Universität Potsdam
Department Linguistik
Modul: Classification Approaches for Social Media Text
Kursleiterin: Tatjana Scheffler
SoSe 2018

---

Projektbericht:

# Hate Speech Classifier mit Sklearn & SpaCy

---

Name: Olha Zolotarenko
Studiengang, Fachsemester: Computerlinguistik, 4.Fachsemester
Matr.Nr.: 787894
E- Mail: zolotarenko@uni-potsdam.de
Abgabedatum: 24.09.2018

**Inhaltsverzeichnis**

## 1 Aufgabe und Daten

Aufgabe:
Das Ziel des Classifiers besteht darin, die *Hate-Speech-Tweets vs. Nicht-Hate-Speech-Tweets* (Binary Classification) zu unterscheiden lernen. Als persönliche Aufgabe gilt die eigenständige Ausarbeitung von dem Klassiker, besseres Verstehen von den einzelnen Bestandteilen und der Versuch, mit den Features für HateSpeech-Klassifizierer zu experimentieren. Dabei war mir wichtig, den Weg zu dem sentimentbasierten Klassifizierer zu entdecken, was ich näher in den Verbesserungsvorschlägen (siege Teil 4) beschreibe.

Daten
Manuell annotierte Daten von Shared Task *Germeval 2018*:

- 1688 - OFFENSE (1)
- 3320 - OTHER (0)

Um binär klassifizieren zu können, wurden die Tags manuell zu 1/0 konvertiert.

Bewertungsmaße:
Die Daten zeigen, dass man mit unbalancierte Klassen arbeiten muss. Deswegen ist hier die beste Entscheidung, die *ROC-AUC-Bewertung* (nicht sensitiv zu unbalancierten Klassen, da alle Grenzen zwischen den Klassen berücksichtigt werden) zu benutzen.
Außerdem wurde auch *Accuracy* (Genauigkeit) als Bewertungsmaß verwendet, da es mehr als Grad an Übereinstimmung mit den annontierten Daten gilt.

## 2 Beschreibung des Lernverfahrens

In der Tabelle 1 kann man die allgemeinen Informationen zu dem gebauten Classifier entnehmen.

Tabelle 1. Skizzierte Informationen zu dem Classifier

| | |
|---|---|
| **Vectorizer** | Als Hauptvektorizer wurde *CountVectorizer* benutzt. Es wurde aber versucht, mit dem TfidfVectorizer die Verbesserung zu schaffen. *TfidfVectorizer* hat die Ergebnisse nicht beeinflusst. Es besteht dennoch die Möglichkeit, diesen Vectorizer für spätere Tests in der Klasse doch zu benutzen (die unteren Zeilen im Main-Teil unter den entsprechenden von CoutVectorizer). |
| **Features** | BoW - *Unigramme*, max_df =1.0, min_df = 1 (keine Wörter werden ignoriert) |
| **Module** | Sklearn (Machine Learning), SpaCy (Tokenizer), Pandas (Strukturen&Analyse), Matplotlib (Graphische Analyse) |
| **Klassifizierer** | - LogisticRegression (LR)<br>- Support Vector Classification (SVC)<br>- Multi-layer Perceptron (MLP)<br>- MultinomialNB<br>- BernoulliNB<br>- RandomForestClassifier (RFC)<br>- K-Means Clustering (unsupervised learning) |

Bei dem Lernverfahren handelt es sich um *überwachtes maschinelles Lernen* – das gebaute Modell muss anhand von Trainingsdaten den *Label* (Klasse) vorhersagen lernen. Sind die Ergebnisse nicht befriedigend, so werden die zuvor ausgewählten Features verändert/angepasst, damit das System bessere Vorhersagen treffen kann. Das Algorithmus muss also von dem Input zum Output den richtigen Weg finden.

Kurz zu den benutzten Algorithmen [5]:

- *Logistic Regression* – passt die eine Linie (in unserem binären Fall) an das Dataset an, um die Klassen unterscheiden zu können;
- *Support Vector Classification* – finden die Grenze, die Klassen mit maximalem Abstand voneinander trennt;
- *Multi-layer Perceptron – mehrlagiges* künstliches neuronales Netz, das einen Eingabevektor in einen Ausgabevektor umwandelt, wobei die Gewichte der Verbindungen in den «hidden layers» das Ergebnis entscheiden.

- *Bayes-Methoden* (Multinomial, Bernoulli) – das Algorithmus lernt, welche Wörter, mit welcher Wahrscheinlichkeit in welcher Gruppe auftreten können (basiert auf dem Bayesschen Theorem);
- *RandomForestClassifier* – Klassifikation anhand von Entscheidungsbäumen: die Klasse mit den meisten Stimmen entscheidet.

## 3 Beschreibung des Systems

In dem Programm wurde eine Python-Klasse `Hate_Classifier()` erstellt, die die wesentlichen Schritte vor der Klassifizierung ermöglicht. Das Bild 1 veranschaulicht das Verfahren:

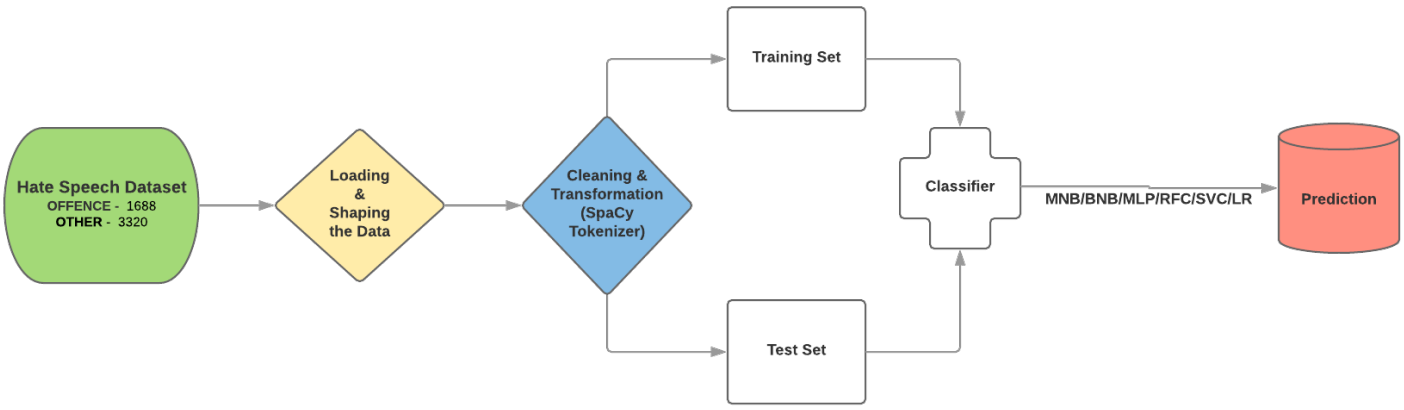


Bild 1. Von Daten bis Vorhersagen mit verschiedenen Klassifizierern und SpaCy Tokenizer

1. Unbearbeitete Hate Speech Tweets
2. Loading & Shaping Data
    - Manuelle Transformation von OFFENSE/OTHER Tags in binäre Variante 1/0
    - Data für Sklearn anpassen und in .csv Format umwandeln
3. Transformation & Cleaning (SpaCy Parser)
    - Stopwörter, Satzzeichen und Pronomen werden gelöscht
    - Alle Wörter sind in lower case umgewandelt
    - Alle Tags-Mentions (@…) werden gelöscht
4. Train & Test Datasets
5. Predictions zu den Klassen werden gemacht

Nachdem man die Vorhersagen für den Test-Set bekommen hat, wurden die Ergebnisse evaluiert.

## 4 Ergebnisse und Diskussion

Für die Evaluation der Ergebnisse von den getesteten Klassifizierern wurden *Accuracy* und (hauptsächlich, da die Daten umbalanciert sind) *AUC score* benutzt.

Die verwendete Scala für das Einschätzen von den AUC [2]:

- .90-1 = sehr gut (A)
- .80-.90 = gut (B)
- .70-.80 = befriedigend (C)
- .60-.70 = schlecht (D)
- .50-.60 = sehr schlecht (F)

Die getesteten Classifier haben wie folgt abgeschnitten (siehe Tabelle 2 und Bild 2):

Tabelle 2. Zusammenfassung der Ergebnisse von den Klassifizierern

| Classifier | Runtime in seconds | Accuracy on Train Data | Accuracy on Test Data | AUC score |
|---|---|---|---|---|
| **SVC** | 38 | 0.6570144782825761 | 0.6866267465069861 | 0.79 |
| **BernoulliNB** | 25 | 0.9078881677483774 | 0.7325349301397206 | 0.82 |
| **RFC** | 30 | 0.9692960559161258 | 0.7584830339321357 | 0.74 |
| **LR** | 26 | 0.9867698452321517 | 0.7524950099800399 | 0.80 |
| **MLP** | 89 | 1.0 | 0.7544910179640718 | 0.78 |
| **MultinomialNB** | 36 | 0.963554667998003 | 0.781437125748503 | 0.83 |

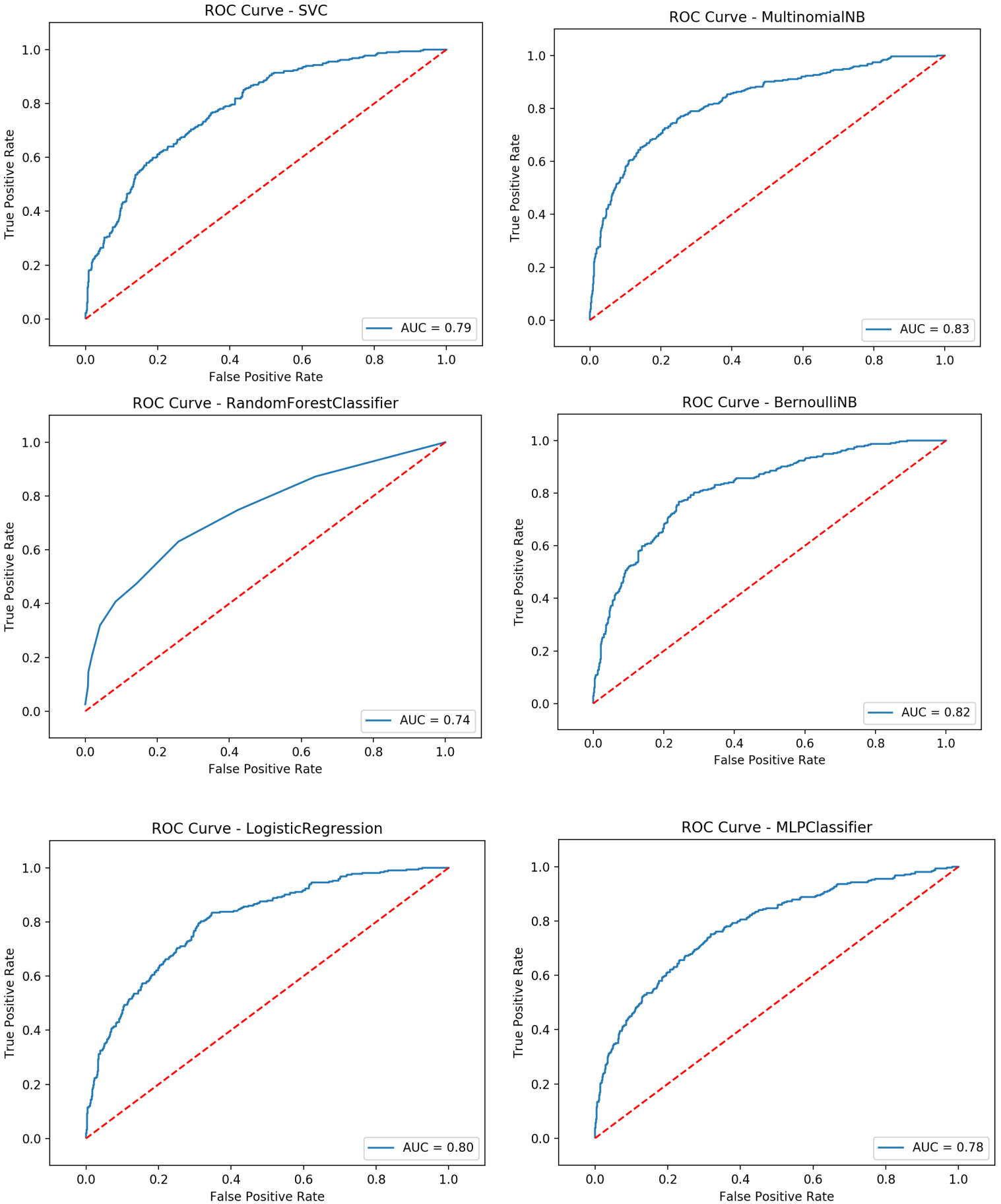


Bild 2. ROC-AUC Ergebnisse von 6 Klassifizierern

Wie man sehen kann, hat bei dieser Implementierung *Naive Bayes* (MultinomialNB) mit AUC von 0.83 gefolgt von BernoulliNB mit 0.82 am besten abgeschnitten. Den dritten Platz nimmt *Logistik Regression* mit 0.80 AUC score. Nach der oben vorgeschlagener Scala kann man die Ergebnisse als *gut* (B) bewerten – die drei besten Classifier sind ca. **30%** besser als random classification.

Allgemeine Beobachtungen:

1. Die Unigramme liefern die besten Ergebnisse. Beim steigern auf Bi- oder Trigramme fallen die Ergebnisse in AUC score auf 70% bis 60% entsprechend.
2. Das Entfernen von Antworten und Erwähnungen (@mustermann…) hat die meisten Klassifizieren um 0.1 AUC verbessert.
3. Das Entfernen von Zahlen in den Tweets hat die Ergebnisse von den meisten Classifiers deutlich verschlechtert. Dies liegt wahrscheinlich daran, dass Zahlen doch die wichtigen Features in Tweets sein müssen.
4. K-Means Clustering hat nur 55% erzielt und ist somit vom Training ausgeschlossen worden. Zitierend aus Wikipedia Artikel zu «K-Means Clustering», dass das Algorithmus Gruppen mit geringer Varianz und ähnlicher Größe bevorzuget, kann man vermuten warum: die HateSpeech Daten sind umbalanciert.

Verbesserungsvorschlag

Als weitere Schritte für die Verbesserung des Klassifizieren von HateSpeech-Tweets würde ich das Folgende vorschlagen:

1) Mehr *Klassifizierungsalgorithmen* testen, z.B. Convolutional Neural Networks (CNNs) (z.B. Keras) [4]
   - Andere Klassifizieralgorithmen können mit den ausgewählten Features bessere Performanz zeigen;
   - Besonders gute Ergebnisse bei der Satzklassifizierung und dem semantischen Parsen.

2) *Lexical Approach – Sentiment* ins Spiel bringen
   - Man kann eine Hypothese aufstellen, dass die Hate-Tweets (OFFENSE) im Vergleich zu ganz normalen Neutralen (OTHER) einen negativen Sentiment beinhalten. Das sogenannte l*exical approach* kann eine gute Lösung in der Sentimentanalyse sein (wenn man die Implementierung von Ricky Kim betrachtet [7], ist sie 15.31% besser, als unten erwähnter TextBlob Sentiment Tool);

- Emoticons und Akronyme (wie «LOL») als Features benutzen (bei VADER-*Sentiment-Analysis* z.B. möglich, siehe Punkt 3).

3) Andere *NLP-Tools* – *Textblob* (oder vielleicht ähnliche NLP Tools wie z.B *VADER-Sentiment-Analysis* (bis jetzt nur für Englisch vorhanden)) statt SpaCy ausprobieren
   - *TextBlob* ist ein anderer Modul zur Textverarbeitung. Nach vielen Quellen im Internet ist der Modul einer der Besten;
   - VADER stützt auf *lexical features* und hat große dicts annotiret mithilfe von Amazon Mechanical Turk, um Sentiment zu bestimmen. Vorteilhaft ist außerdem auch Erkennung von Emiticons.

4) Andere *Vectorizer* benutzen– z.B. *word2vec* (Alternative zu «continuous-bags-of-words») [4]
   - Word2vec kann die sogenannten «skip-grams» benutzen, die die Kontextwörter anhand von dem aktuell analysierten Wort vorhersagen können.

5) *GridSearchCV* als Estimator benutzen, um schnell die besten Parameter einzuschätzen.

6) Zuletzt natürlich – *mehr Daten* gewinnen und benutzen.

Fazit

Das Verfahren ist insgesamt ***ganz gut*** geeignet für die gestellte Aufgabe, kann und soll aber noch verbessert werden. Idealerweise könnten die oben beschriebenen Verbesserungsvorschläge mehr Licht auf das Verhalten von Hate-Speech Tweets werfen und in der Zukunft helfen, optimale Klassifizierer für diese Aufgabe zu bauen.

## Literaturnachweise


(1) JCharis Jesse, (12 Jun 2018). Natural Language Processing Tutorials Abgerufen von: https://github.com/Jcharis/Natural-Language-Processing-Tutorials/blob/master/Text%20Classification%20With%20Machine%20Learning%2CSpaCy%2CSklearn(Sentiment%20Analysis)/Text%20Classification%20%26%20Sentiment%20Analysis%20with%20SpaCy%2CSklearn.ipynb

(2) MOUTAI10, (24 Jan 2015). How to plot a ROC Curve in Scikit learn? Abgerufen von: https://datamize.wordpress.com/2015/01/24/how-to-plot-a-roc-curve-in-scikit-learn/

(3) Felipe, (07 Jul 2018). Visualizing Machine Learning Models: Examples with Scikit-learn, XGB and Matplotlib. Abgerufen von: http://queirozf.com/entries/visualizing-machine-learning-models-examples-with-scikit-learn-and-matplotlib

(4) Björn Gambäck, Utpal Kumar Sikdar, Using Convolutional Neural Networks to Classify Hate-Speech. Abgerufen von:

(5) Microsoft Azure, (18 Dez 2017). Auswählen von Algorithmen für Microsoft Azure Machine Learning. Abgerufen von: https://docs.microsoft.com/de-de/azure/machine-learning/studio/algorithm-choice

(6) Blinov P. D., Klekovkina M. V., Kotelnikov E. V., Pestov O. A., Research of lexical approach and machine learning methods for sentiment analysis. Abgerufen von: http://www.dialog-21.ru/media/1226/blinovpd.pdf

(7) Ricky Kim, (27 Feb 2018). Twitter Sentiment Analysis Part5. Abgerufen von: https://github.com/tthustla/twitter_sentiment_analysis_part5/blob/master/Capstone_part4-Copy3.ipynb