während Hillary gerne mehrere kurze Sätze

und somit viele Punkte in ihre Tweets einbringt. Andere könnten das Projekt erweitern, indem sie weitere Features hinzufügen könnten, um die accuracy noch weiter zu verbessern: Dazu könnten beispielsweise andere Features bezüglich sentiment analysis oder ein Feature zum Festhalten oft benutzter Wörter oder Ausdrücke zählen.

2 Programmteile

Das Programm hat eine relativ simple Ordnerstruktur. Alle Klassen liegen im source Unterordner, die Unit Tests, der Korpus und die main befinden sich im root Ordner. Beim ersten Ausführen des Programms wird ein Ordner namens csvs erstellt, in dem sich alle selbst erstellten csvs befinden, also das gesplittete Korpus, die csv mit den Modellen und die Ergebnisse. Im source Ordner befinden sich die drei Hauptklassen: SplitTweetCorpus() in preprocessing.py, ProcessData() in data_processing.py und FeatureAnalysis() in feature_analysis.py. SplitTweetCorpus() repräsentiert Schritt 2 (Splitten des Korpus), ProcessData() ist für Schritte 3-5 gedacht, also das Einlesen der Train- bzw. Test-Datei, dem Verarbeiten der Daten, um auch linguistische Informationen aus ihnen ziehen zu können, dem Extrahieren der Features und dem Ausschreiben der Modelle in Form einer .csv-Datei. FeatureAnalysis() ist für die restlichen Schritte verantwortlich, also das Extrahieren der Featurevektoren aus den Test-Tweets, dem Vergleich mit den Modellen beider Autoren und dem Ausrechnen der Accuarcy und dem Ausschreiben in eine .csv-Datei. SplitTweetCorpus() ist mit Abstand die kürzeste Klasse. Hier wird lediglich das Korpus eingelesen, in einer Liste gespeichert, mithilfe einer Test-Train-Split-Methode aus dem sklearn-Modul in drei Teile geteilt (70/20/10) und in drei verschiede .csv-Dateien ausgeschrieben (test_set.csv, train_set.csv, val_set.csv). Ich benutze in allen Klassen das os-Modul, um in einen Unterordner zu schreiben bzw. um aus einen Unterordner zu lesen. ProcessData(), mit der öffentlichen Methode process_data(), liest das Train Set wieder ein, segmentiert die Sätze in den einzelnen Tweets, und taggt und lemmatisiert die einzelnen Wörter. Links werden aus den Tweets entfernt, da sie keine linguistische Bedeutung im Rahmen meiner gewählten Features haben - hierführ benutze ich das re-Modul, um eine regex anzuwenden. Daraufhin werden die Modelle ermittelt, indem der Mittelwert aller Features für jeden Train-Tweet errechnet wird. Jene aggregierten Statistikvektoren, einer pro Autor, werden dann in stats.csv ausgeschrieben. Für das Errechnen des Mittelwertes benutze ich das statistics-Modul. Featureanalysis() erbt von ProcessData(), da die Funktionen zum Extrahieren der Features "recycled" werden. Die Methode classify_tweet liest das Test- bzw. Validierungsset und stats.csv, also die generierten Modelle, ein. Danach wird für jeden einzelnen Test-Tweet der Featurevektor errechnet und der minimale Abstand zu den Vektoren der beiden Autoren berechnet: Der Tweet wird mit dem Autor mit dem kürzerem Abstand getaggt. Die Resultate werden dann in die results.csv geschrieben: Die erste Spalte ist der rohe Tweet, der zweite der/die vermutete Autor/in, und der/die eigentliche Autor/in. Die accuracy wird im Terminal

ausgegeben und rangiert von 50 bis 55 Prozent.

3 Reflektion

Da ich leider mitten in der Arbeit an dem Projekt eine Klausur schreiben musste, war mein Arbeitsprozess etwas fragmentiert. Vor der Klausur habe ich das Splitten und das Segmentieren, Taggen und Lemmatisieren gemacht, und erst ca. 2 Wochen später wieder mit dem Projekt weitergemacht. Das war unschön, da ich so erst wieder reinkommen musste. Am leichtesten war mit Abstand das Splitten des Korpus, diese Klasse habe ich innerhalb weniger Stunden geschrieben und eigentlich nicht im Verlauf des gesamten Projekts verändert. Am schwersten war das bugfixing. Ich hatte ein größeres Problem, das ich mir erst nicht erklären konnte. Das eine war, dass ich im Klassifizierungsteil öfter IndexErrors bekommen hatte - aus irgendeinem Grund waren zwei Listen, die gleich hätten sein sollen, ungleich. Ich habe mehrere Stunden mit Testen verbracht, um zu sehen, wo genau denn bloß dieser Fehler herkommen könnte. Was mir nicht aufegfallen war: Es gibt im Korpus ungefähr drei Tweets, die nur aus einem Link bestehen - besagter Link wird von der regex entfernt, wodurch nur noch ein Tweet mit dem leeren String bleibt! Ansonsten war das Extrahieren der Features nicht einfach, aber auch nicht schwer - jedoch habe ich damit die meiste Zeit verbracht. Die Arbeit unterschied sich nur wenig von meinen Erwartungen - ich bin erst davon ausgegangen, dass es wesentlich schwerer sein würde, als es im Endeffekt war. Einfach war es wie gesagt trotzdem nicht! Bei Sackgassen habe ich entweder gegoogelt, andere Leute gefragt oder einfach einen Tag nichts gemacht, um auf neue Ideen zu kommen. Eigentlich wollte ich noch mit sentiment analysis arbeiten, jedoch gefiel mir keine der angegeben Ressourcen dafür. Es war extrem langsam und unhandlich, weswegen ich es sehr schnell weggelassen habe, noch bevor ich es ins Projekt eingebaut habe. Außerdem habe ich noch überlegt, einen Ladebalken einzubauen, jedoch fehlte mir hierfür die Zeit am Ende. Ich würde nächstes Mal versuchen, meine Zeit besser einzuteilen. Die Klausur mitten im August hat mir Probleme gemacht. Ich hätte früher anfangen sollen, um nicht am Ende so sehr in Zeitdruck zu geraten (meine Krankheit hat das leider noch verschlimmert, die Verlängerung habe ich wirklich gebraucht). So hätte ich noch komplexere Features benutzen können, die wahrscheinlich die accuracy gesteigert hätten. Ich habe eine Review von Thomas Pham erhalten - diese war sehr hilfreich, besonders, da er Windows benutzt und nicht Linux. Es gab einige Enkodierungsprobleme, die ich nicht kommen gesehen habe, da Windows nativ ANSI und nicht UTF-8 benutzt.