NLP Project: Authorship Detection of Tweets

Clemens Biehl, Daniel Wehner, Fabian Otto, Philipp Kapelle

***Abstract*—In our project we worked on the authorship detection for Twitter tweets. This report gives a short summary of the results which we could produce during the project. The usage of token n-grams and part-of-speech n-grams gave promising results, yet for cases with few training data more linguistically motivated features may yield better results than these token-frequency-based approaches.**

I. Introduction and Research Problem

Millions of texts and posts are published on social media platforms such as Twitter or Facebook every day. The identification of the authorship is often a crucial task in Natural Language Processing. This might be helpful when checking the authenticity of a post. In this project we therefore aim to classify Twitter Tweets to identify the authors of the tweets and try to answer the following questions:

- Which types of writing-style features are effective for identifying the authorship of online messages?
- Which classifiers perform best for identifying the authorship of online messages?
- To what extent can authorship-identification techniques be applied to online messages across different genres such as politics or sports?

To answer these questions we tried various combinations of features, classifiers and genres such as politics and sports and evaluated these combinations.

II. Twitter Crawler

In order to fulfill our research problem, we have to acquire many tweets that have been written by a single author in the English language. Therefor we need to identify Twitter accounts, that are operated by a single user and have a large amount of original tweets. To find these Twitter accounts, data from https://www.socialbakers.com has been analyzed.

The site lists the most followed Twitter accounts grouped by country of the user and his/her profession (e.g. Actor, UK). However, it does not distinguish between how many people operate the Twitter account, nor how many tweets have been written. Still one can assume that a Twitter account that is among the most followed accounts worldwide, is named after a single Person, is verified, contains a lot of personal information including original pictures of said person and does not state that it is operated by another person, contains a large amount of original tweets.

Therefore the list provided by the website has been the primary source of the data used. The Twitter accounts selected have been identified by the following steps. First four fields of profession have been selected ('Actor', 'Broadcast Star', 'Politics', 'Sport Star') due to a high density of single-author accounts. To get reliable English language tweets, USA and United Kingdom have been selected as the only two possible countries of origin. Also to get a fair distribution between American English and British English the top twelve accounts per country will be added as potential accounts to be crawled.

Twitter users have a timeline that provides all Tweets, Retweets and information about the account itself, such as total amount of Tweets tweeted. In this amount Retweets are counted as Tweets. During the process of identification the goal was set to get at least 20 distinct authors per category with at least 1000 original tweets each and an about equal distribution between US and UK citizens. Since the only information on the amount of tweets a user has sent includes Retweets, which contain text the user

TU Darmstadt, WS 2017/2018.

hasn't written, accounts with less than 3000 Tweets in total have been filtered out. Finally an extra category of Twitter accounts with less tweets and possibly not only English language tweets has been added with selected accounts that may be interesting ('FunMix').

After the accounts have been identified, each of them has been reviewed and determined, whether or not it is likely, that it is run by a single user. Also the total amount of tweets has been checked. Disqualified accounts then have been replaced by the next Twitter account from the list provided by https://www.socialbakers.com, that fulfills the requirements.

The list of users has been stored in a JSON file containing the names, countries and Twitter handles of the users. This JSON file was then read by the Twitter crawler, and each user's timeline requested from Twitter using the TwitterAPI and the Java library 'twitter4j'.

The timeline is divided into pages containing up to 100 Tweets, that each have to be requested individually. To limit the traffic and optimize the data output, a single page was requested, filtered locally for original tweets only and checked, if the 1000 Tweets were reached. If not, the process was repeated, which is the reason for minor differences between the amount of Tweets per user. At the end of the crawling step, all Tweets were saved as JSON files.

In total 100 Twitter accounts have been identified and crawled, resulting in 1000 to 1099 (mean 1043) original tweets per author.

III. Data Provisioning

As previously described, crawled tweets from Twitter will be analyzed. We focused on tweets of 20 English authors per category: politicians, sportsmen, actors and broadcast stars. For instance, we collected tweets from the following accounts of politicians:

realDonaldTrump, BarackObama, ChuckGrassley, RepJaredPolis, BorisJohnson, clairecmc, ChrisChristie, jahimes, jeremycorbyn, CarolineLucas, David_Cameron, BernieSanders, RonPaul, SpeakerRyan, mike_pence, DavidLammy, timfarron, Ed_Miliband, ChukaUmunna, tom_watson

Fig. 1. JSON representation of a tweet written by Barack Obama that was crawled from Twitter


```
{
   "in_reply_to_status_id_str":null,
   "in_reply_to_status_id":null,
   "coordinates":null,
   "created_at":"Mon Jan 15 14:46:02 +0000 2018",
   "truncated":false,
   "in_reply_to_user_id_str":null,
   "source":"<a href=\"http://twitter.com/download/
     iphone\" rel=\"nofollow\">Twitter for
     iPhone<\/a>",
   "retweet_count":367963,
   "retweeted":false,
   "geo":null,
   "in_reply_to_screen_name":null,
   "is_quote_status":false,
   "entities":{
      "urls":[

      ],
      "hashtags":[

      ],
      "user_mentions":[

      ],
      "symbols":[

      ]
   },
   "full_text":"Dr. King was 26 when the Montgomery
      bus boycott began. He started small, rallying
      others who believed their efforts mattered,
      pressing on through challenges and
      doubts to change our world (...)",
   "id_str":"952914779458424832",
   "in_reply_to_user_id":null,
   "display_text_range":[
      0,
      279
   ],
   "favorite_count":1393878,
   "id":952914779458424832,
   "place":null,
   "contributors":null,
   "lang":"en",
   "user":{
      "id_str":"813286",
      "id":813286
   },
   "favorited":false
}
```

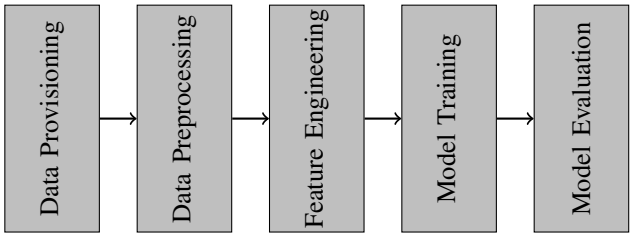

The number of politicians' tweets collected per account is 1,000, which makes 20,000 tweets in total. Thus, when guessing the authors randomly, we would get an accuracy of approximately 5 % (since the training data consists of tweets from 20 different authors). This is our baseline. We should at least have an accuracy better than 5 %.

IV. Pipeline

The general process for the authorship identification follows a usual NLP pipeline approach as depicted in Figure 2. The single steps will be described further in the following sections. The *Data Provisiong*

was described in the last section. The preprocessing depends on the features which are used, e. g. the contextuality measure requires part-of-speech tagging.

Fig. 2. Visualization of the pipeline built for the authorship identification. After providing the data by the crawler, the data is preprocessed. The feature extraction phase takes care of creating useful features for classification for the training of the model. The last step is the evaluation of the model.



Data Provisioning: For information see section III

Feature Engineering: Please see section V

Model Training and Evaluation: Please refer to section VI

V. Feature Engineering

The feature engineering phase is crucial for the success of the project since the performance of the classifiers depends significantly on the quality of the features used. Table I summarizes the features which we have taken into account when training the classifiers.

With emoticons playing a central role in social media they represent a good feature to consider. Therefore, the emoticon ratio (ER) is calculated (the number of tokens which represent emoticons divided by the total number of tokens in the text).

Other features like sentence features did not perform well since twitter texts are very short.

$$\text{ER} = \frac{\text{\# of emoticon tokens}}{\text{\# of tokens}} \tag{1}$$

Also very characteristic for specific authors is the number of hashtags they use when writing a tweet,

TABLE I
List of features which were considered for training the classifiers.

Feature	Explanation
Total number of chars	How long is the tweet
Emoticon ratio	Proportion of emoticons in the text.
Number of hashtags	How many hashtags are used in the tweet.
Word frequencies	Which words are used frequently
Lexical diversity	How rich is the vocabulary of the author?
Contextuality Measure	Score between 0 and 100 (0 very context dependent = many pronouns, adverbs, ...; 100 not content dependent = many nouns, ...)
Exclamation Ratio	How many exclamation marks are used?
Superlative Ratio	Proportion of superlatives in the tweet?
PastVsFuture	Ratio of verbs in past tense/present tense
SpellingErrorRatio	Ratio of spelling errors on all tokens

the word frequencies (does the author prefer specific words over other words) and the lexical diversity (also known as type-token ratio [TTR] which analyzes how many different words are used in the tweet)

$$\text{TTR} = \frac{V(N)}{N} \tag{2}$$

Other measures for lexical diversity/vocabulary richness:

$$\text{Yule's } K = C\left[-\frac{1}{N} + \sum_{m=1}^{m_{max}} V(m,N)\left(\frac{m}{N}\right)^2\right] \tag{3}$$

$$\text{Simpson's } D = \sum_{m=1}^{m_{max}} V(m,N)\frac{m}{N}\frac{m-1}{N-1} \tag{4}$$

$$\text{Herdan } V_m = \sqrt{\sum_{m=1}^{m_{max}} V(m,N)\left(\frac{m}{N}\right)^2 - \frac{1}{V(N)}} \tag{5}$$

$$\text{Sichel's } S = \frac{V(1,N)}{V(N)} \tag{6}$$

$$\text{Honore's } R = 100\frac{\log(N)}{1 - \frac{V(1,N)}{V(N)}} \tag{7}$$

$$\text{Brunet's } W = N^{V-c} \quad \text{usually } c = 0.17 \tag{8}$$

$$\text{Uber Index} = \frac{\log(N)^2}{\log(N) - \log(V(N))} \tag{9}$$

N	Length of text
$V(N)$	Size of vocabulary
$V(m, N)$	Number of words in N occuring m times
$V(1, N)$	Number of **Hapax Legomena**
$V(2, N)$	Number of **Hapax Dislegomena**
m_{max}	maximal frequency

Another feature of interest is the contextuality measure which indicates how context-dependent a text is. The contextuality measure produces values between 0 and 100. A value of 0 indicates a very context-dependent text which contains many pronouns, adverbs, etc. The higher the value the less context-dependent the text is (the text is then said to be *formal* as opposed to *contextual*). This is the formula which computes the score:

$$F = \frac{n + a + p + d - pr - v - ad - if + 100}{2} \tag{10}$$

n	noun frequency
a	adjective frequency
p	preposition frequency
d	determiner frequency
pr	pronoun frequency
v	verb frequency
ad	adverb frequency
if	interjection frequency

VI. Evaluation

We used accuracy to evaluate the classifiers bsed on a train/test split of 90/10 %, since our dataset is balanced and there is plenty of training data. As a basis we used the `WekaTwitterSentimentDemo` which we found in the DKPro TC GitHub Repository resulting in an accuracy of approximately $13\,\%_{\text{Naive Bayes}}$, approximately $8.5\,\%_{\text{Random Forest}}$ and approximately $8\,\%_{\text{Logistic}}$ (Used features in the demo: EmoticonRate and Number of Hashtags, Number of Tokens per Sentence). We used this as the baseline and added further features. The evaluation is based on a test set of 10% of the tweets per genre. The results for different features and classifiers are summarized in tables II (**Naive Bayes**), III (**Random Forest**) and IV (**Logistic Regression**) and in figures 3 (**Naive Bayes**), 5 (**Random Forest**) and 6 (**Logistic Regression**) respectively.

Fig. 3. Evaluation results for different feature setups. Results for **Naive Bayes** classifier

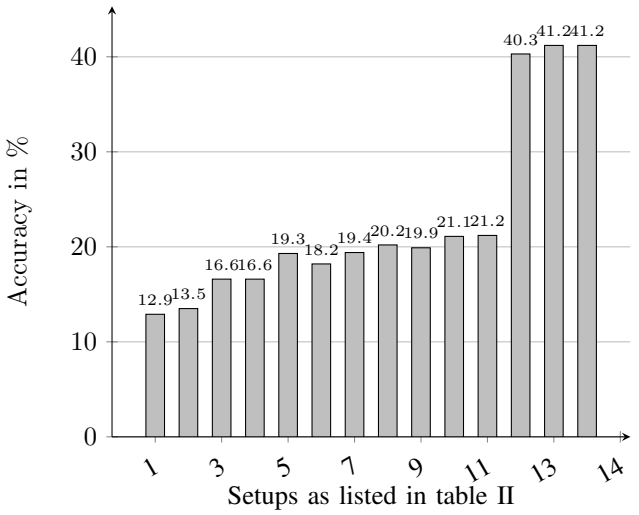


Figure 4 shows the performance of the Naive Bayes classifier using the features in row 13 of table II. The performance of the classifier using these features, which were engineered on the politics domain tweets, decreases on all other domains apart from the funmix domain. This is surprising since one would not expect politicians' style of writing tweets to be similar to the style of people who own fun accounts (such as *@Lord_Voldemort7*).
Training the Naive Bayes classifier across all domains yielded an accuracy of about 7 %, which is above the 5 % baseline but shows that the features do not generalize well across domain.

TABLE II
EVALUATION RESULTS FOR DIFFERENT FEATURE SETUPS ORDERED BY INCREASING ACCURACY. CLASSIFIER = **NAIVE BAYES**

Naive Bayes		
Nr.	**Setup**	**Accuracy**
1	`WekaTwitterSentimentDemo`	**12.88 %**
2	TTR (Type-Token-Ratio)	**13.48 %**
3	TTR + Contextuality Measure	**16.60 %**
4	TTR + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces	**16.59 %**
5	TTR + Contextuality Measure + Exclamation Ratio	**19.25 %**
6	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio	**18.19 %**
7	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture	**19.38 %**
8	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence	**20.22 %**
9	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence + Avg Nr of Chars per Token + Nr of Chars + Nr of Sentences + Nr of Tokens	**19.90 %**
10	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces	**21.06 %**
11	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces + Yule's K + Herdan V_m	**21.19 %**
12	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces + N-Grams (up to trigrams)	**40.26 %**
13	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces + N-Grams (up to trigrams) + POS N-Grams (up to trigrams)	**41.20 %**
14	TTR + Contextuality Measure + Exclamation Ratio + PastVsFuture + Avg Nr of Chars per Sentence + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces + N-Grams (up to trigrams) + POS N-Grams (up to trigrams) + SpellingErrorRatio	**41.20 %**

VII. DEEP LEARNING

Extracting the features is very tedious and it is more or less a trial and error process. There are many combinations of features that have to be taken into account and which have to be evaluated. A remedy for that is for example a **Deep Learning** approach. Such approaches have become very famous recently

Fig. 4. Evaluation results of Naive Bayes on different domains with optimal features. Results for **Naive Bayes** classifier

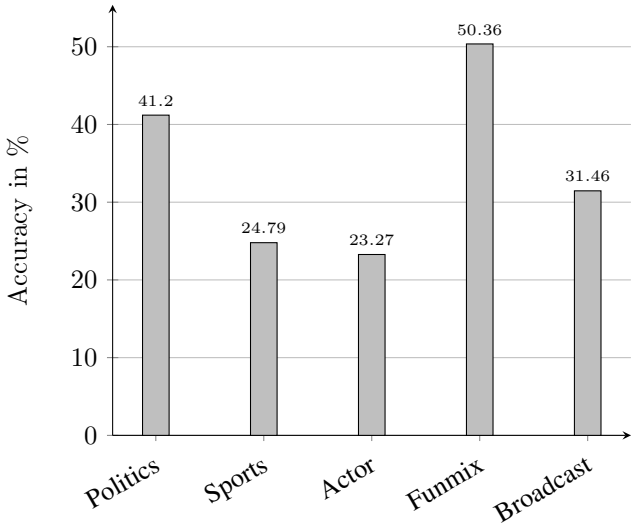


TABLE III
EVALUATION RESULTS FOR DIFFERENT FEATURE SETUPS ORDERED BY INCREASING ACCURACY. CLASSIFIER = **RANDOM FORESTS**

Random Forest		
Nr.	**Setup**	**Accuracy**
1	`WekaTwitterSentimentDemo`	**8.49 %**
2	TTR (Type-Token-Ratio)	**8.49 %**
3	TTR + Contextuality Measure	**8.49 %**
4	TTR + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces	**95.02 %**
5	TTR + Contextuality Measure + Exclamation Ratio	**8.49 %**
6	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio + PastVsFuture	**8.49 %**
7	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio	**8.49 %**
8	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio + Nr of Tokens + Character Features + N-Grams	**95.08 %**
9	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio + Nr of Tokens + Character Features + N-Grams + POS-NGrams	**95.03 %**

Fig. 5. Evaluation results for different feature setups. Results for **Random Forest** classifier

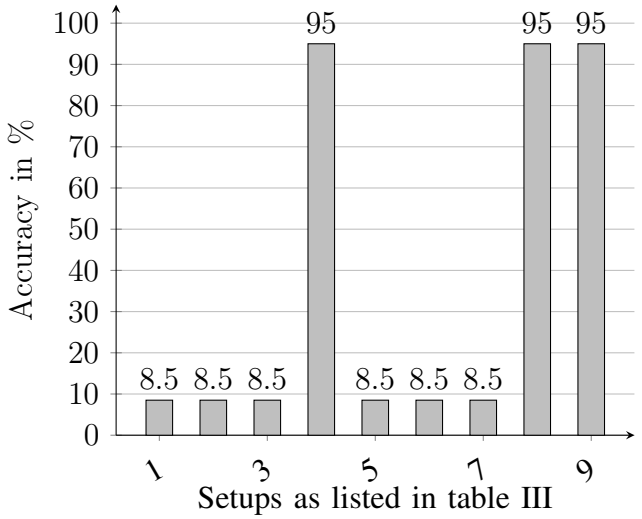


Fig. 6. Evaluation results for different feature setups. Results for **Logistic Regression** classifier

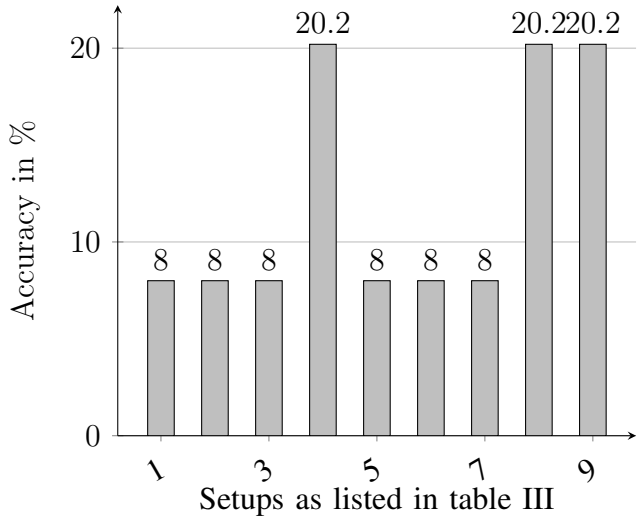


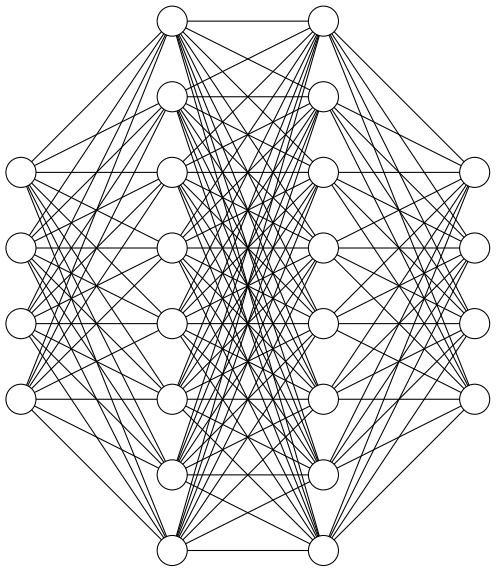
TABLE IV
EVALUATION RESULTS FOR DIFFERENT FEATURE SETUPS ORDERED BY INCREASING ACCURACY. CLASSIFIER = **LOGISTIC REGRESSION**

Logistic Regression		
Nr.	**Setup**	**Accuracy**
1	`WekaTwitterSentimentDemo`	**7.96 %**
2	TTR (Type-Token-Ratio)	**7.96 %**
3	TTR + Contextuality Measure	**7.96 %**
4	TTR + UpperCase + Alphabetic + Digits + WhiteSpaces + TabSpaces	**20.20 %**
5	TTR + Contextuality Measure + Exclamation Ratio	**7.96 %**
6	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio + PastVsFuture	**7.96 %**
7	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio	**7.96 %**
8	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio + Nr of Tokens + Character Features + N-Grams	**20.20 %**
9	TTR + Contextuality Measure + Exclamation Ratio + Superlative Ratio + Nr of Tokens + Character Features + N-Grams + POS-NGrams	**20.20 %**

and also for Natural Language Processing there are several application areas for such methods. Deep Learning is capable of automating this combersome process by making use of several layers that are responsible for feature extraction and transformation. The results of one layer represent the input of the next layers. Very often **Artificial Neural Networks (ANN)** are used in such cases.

We used a *keras* implementation of a convolutional

Fig. 7. A simple neural network with two hidden layers.



LSTM network which is provided with DKPro TC 1.0.0 and made the following modifications. We increased the batch size to 50, reduced the initial dropout probability to 0.10 and changed the loss function to *sparse_categorical_crossentropy* so that our categorical labels can be conveniently used (therefore the final dense layer with a sigmoid

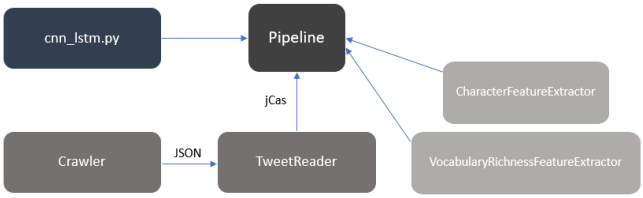
activation function had to be enlarged as well). The network consists of an embedding layer, a dropout layer, a convolution layer (64 filters, kernel size 5×5, stride size of 1, valid padding, 4×4 max pooling), an LSTM hidden layer with 70 units and a dense layer with a sigmoid activation function applied. For the embedding layer of this neural network using 50-dimensional Glove embeddings which were pretrained on Twitter data did not improve the accuracy, therefore the embeddings are trained from scratch. We use a *tensorflow* backend and the Adam optimizer implementation provided by this framework.

After 10 training epochs the network achieves an accuracy of above 95%. This is a very high score for the task of authorship identification that can only be achieved, because plenty of training data (tweets) for the chosen authors is available on Twitter. This might not always be the case in other real life applications. Another explanation might be that the model is overfitting. Since an exhaustive hyperparameter tuning is computationally very expensive we limited the experiments to the parameters mentioned above to achieve a good baseline accuracy for the neural network. Thus, the achieved accuracy might still be suboptimal and future projects can build upon this. Further experiments will have to examine how sensitive the neural network reacts to a reduction of training examples, further parameter optimizations and changes in the network architecture and whether the dropout probability has to be further increased to prevent overfitting.

VIII. Components of the Implementation

The implementation uses DKPro TC for the main processing pipeline and *jsoup* for the crawler. The Java class *Crawler* reads twitter posts, the Java class *TweetReader* reads the resulting JSON using the *gson* framework and adds the plain tweet text to the jCas to make it available for further processing. The *TweetReader* class implements the interface *TCReaderSingleLabel* and is used in the main class *Pipeline* as a DKPro TC reader. In the *Pipeline* class all further preprocessing steps are defined, different classifiers are included and the evaluation is performed with the *BatchTrainTestReport* of DKPro TC. In the *Pipeline* class the *Weka* implementation of the mentioned classifiers is used. Furthermore, experiments with a neural network implemented in the python script *cnn_lstm.py* were performed using the *keras* library which is integrated with DKPro TC version 1.0.0. Additional feature extractors are provided by other Java classes such as *CharacterFeatureExtractor* (for UpperCase, Alphabetic, Digit and WhiteSpaces frequency features) and *VocabularyRichnessFeatureExtractor*. An overview of the main components is given in figure 8.

Fig. 8. Overview of implementation components



IX. Conclusion

The experiments showed that character-based linguistic style features such as the frequencies of upper case, alphabetic, digit and white- and tabspace characters are good indicators for an author's style in writing tweets. Likewise the Contextuality Measure and PastVsFuture features increased accuracy. In contrast, sentence-based features such as the number of sentences or the average number of characters in a sentence decreased accuracy, which can be explained by the general brevity of tweets and the fact that tweets often do not include clear sentence boundaries. Some of the measures for lexical diversity/vocabulary richness did not contribute to the accuracy, therefore only Yule's K and Herdan V_m were used. Using N-Grams and Part-of-Speech N-Grams significantly improved the accuracy, presumably because they incorporate the author's specific vocabulary.

The highest accuracy of 95.08 % using a random forest classifier and the features shown in table III (column 8) is remarkably high for the task of authorship identification, but does not generalize well to other domains, since the vocabulary used for the N-Gram features is very specific with respect to the domain. Training a Naive Bayes classifier

across all domains indicated that the features do not generalize well across domains.

Future experiments will have to investigate if the features used here can be successfully applied in other domains. It can be assumed that with fewer training examples the linguistic style features become more important than e. g. N-Grams, since these would be too sparse to represent meaningful features with little data. Furthermore, deep learning has to be investigated further to make use an efficient implicit feature extraction during the classifier training.

The code as well as the responsibilities of the group members for this NLP project can be found in the Github repository: https://github.com/BoboDance/NLP4Web_Project.

REFERENCES

[1] Zheng, Rong and Li, Jiexun and Chen, Hsinchun and Huang, Zan. A Framework for Authorship Identification of Online Messages: Writing-Style Features and Classification Techniques. *Journal of the American Society for Information Science and Technology*, vol. 57, no. 3, pp. 378-393, 2006.

[2] Hout, Roeland and Vermeer, Anne. Comparing measures of lexical richness. *Modelling and assessing vocabulary knowledge*, pp. 93-116, 2007.

[3] Fissette, Marcia. Author identification in short texts. 2010.

[4] Green, Rachel M. and Sheppard, John W. Comparing Frequency- and Style-Based Features for Twitter Author Identification. *AAAI Press*, 2013.

[5] Heylighen, Francis; Dewaele, Jean-Marc. Variation in the Contextuality of Language: An Empirical Measure. *Foundations of Science*, vol. 7, no. 3, pp. 293-340, September 2002.

[6] Tanaka-Ishii, Kumiko and Aihara, Shunsuke. Computational Constancy Measures of TextsYule's K and Rnyi's Entropy. *Computational Linguistics* vol. 41, no. 3, pp. 481-502, September 2015.

[7] Tweedie, Fiona J. and BaayenHow, R. Harald. Variable May a Constant be? Measures of Lexical Richness in Perspective. *Computers and the Humanities* vol. 32, no. 5, pp. 323-352, September 1998.

[8] Bhatia Archna et al. TweetNLP, Carnegie Mellon. http://www.cs.cmu.edu/~ark/TweetNLP

[9] Jeffrey Pennington and Richard Socher and Christopher D. Manning. GloVe: Global Vectors for Word Representation. http://www.aclweb.org/anthology/D14-1162