Projektseminar Angewandte Informationswissenschaft

Dokumentation

Name: Sabrina Wirkner

Matrikelnummer: 2183349

GitHub-Benutzername: SabrinaWirkner

Sentiment Analyse mit Twitterdaten – Bau einer Klassifikation (Vergleich von Algorithmen des Machine Learnings)

Das Projekt beschäftigt sich mit der Sentiment Analyse von ereignisspezifischen Twitterdaten und dient als Grundlage eines größeren Projektes zum Aufbau einer Klassifikation mit den bestmöglichen Ergebnissen.

„Sentiment analysis [is] the extraction of sentiment, the positive or negative orientation that a writer expresses toward some object.“ (Jurafsky & Martin, 2008, Kapitel 7, S.1). In diesem Projekt wird eine Klassifikation erstellt, welche prüft, ob der Autor eines Tweets mit diesem eine positive, neutrale oder negative Empfindung ausdrücken möchte.

Projektidee

Im Projekt werden Methoden des Machine Learnings verwendet, um Tweets ein Sentiment (positiv, neutral, negativ) zuzuordnen. Die Ergebnisse der Methoden sollen nach Precision, Recall und Accuracy ausgewertet und mit intellektuell klassifizierten Twitterdaten verglichen werden. Das Ziel ist es, am Ende zu sehen, welcher Algorithmus die besten Ergebnisse erzielt hat, sowie welche Probleme es gab und Lösungsansätze zu sammeln, um das Programm in einer späteren Version effektiver zu machen. Das Projekt dient also vor allem dazu, einen Einblick in die Sentiment Analyse zu verschaffen.

Als Datensätze wurden im Juli 2016 ereignisspezifische, englische Tweets zur Comic-Con in San Diego gesammelt. Daraus ergaben sich drei Datensätze: 419 Tweets vor der Convention, 742 währenddessen und 807 nach der Comic-Con. Insgesamt standen also 1968 Tweets als Testdaten zur Verfügung. Diese wurden intellektuell nach positiv, neutral und negativ klassifiziert.

Auf der Basis dieser Daten soll im Projekt ein Sentiment Lexikon erstellt werden, welches jedem Wort ein positives, neutrales und negatives Gewicht zuordnet. Außerdem soll ein zweites Sentiment Lexikon erstellt werden, welches auf dem ersten Lexikon basiert, aber dessen Gewichte durch die Nutzung von Pointwise Mutual Information (PMI) verändert wurden.

Als Methoden werden der Naive Bayes Algorithmus, der Maximum Entropy Models (MaxEnt) Algorithmus und der Support Vector Machine (SVM) Algorithmus implementiert, wobei Naive Bayes und MaxEnt zusätzlich zum normalen Sentiment Lexikon noch mit dem PMI Sentiment Lexikon getestet werden.

Die Ergebnisse der Methoden werden ausgewertet und visualisiert, um ihre Leistung zu vergleichen.

Projektplan / Projektaufbau

Das Projekt wurde in Python implementiert und besteht aus fünf Teilen:

1. Datenverarbeitung
2. Erstellen der Sentiment Lexika
3. Implementieren der Machine Learning Algorithmen
4. Analyse / Auswertung
5. Visualisierung

Außerdem wurden folgende Bibliotheken verwendet:

- CSV
- NLTK (Natural Language Toolkit)
- Math
- Tabulate
- Matplotlib

Auf diese wird in den jeweiligen Methoden, in denen sie benutzt wurden, noch weiter eingegangen.

Datenverarbeitung:

1. Schritt: Datenverarbeitung

Funktion: process (data)

Bevor das Sentiment Lexikon erstellt und die Datensätze klassifiziert werden können, müssen sie verarbeitet werden. Der Grund dafür ist, dass für das Sentiment Lexikon das Vokabular aller Tweets gesammelt und der Wortstamm gebildet werden muss. Damit ist gesichert, dass allen Wortformen eines Wortes dieselben Gewichte zugeordnet werden. Außerdem ist es bei einem kleinen Datensatz wie diesem essentiell, dass Wortformen zusammengefasst werden, um ein möglichst hohes Vorkommen der Wörter in Tweets zu erzielen.

Die Methode bekommt den Datensatz (CSV-Datei) übergeben. Jede Zeile besteht aus dem intellektuell zugeordnetem Sentiment, dem Datum, dem Tweet und dem Hashtag. Für die Datenverarbeitung werden alle Tweets in eine Liste gespeichert.

Zunächst werden alle Wörter der Tweets in Kleinbuchstaben überführt. Obwohl Satzzeichen (wie Ausrufe- und Fragezeichen) ein Sentiment enthalten können („I think it's good?" vs. „I think it's good!"), werden diese gelöscht und es wird nur mit den Wörtern gearbeitet. Genauso wird mit Emoticons verfahren. Allerdings ist ein Ziel für eine spätere Version des Programms das Beachten von Sentiment von Satzzeichen und Emoticons.

Für jeden Tweet wird nun eine Liste seiner Worte erstellt. Aus dieser werden Stoppworte (Wörter, die für die Analyse keine oder geringe Relevanz haben, wie Pronomen oder Konjunktionen) gelöscht. Dafür wird die Stoppwortliste der NLTK-Bibliothek verwendet. Schließlich werden die Wortstämme mit dem englischen SnowballStemmer der NLTK-Bibliothek gebildet. Die Methode gibt eine Liste aller Tweets zurück, wobei ein Tweet nun eine Liste von Wortstämmen ist.

2.Schritt: Erstellen eines Dictionaries

Funktion: make_dictionary(data, data_list)

Diese Funktion bekommt unseren Datensatz und die Datenliste aus process(data) übergeben. Sie gibt ein Dictionary zurück, dessen Schlüssel die Tweets und dessen Objekte das intellektuell zugeordnete Sentiment sind. Da eine Liste als Datentyp für ein Dictionary nicht verwendet werden darf, werden die Tweets zu einem String überführt, welcher aus den Listeneinträgen und Leerzeichen zwischen diesen besteht.

Das Dictionary ist essentiell, um später das ursprüngliche Sentiment der Tweets aus unserer Datenliste prüfen zu können. Es eignet sich jedoch nicht als Datentyp für unsere weiteren Methoden, da die Reihenfolge der CSV-Datei nicht erhalten bleibt.

Erstellen der Sentiment Lexika:

3. Schritt: Erstellen des Sentiment Lexikons

Funktion: make_sentiment_lexicon(data_list, data_dict)

Diese Funktion erhält unsere Datenlist aus process(data) und unser Dictionary aus make_dictionary(data, data_list). In dieser wird das Sentiment Lexikon erstellt, bei welchem es sich um eine Textdatei handelt mit folgendem Aufbau:

Wort positives Gewicht neutrales Gewicht negatives Gewicht
Wort2 positives Gewicht neutrales Gewicht negatives Gewicht
Etc.

Für das Lexikon werden zunächst alle verschiedenen Wörter gesammelt, die in den Tweets vorkommen. Diese bilden das Vokabular. Sie werden in einer Liste gespeichert, die anschließend alphabetisch sortiert wird, sodass es später möglich ist, im Lexikon ein bestimmtes Wort zu finden.

Bei den Gewichten handelt es sich um P(Wort | Klasse), also die Wahrscheinlichkeit, dass ein Wort einer Klasse (positiv, neutral, negativ) zugeordnet wird. Diese lässt sich berechnen, indem das Vorkommen eines Wortes in der jeweiligen Klasse geteilt wird durch die Anzahl der Wörter, die insgesamt in Tweets mit diesem Sentiment vorkommen. Wenn also das Wort „great“ 250 Mal in positiven Tweets vorkommt und

es insgesamt 2300 Wörter in positiven Tweets gibt, ist P(great | positiv) = 250 / 2300 (vgl. Du Toit, 2015).

In der Funktion wird P(Wort | Klasse) für jedes Wort des Vokabulars und jede der drei Klassen berechnet, wobei das Ergebnis mit 1000 multipliziert wird, um mit einfacheren Zahlen arbeiten zu können. Anschließend wurden die Werte auf drei Stellen nach dem Komme gerundet.

Außerdem werden ungesehene Wort-Klasse-Fälle, also wenn ein Wort nie in einem Tweet mit einem der drei Sentiments vorkommt, abgefangen. In diesem Fall ist das Gewicht für die Klasse „0“.

In dieser Methode wird des Weiteren bereits für den Naive Bayes Algorithmus P(Class), die Wahrscheinlichkeit für die Klasse, berechnet und global gespeichert. Die ist abhängig von der Anzahl positiver, neutraler und negativer Tweets und ergibt sich aus der Anzahl an Tweets dieser Klasse dividiert mit der Anzahl an Tweets insgesamt, wobei das Ergebnis auf fünf Stellen nach dem Komma gerundet wurde.

Für die Rechnungen wurde „Decimal“ aus der Math-Bibliothek verwendet, damit Kommastellen bei der Division erhalten bleiben.

Zu Beginn des Projekts war geplant, die Gewichte mit Laplace-Smoothing zu berechnen, um ungesehene Wort-Klasse-Fälle besser zu verarbeiten (jedes Wort tritt hier mindestens einmal pro Klasse auf). Es wurde jedoch festgestellt, dass sich Laplace Smoothing nicht eignet, wenn die Klassen nicht gleich wahrscheinlich sind. In diesem Fall wurde die negative Klasse, die am wenigstens vorkommt, stark favorisiert, sodass dem Großteil der Tweets negatives Sentiment zugeordnet wurde.

Beispieleinträge aus dem fertigen Sentiment Lexikon:

great	0.866	0.037	0.106
hate	0.0	0.019	1.161
star	0.619	1.335	0.422

3.Schritt: Erstellen eines Sentiment Lexikons mithilfe von PMI

Funktion: do_pmi(sentiment_lexicon, data_list)

Die Funktion bekommt das erste Sentiment Lexikon aus make_sentiment_lexicon übergeben, sowie die Datenliste aus process und erstellt eine TXT-Datei mit einem zweiten Sentiment Lexikon.

Pointwise Mutual Information kann verwendet werden, um die Gewichte des Sentiment Lexikons zu verbessern, indem die Beziehungen unter den Wörtern berücksichtigt werden (vgl. Mullen & Collier, 2004).

Um den Zusammenhang zwischen den Wörtern festzuhalten, wird eine Wort-Wort-Matrix erstellt. Für jedes Wort wird gezählt, wie oft es zusammen mit jedem anderen Wort (Kontextwort) vorkommt. Hat man ein Vokabular von z.B. 3000 Wörtern, erhält man also 3000 Werte für jedes Wort. Normalerweise sagt man, dass ein Wort mit einem anderen Wort zusammen auftritt, wenn der Abstand bei maximal 4 Wörtern liegt. Da die Testmenge an Daten jedoch verhältnismäßig gering ist, wird in diesem Projekt gezählt, wie oft ein Wort zusammen mit einem anderem in einem Tweet vorkommt. In der Funktion ist die Matrix eine Liste von Listen, die für jedes Wort diese Werte enthalten (also im Beispiel 3000 Listen mit jeweils 3000 Werten).

Als nächstes werden in der Funktion die Werte des gemeinsamen Auftretens mit Joint Probabilities ersetzt. Diese werden wie folgt berechnet: Man teilt den ursprünglichen Wert durch die Summe aller Kontextwort-Werte des Wortes und multipliziert dies mit dem Quotienten aus dieser Summe und der Summe aller Kontextwort-Werte von allen Wörtern (vgl. Mullen & Collier, 2004). Zum Beispiel: Man hat das Wort „Apfel“ und das Kontextwort „Kuchen“. „Apfel“ tritt zwei Mal gemeinsam mit „Kuchen“ auf und fünf Mal gemeinsam mit allen Kontextworten. Insgesamt gibt es 32 Fälle, in denen Worte mit Kontextworten gemeinsam vorkommen (wobei nur entweder (Apfel | Kuchen) oder (Kuchen | Apfel) gezählt wird). Als Rechnung für die Joint Probability ergibt sich also:

(2 / 5) * (5 / 32)

Die Funktion berechnet dies für jedes Wort mit jedem Kontextwort und ersetzt die Werte in den Listen. Es ergibt sich also eine neue „Matrix“.

Mit den Joint Probabilities lassen sich nun P(w), Wahrscheinlichkeit des Wortes, und P(c), Wahrscheinlichkeit des Kontextwortes, errechnen. In der Funktion wird eine Liste von Listen erstellt, wobei jede Liste die Form [Vokabel, P(w), P(c)] besitzt.

Diese Werte werden nun in die PMI-Formel eingesetzt, um für alle Wort-Kontextwort-Fälle den PMI-Wert zu errechnen, der für das neue Sentiment Lexikon gebraucht wird. (PMI-Formel: Der Logarithmus des Basis Zwei von (Joint-Probability / (P(w) * P(c)).)

In der Regel nutzt man Pointwise Mutual Information auf einer sehr großen Datenmenge (indem z.B. Seiten über Suchmaschinen gecrawled werden) und ermittelt die Beziehungen zu typischen positiven oder negativen Wörtern. Da das Projekt dazu dient, mit kleinen, (ereignis-)spezifischen Datenmengen zu arbeiten, wird hier anders von den PMI-Werten gebraucht gemacht: Es werden die Werte aus dem ursprünglichen Sentiment Lexikon verwendet. Zu jedem Gewicht wird mit dem Kontextwort das Produkt aus dessen Gewicht und dem PMI addiert.

Hat man zum Beispiel:

great 1,5 0,8 0,02

excellent 1,7 0,6 0,2

PMI(great, excellent) = 0,6

So rechnet man:

great 1,5+0,6*1,7 0,8+0,6*0,6 + 0,02+0,6*0,2

Und dies für jedes Kontextwort von great.

Das neue Sentiment Lexikon besitzt dasselbe Format wie das ursprüngliche:

Wort positives Gewicht neutrales Gewicht negatives Gewicht

Wort2 positives Gewicht neutrales Gewicht negatives Gewicht

Etc.

Ein Problem, das beim Anwenden der Funktion aufkam, ist die sehr lange Verarbeitungszeit (20 Minuten), die auf die Schnelle nicht verringert werden kann. Der Grund dafür ist, dass mehrmals für jeden Wert der Liste (3000 * 3000 Werte) die ~2000 Tweets durchlaufen oder ein Wert ausgerechnet wird. Eine kleinere Menge an Daten ergibt an dieser Stelle keinen Sinn, da nur ein vollständiges Lexikon von Nutzen ist.

Implementieren der Machine Learning Algorithmen:

4. Schritt: Naive Bayes Algorithmus

Funktion: do_naive_bayes(sentiment_lexicon, data_list, filename)

Die Naive-Bayes-Funktion bekommt das Sentiment Lexikon aus make_sentiment oder do_pmi, unsere Datenliste aus process und einen Dateinamen für die CSV-Datei, die erstellt wird, übergeben.

Die Methode ordnet jedem Tweet aus unserer Datenlist ein Sentiment zu. Um das Sentiment zu bestimmen, muss für jeden Tweet P(Klasse | Tweet) für jede Klasse berechnet werden. Anschließend sucht man die größte Wahrscheinlichkeit und weist das entsprechende Sentiment zu. Die Ergebnisse werden in einer CSV-Datei gespeichert, welches für alle Daten das Sentiment und den dazugehörigen Tweet enthält.

Um P(Klasse | Tweet) zu berechnen, muss zunächst P(Tweet | Klasse) bestimmt werden. Dieser Wert ergibt sich, indem für alle Worte des Tweets das positive, neutrale oder negative Gewicht aus dem Sentiment Lexikon multipliziert wird (vgl. Jurafsky & Martin, 2008):

P(Tweet | Klasse) = P(Wort1 | Klasse) * P(Wort2 | Klasse) * … * P(WortN | Klasse)

Dabei ergab sich folgendes Problem: Ist für eines der Wörter des Tweets das Gewicht für die Klasse „0“, ist auch P(Tweet | Klasse) „0“. Normalerweise wird dies durch Laplace-Smoothing verhindert. Hier wird das Problem umgangen, indem nur mit dem Wert multipliziert wird, wenn dieser nicht „0“ ist. Ist der Wert „0“, wird mit einer Konstante von 0,025 multipliziert. Dies ist ein Durchschnittswert, mit dem beim Testen die besten Ergebnisse erzielt wurden (getestet wurden zudem 0,035, 0,03 und 0,02).

Schließlich wird P(Klasse | Tweet) für positiv, neutral und negativ berechnet, indem P(Tweet | Klasse) mit P(Class) multipliziert wird, welches bereits in make_sentiment_lexicon errechnet wurde und global zur Verfügung steht. Die drei Werte werden verglichen und der höchste Wert bestimmt die Klasse für den Tweet.

5. Schritt: Maximum Entropy Algorithmus

Funktion: do_max_ent(sentiment_lexicon, data_list, filename)

Die Funktion bekommt das Sentiment Lexikon aus make_sentiment_lexikon oder do_pmi, die Datenliste aus process und einen Dateinamen übergeben, welcher als Name für die CSV-Datei dient, die in der Funktion erstellt wird.

Der MaxEnt-Algorithmus kann verwendet werden, um das Sentiment jeden Tweets zu bestimmen.

Für den Algorithmus wird in der Funktion die gewichtete Feature-Summe für jeden Tweet und jede der drei Klasse berechnet, sodass sich eine positive, neutrale und negative Feature-Summe ergeben. Für die Feature-Summe werden lediglich die entsprechenden Gewichte für jedes Wort des Tweets addiert (vgl. Go, Bhayani & Huang, 2009).

Als nächsten werden in der Funktion diese Werte in die MaxEnt-Formel eingesetzt. Diese sieht für die positive Klasse wie folgt aus:

$$P(\text{positiv}|\text{Tweet}) = e^{(\text{pos\_feature\_sum})} / (e^{(\text{pos\_feature\_sum})} + e^{(\text{neut\_feature\_sum})} + e^{(\text{neg\_feature\_sum})})$$

Wie bei Naive Bayes wird nun die größte Wahrscheinlichkeit ermittelt, welche das Sentiment für den Tweet festlegt. Dieses und der Tweet werden in eine CSV-Datei geschrieben.

6. Schritt: Support Vector Machine Algorithmus

Funktion: do_svm(sentiment_lexicon, data_list, data_dict, filename)

Die Funktion erhält das Sentiment Lexikon aus make_sentiment_lexicon, die Datenliste aus process, das Dictionary aus make_dictionary und einen Dateinamen übergeben, der auch hier als Name für die CSV-Datei, die die Funktion erstellt, gebraucht wird. Für diese Methode wird nur das ursprüngliche Sentiment Lexikon verwendet, da nur mit den Vokabeln, nicht aber mit den Werten, gerechnet wird. Pointwise Mutual Information kann auf diese Weise also nicht genutzt werden, um den Algorithmus zu verbessern

Für den SVM-Algorithmus werden in der Funktion für jeden Tweet Vektoren (Liste von Werten) erstellt, wobei die Werte des Vektors die Anzahl ist, mit der jedes Wort aus dem Vokabular im Tweet auftritt (vgl. Mullen & Collier, 2004). Es wird also wieder eine Art Matrix erstellt, die eine Liste ist aus Listen, welche für jeden Tweet diese Werte enthalten (also bei der Verwendung aller Tweets 1968 Listen mit jeweils ~3700 Einträgen).

Daraufhin wird für jeden Tweet sein Vektor verglichen mit allen restlichen Vektoren, um den Ähnlichsten zu ermitteln. Dafür wird das normalisierte Dot-Produkt (Ähnlichkeitswert) mit jedem Vektor errechnet.

In der Regel bestimmt man die drei oder fünf ähnlichsten Vektoren. Hier werden die drei ähnlichsten Vektoren ermittelt, da dies zu besseren Ergebnissen geführt hat. Das Sentiment des Tweets ergibt sich aus dem intellektuell bestimmten Sentiment dieser ähnlichsten Tweets (positiv wird zugeordnet, wenn mindestens zwei der drei Vektoren von Tweets mit positivem Sentiment sind etc.).

Der Tweet und das zugewiesene Sentiment werden in eine CSV-Datei geschrieben.

Während die Methode bei der kleinsten Datenmenge (~400 Tweets) noch verhältnismäßig schnell verarbeitet wurde (einige Minuten), hat sie bei der Verwendung aller Daten (~2000 Tweets) eine Laufzeit von etwa 1 ½ Stunden. Auch hier liegt das Problem erneut in dem mehrmaligen Durchlaufen langer Schleifen.

Analyse / Auswertung:

7. Schritt: Analyse

Funktion: analyze(int_data, data, name, filename)

Die Funktion bekommt die CSV-Datei mit den intellektuell zugeordneten Sentiments übergeben, sowie die neue Datei, die in do_naive_bayes, do_max_ent oder do_svm erstellt wurde, einen Namen, der als Titel für die Analyse dient, und einen Dateinamen für die TXT-Datei, die bei der Auswertung erstellt wird.

Um die Leistung der angewandten Algorithmen auszuwerten, wird Gebrauch gemacht von der Gold-System-Labels Matrix (vgl. Jurafsky & Martin, 2008). Diese stellt dar, wie oft das Sentiment, das vom System bestimmt wurde, mit dem übereinstimmt, das intellektuell zugeordnet wurde. Mit diesen Werten kann am Ende Precision, Recall und Accuracy für den Algorithmus bestimmt werden, wobei zwischen macro- und microaverage Werten unterschieden wird.

Zunächst wird die Confusion Matrix (siehe Abbildung 1) erstellt, die alle Werte enthält:

Confusion Matrix:

Naive Bayes	pos	neut	neg
pos	265	152	38
neut	299	951	111
neg	29	51	72

Abbildung 1, Confusion Matrix

Die obere Zeile (Spalten) zeigt die Sentiments, die intellektuell bestimmt wurden, während die linke Spalte (Zeilen) die vom Algorithmus zugeordneten Sentiments zeigt.

Um Precision, Recall und Accuracy zu bestimmen, muss die Tabelle als nächstes aufgeteilt werden zu Contingency Tables für das Zuordnen von positivem, neutralem und negativem Sentiment. Der Contingency Table (siehe Abbildung 2) für die positive Klasse sieht z.B. so aus:

Contingency Table Positive:

pos	yes	no
yes	265	190
no	328	1185

Abbildung 2, Contingency Table

Der Yes-Yes-Wert (265) gibt die Anzahl an Tweets an, für die sowohl Mensch als auch System das positive Sentiment bestimmt haben. Unter dem Yes-No-Wert (328) versteht man die Menge an Tweets, die intellektuell positiv und vom System negativ deklariert wurden. Etc.

Als letztes erstellt die Methode den Pooled Table (siehe Abbildung 3), der sich aus den drei Contingency Tables ergibt:

Pooled Table:

	yes	no
yes	1288	1010
no	680	3256

Abbildung 3, Pooled Table

Mit den Werten dieser vier Tabellen werden Precision, Recall und Accuracy berechnet.

Für die macroaverage Werte werden die einzelnen Werte für die Contingency Tables berechnet und dann durch drei dividiert, während sich die microaverage Werte aus dem Pooled Table ergeben.

Die Precision wird berechnet, indem man den True-Positive-Wert (TP) (yes-yes) durch den False-Positive-Wert (FP) (no-yes) teilt. Eine hohe Precision sagt aus, dass wenig Tweets, die (z.B.) nicht positiv sind, als positiv bestimmt wurden. Der Recall ergibt sich aus dem Quotienten vom TP-Wert und der Summe des TP-Werts und des False-Negative-Werts (FN) (yes-no). Ein hoher Recall bedeutet, dass möglichst viele (z.B.) positive Tweets positives Sentiment zugeordnet bekommen haben. Die Accuracy ist ein allgemeiner Wert über die Leistung der Algorithmen. Für diese wird die Summe der TP- und True-Negative-Werte (TN) (no-no) durch die Summe aller Werte geteilt (vgl. Jurafsky & Martin, 2008).

Des Weiteren speichert die Funktion alle dieser Werte global, damit bei der Visualisierung Zugriff auf diese besteht.

Mithilfe der Tabulate-Bibliothek werden die Tabellen erstellt und gemeinsam mit Precision, Recall und Accuracy in eine TXT-Datei geschrieben.

Visualisierung:

8. Schritt: Visualisierung

> **Funktion:** visualize(int_data, data, data2, data3, name, name2, name3, a_name, a_name2, a_name3, a_name4, a_name5, filename)

Für die Visulisierung werden die CSV-Tabelle mit den intellektuell bestimmten Sentiments, die CSV-Tabellen der drei Algorithmen aus do_naive_bayes, do_max_ent und do_svm, die Namen der Algorithmen, die Dateinamen der TXT-Dateien der Analyse aus analyze, sowie der Dateiname für die PDF-Datei, die die Funktion erstellt, übergeben.

In der Visualisierung werden drei verschiedene Themen der Analyse visualisiert, wobei die Balken- und Kreisdiagramme mit der Matplotlib-Bibliothek auf Basis der Matplotlib-Examples erstellt werden.

Zunächst werden drei Balkendiagramme erstellt, die für Naive Bayes, MaxEnt und SVM die macro- und microaverage Precision (siehe Abbildung 4), den Recall und die Accuracy vergleichen. Hierfür wird das Sentiment Lexikon ohne Einfluss von PMI verwendet, sowie die global gespeicherten Werte von analyze.

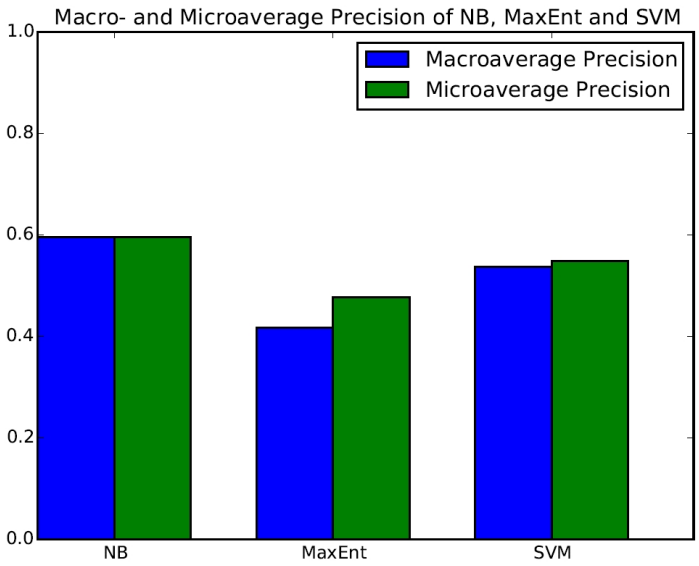

Abbildung 4, Macro- and Microaverage Precision

Als nächstes soll das Verhältnis von positiven, neutralen und negativen Sentiments, die vom System bestimmt wurden, mit den intellektuell zugeordneten Sentiments verglichen werden. Dafür werden in der Funktion vier Kreisdiagramm erstellt, die das Verhältnis für das intellektuelle Klassifizieren (siehe Abbildung 5), Naive Bayes, MaxEnt und SVM darstellen.

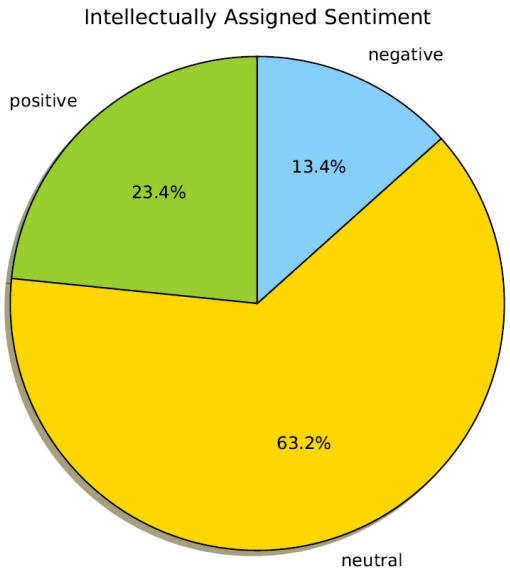

Abbildung 5, Intellectually Assigned Sentiment

Als letztes soll die Leistung von PMI visualisiert werden. Dafür werden zwei Balkendiagramme erstellt, wobei das erste die macroaverage Werte für Naive Bayes, Naive Bayes mit PMI, MaxEnt und MaxEnt mit PMI (siehe Abbildung 6) zeigt und das zweite die entsprechenden microaverage Werte.

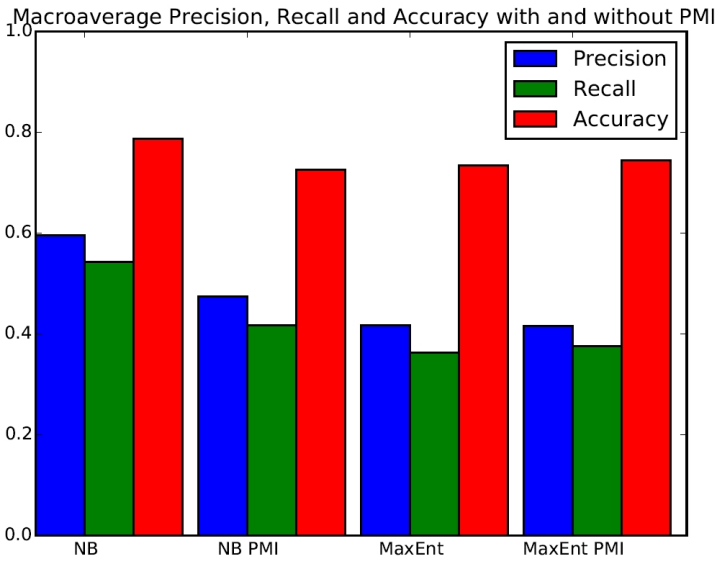

Abbildung 6, Macroaverage Precision, Recall and Accuracy

Alle erstellten Diagramme speichert die Funktion zusammen in eine PDF-Datei.

Main-Methode:

Die Default-Version des Programms arbeitet mit dem kleinsten Datensatz der Twitterdaten (~400 Tweets), um die geringste Laufzeit zu garantieren (ca. 30 Minuten).

Zunächst werden alle drei Datensätze in process verarbeitet und es werden entsprechende Dictionaries in make_dicitionary erstellt. Die resultierenden Datenlisten und Dictionaries werden jeweils vereinigt, um aus diesen Daten das erste Sentiment Lexikon mit make_sentiment_lexicon zu erstellen.

Anschließend wird das zweie Sentiment Lexikon mithilfe von PMI in do_pmi erstellt. Im Ordner befinden sich nun zwei TXT-Dateien: sentiment_lexicon.txt und sentiment_lexikon_pmi.txt.

Daraufhin werden der Naive-Bayes-Algorithmus und MaxEnt-Algorithmus nacheinander jeweils mit beiden Sentiment Lexika in do_naive_bayes und do_max_ent angewendet. Der SVM-Algorithmus wird in do_svm mit dem ersten Sentiment-Lexikon aufgerufen. Im Ordner sind nun die entsprechenden fünf CSV-Dateien hinzugekommen.

Die Daten werden mit analyzed ausgewertet, sodass weitere fünf TXT-Dateien mit der Auswertung der Algorithmen im Ordner erstellt werden.

Als letztes wird visualize aufgerufen, wodurch für die Ergebnisse Diagramme erstellt werden, die in einer PDF-Datei (visualization.pdf) gespeichert werden.

Während der Anwendung des Programms gibt die Konsole entsprechende Hinweise zum Stand des Programms und Dauer der Laufzeit aus.

Auskommentiert befinden sich im Programm noch zwei weitere Versionen der Main-Methode, die exakt gleich ablaufen: Eine Version, die mit dem vollen Datensatz (~2000 Tweets) arbeitet, und eine Version, die einzeln die beiden anderen Datensätze (~700 Tweets und ~800 Tweets) verwendet. Da beide Versionen jeweils ca. zwei Stunden laufen, sind sie nicht der Default und dienen nicht zum Testen des Programms, sondern nur zur weiteren Auswertung der Algorithmen.

Anleitung zur Ausführung / Benutzung

Um das Programm auszuführen muss die Python-Datei (sentiment_analysis.py), sowie der Data-Ordner mit den intellektuell klassifizierten Tweets vorliegen. Im Repository befindet sich außerdem ein Results-Ordner, der alle CSV- TXT- und PDF-Dateien enthält, die bei der Auswertung aller Datensätze einzeln und zusammen, entstehen.

Vor dem Starten des Programms soll überprüft werde, ob alle Bibliotheken korrekt importiert werden. Ist dies nicht der Fall, müssen sie installiert werden. Für das Programm wurde nur tabulate installiert, aber es ist nicht auszuschließen, ob für andere Bibliotheken bereits früher etwas installiert wurde (nltk, csv, matplotlib, numpy, math). Tabulate lässt sich wie folgt in der Konsole installieren:

```
pip install tabulate
```

Startet man das Programm, werden die Algorithmen auf den kleinsten Twitter-Datensatz (comiccon_before_classified.csv) angewendet. Die Konsole zeigt an, woran das Programm gerade arbeitet und was bereits getan wurde. Außerdem wird man darauf hingewiesen, wenn ein Schritt länger als wenige Minuten dauert. Insgesamt läuft das Programm in etwa eine halbe Stunde. Wenn es fertig durchlaufen ist, erscheint der Satz „Done.“.

Nach dem Ausführen des Programms befinden sich im Ordner folgende Daten:

- Fünf CSV-Dateien der Tweets mit ihren zugeordneten Sentiments von Naive Bayes, MaxEnt und SVM
- Zwei TXT-Dateien für die Sentiment Lexika
- Fünf TXT-Dateien für die Analyse der Algorithmen
- Eine PDF-Datei, die die Diagramme für die Visualisierung enthält

Möchte man den gesamten Datensatz auswerten, sind folgende Schritte zu tun:

1. Kommentiere den ersten Block des Codes der Main-Methode aus.
2. Lösche die Kommentierung um den zweiten Block der Main-Methode („Long Version using the Joined Data Set“).
3. Starte das Programm.

Möchte man die anderen beiden Datensätze auswerten, sind folgende Schritte zu tun:

1. Kommentiere den ersten und zweiten Block des Codes der Main-Methode aus.
2. Lösche die Kommentierung um den dritten Block der Main-Methode („Analyzing the After and During Sets of Twitter Data).
3. Starte das Programm.

Das Programm läuft auf dieselbe Weise, jedoch dauert die Ausführung in etwa zwei Stunden, weshalb diese Versionen nicht der Default sind.

Alle Teile der Main-Methode können nacheinander ausgeführt werden, da die Dateien nicht überschrieben, sondern entsprechend neue erstellt werden.

Reflexion

Das Ziel des Projekts war es, ein Programm zu erstellen, welches Twitterdaten nach Sentiment (positiv, neutral, negativ) klassifizieren kann und welches verschiedene Methoden verwendet, sodass diese am Ende verglichen werden können. Alle ausgewählten Methoden des Machine Learnings (Naive Bayes, Maximum Entropy Model, Support Vector Machine, Pointwise Mutual Information) konnten erfolgreich implementiert und auf die Datensätze angepasst werden und es wurden zufriedenstellende Ergebnisse bei der Klassifikation erreicht. Insgesamt lag die Accuracy zwischen 68 und 78 Prozent, was ein gutes Ergebnis ist, wenn man bedenkt, dass nur die Basismethoden implementiert wurden.

Ursprünglich war es Teil des Projektplans, dass neben der Erstellung eigener Sentiment Lexika auch ein professionelles Sentiment-Lexikon getestet wird. Da dieses jedoch nur positive und negative Gewichte enthält, hätten dafür alle Algorithmen ein zweiten Mal, in auf das neue Sentiment Lexikon angepasster Form, implementiert werden müssen. Da dafür die Zeit nicht ausreichte, wurde die Idee auf eine zukünftige Version des Programms verschoben.

Des Weiteren war zu Beginn des Projekts geplant, auch die drei verschiedenen Datensätze miteinander zu vergleichen, indem in etwa Diagramme zum Verhältnis der verschiedenen Sentiments erstellt werden. Damit dies getestet werden kann, müsste das Programm jedoch hintereinander alle Datensätze verarbeiten, was zu einer Laufzeit von ca. vier Stunden führen würde. Aufgrund dessen wurde es möglich gemacht, die Datensätze einzeln auswerten zu lassen, sodass am Ende alle Analysen im Ordner zur Verfügung stehen und eigenständig verglichen werden können.

Neben der langen Laufzeit war ein Problem, das nicht gelöst werden konnte, die Anwendung des Laplace-Smoothings. Wie bereits erwähnt, führte dieses zu einer Favorisierung der negativen Klasse. Obwohl der Grund des Problems erkannt wurde, ist weitere Recherche nötig, um es eventuell in der Zukunft lösen zu können.

Betrachtet man die Ergebnisse mit der Verwendung des PMI-Lexikons, so stellt man fest, dass dieses für Naive Bayes die Ergebnisse verschlechterte, während es die Ergebnisse von MaxEnt verbesserte. Es wäre interessant, den Grund dafür herauszufinden, um die Methoden noch weiter optimieren zu können.

Fazit

Alles in allem erzielte das Projekt zufriedenstellende Ergebnisse.

Für eine spätere Version des Projekts ist geplant, Laplace-Smoothing funktionierend zu implementieren und eventuell die Laufzeit zu verbessern.

Der Grund für das Speichern der Daten in Dateien ist es, zu ermöglichen, später mit den Daten weiterarbeiten zu können und Datenverlust zu vermeiden. Für eine allgemeine Version des Projekts ist es jedoch benutzerfreundlicher, eine GUI zu erstellen, die es möglich macht, eigene Datensätze analysieren zu lassen und die zu verwendeten Methoden auszuwählen, sowie wie diese analysiert und visualisiert werden sollen.

Im Projekt fiel auf, dass im Programm mehrmals sehr lange Listen erstellt wurden, bei denen der Großteil der Werte „0“ waren. Eine Lösung dafür ist die Verwendung von Sparse-Embeddings. Zwar war die Implementierung zeitlich nicht mehr möglich, kann jedoch zukünftig umgesetzt werden und führt eventuell auch zu einer Verkürzung der Laufzeit.

Da das Projekt bisher nur eine Basis für die Sentiment Analyse darstellt, ist ein Ziel die Optimierung der Algorithmen. Dazu zählt unter anderem die Beachtung von Negation, dem Sentiment von Satzzeichen und Emoticons, Ironie, Abkürzungen und Umgangssprache. Dafür muss zunächst weiter recherchiert und getestet werden, aber es sollte möglich sein, einige dieser Punkte zu umzusetzen.

Insgesamt bietet das Projekt die Grundlage für verschiedene Methoden der Sentiment Analyse, auf der in Zukunft weitergearbeitet werden kann.

Quellen

Jurafsky, D., Martin, J. H. (2008). *Speech and Language Processing*. New Jearsey: Prentice Hall.

Du Toit, J. (4. August 2015). *Cloud Academy Blog*. Abgerufen am 25. August 2016 von http://cloudacademy.com/blog/naive-bayes-classifier/

Mullen, T., Collier, N. (2004). *Sentiment analysis using support vector machines with diverse information sources*. National Institute of Informatics (NII), Japan.

Go, A., Bhayani, R. & Huang, L. (2009). *Twitter Sentiment Classification using Distant* Supervision. University Stanford, USA.