Classification of Newspaper Articles

Patrick Schwarz
Graz University of Technology
patrick.schwarz@student.tugraz.at

Matthias Krawanja
Graz University of Technology
krawanja@student.tugraz.at

Moritz Wiesinger
Graz University of Technology
moritz.wiesinger@student.tugraz.at

ABSTRACT

Nowadays it is hard to determine if an article you read online is trustworthy or not. Many articles are copied over and over on the internet until it is impossible to find the source of the original article.

This project tries to look at articles of different newspapers to find some characteristics to differentiate between them. To find such characteristics, the main topic of the project describes the gathering, preprocessing and evaluating of datasets for the newspapers. The results show us that it is possible to find differences between newspapers. The harder part was to find characteristic differences between renowned newspapers. We think this is because the language used in them is nearly the same.

ACM Reference Format:
Patrick Schwarz, Matthias Krawanja, and Moritz Wiesinger. 2018. Classification of Newspaper Articles. In *Proceedings of July 2018, Graz, Austria (Knowledge Discovery and Data Mining 2),* Patrick Schwarz, Matthias Krawanja, and Moritz Wiesinger (Eds.). ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

There are several advantages and disadvantages that the rapid exchange of information through the internet brings along. It provides us with the ability to know everything right after it happened. The question we must ask here is "Can we trust this information?".

To give an answer to this question, it is essential to know the source of the information. One can argue that some sources are more legitimate than others. Surely there are more aspects in defining valid information but for this project we concentrate on finding the source of information.

The scope of this project is to find the original newspaper of articles found on the web. For this purpose the articles of the newspapers "Der Standard" [1], "Die Presse"[2] and "Kronen Zeitung"[3] have been used. By crawling the official website of these three newspapers we gathered information to train a machine learning algorithm. The computer then classifies which newspaper authored the newly shown article. Different preprocessing techniques allowed us to improve the actual data of the original information.

After all these steps it was possible to find differences between the newspapers. Furthermore, we found out that some newspapers can be easily classified while others will not show any noticeable difference.

[1]https://derstandard.at
[2]https://diepresse.com
[3]https://www.krone.at

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Knowledge Discovery and Data Mining 2, 2018
© 2018 Copyright held by the owner/author(s).

2 DATA ACQUISITION

In this section we will go over the steps needed to produce reasonable training data. Furthermore, we will explain the basics on the features used to differentiate between newspapers.

To gather many articles three major newspapers from Austria were chosen:

- Der Standard
- Kronen Zeitung
- Die Presse

These newspapers were selected because their perceived political orientation is different.[2] To acquire the data from the newspapers, a crawler had to be written which extracts the required information from the newspapers' websites.

The crawler is written in Python and uses a web crawling framework called Scrapy [4].[1] The framework performs the web requests, handles possible redirections, network issues and supports the definition of individual spiders. Spiders can be used to define the detailed information extraction procedure depending on either the kind of information expected on the page or the content type received.

As crawling was not the focus of this project but only a necessary preliminary step rather simple web crawlers tailored to the selected newspapers were written. The crawlers extract the news article's headline, the lead text, the main content, the date it was published and possible sub headlines. The reason why more information was gathered is that at the time of writing the crawler it was not clear which features will be used for classification. Furthermore, it did not affect the crawl runtime in a noticeable manner.

For news articles loaded from "Der Standard" and "Die Presse" the crawler also extracts the category and subcategory the article is in, although this information was not used during the subsequent steps. Most articles contain an image with a description which is crawled too. The gathered data is parsed by Scrapy and can be queried with Cascading Style Sheet (CSS) or XPath selectors. To avoid any IP black listing of the crawlers, parallel processing of the queue was disabled. Instead a delay of 0.7 seconds between each request was defined.

Where possible the publish date of the articles was limited to April 2018 as we wanted to use news articles from different papers about the same topic which is less likely if the articles were written at completely different points in time.

[4]https://scrapy.org

2.1 Kronen Zeitung

For this newspaper we tried several approaches. First, we tried to use the news feed provided by the website. The problem was that the news feed just delivered the last 20 articles published. Since we couldn't find any other entry point to get more than just 20 articles of the past we had to instead use the ID that was shown in the link to an article. These article IDs are increasing and therefore easy to find but not all of them are reachable. The unreachable URLs are probably due to deleted articles. Even though it could have been possible that the site itself would ban because of calls to unreachable content it did not and we could collect 11,683 articles by iterating over 65,000 IDs.

2.2 Der Standard

The crawling for this newspaper was based on the archive page. The archive lists all articles published on a given day. So as a first step all days of the month to be crawled are loaded and the links to the actual articles are enqueued for the main crawling step. With this approach all articles published in April 2018 were crawled which were a total number of 5114.

2.3 Die Presse

This newspaper does not provide an archive where all articles are listed but they do offer a search page. Requests that do not contain a valid session ID cookie receive a "No results found" page as answer independent of the search parameters. To circumvent this issue a valid session cookie value was copied from a browser and added to the crawl requests. The search form allows the search keyword to be empty which will return all articles published in the given time range. The result is split into several pages so each result page must be queried. This way 3596 articles from April were fetched.

3 PREPROCESSING

In this section, we cover the steps we took to further streamline our data to make it ready for training.

3.1 Obvious Indicators

Before the crawled data could be used for machine learning tasks it needed to be preprocessed. One of the last things we noticed but one of the first things to mention here is the obvious indicators for a newspaper. These are for example the name of the newspaper inside an article or a link to their website that is referenced somewhere. These indicators had to be removed because we feared that the algorithm would just take them to classify the articles and that is not what we intended to achieve in this project.

3.2 Special Characters, Punctuation and Whitespaces

Instead of using a black listing approach where all special and weird characters are black listed one by one and then removed from the dataset using some string replacing technique we chose a different path. Our solution was to white list everything we wanted to be left inside the dataset and remove everything else using regular expressions. We found that this was a far easier solution than the other way around.

This white listing of content included of course the normal German alphabet (including "Umlauts") and the Arabic numbers. But in addition to that we found that many words in the current news have special characters such as the words "Erdoğan" or "Orbán". So to keep the original words whole we also included a lot of accented characters into our whitelist.

In addition to white listing characters, we also removed all punctuation from the articles. We didn't make special cases for questions or exclamations in our algorithm, so this was a useful step to further simplify our dataset. Furthermore, we also replaced "-" in between words with normal whitespaces and removed content inside brackets of any kind. We also removed any kind of special Unicode whitespaces, multiples of whitespaces or newlines and standardized them all with normal single whitespaces to get a cleaner dataset.

The resulting function for white listing and erasure of other unwanted characters can be seen in the code listing below.

```
def remove_unwanted_symbols(line: str):
 ret = line

 # To lower case
 ret = ret.lower()

 # Remove unwanted symbols and unicode whitespaces
 ret = re.sub(r'[-/\u00a0\u1680\u180e\u2000-\u200b
              \u202f\u205f\u3000\ufeff]', ' ', ret)

 # Remove everything inside brackets
 ret = re.sub(r'[(\[{].*[)}\]]', '', ret)

 ret = re.sub(r'[^a-zöäüßáćšžíõøčàâçéèêëîïóòôûúùÿðğñæœ ]+',
               '', ret, flags=re.IGNORECASE)

 return ret
```

3.3 Further Improvements and Normalization

To make the training set even better we decided on taking training articles from the same timeframe so that we potentially have more articles about the same events from different newspapers. Additionally, we took the same number of articles from each newspaper to reduce overfitting for one newspaper.

After all mentioned preprocessing steps, the whole training set consisted of 9000 articles with a vocabulary of around 156000 different words. To improve the information even more we had to normalize our vocabulary.

3.4 Lemmatization and Stemming

The next step in our preprocessing concerns lemmatization where we are trying to find the right base form and stem of each word. This allows us to recognize words with the same meaning because of the same stem. As an example, the word "apples" gets reduced to just "apple", the word "swimming" gets reduced to "swim" and the words "am", "are" and "is" get reduced to "be".[8]

For this purpose, we are using a python lemmatization framework called spaCy. [7] This package also provides a German language pack which consists of word stems and their modifications in the German language. After using spaCy to lemmatize our training

data, the whole vocabulary got reduced to 98000 different words which means a lot of words have not been recognized as the same word until now. The number of words shows us how important this preprocessing step is.

3.5 Stop Words

With the identification of the word stems comes also the erasure of stop words. These are extremely common words which do not help us gain any important information out of the articles.[6] The spaCy framework helps us with the erasure of the stop words as well. A list of common words comes with the German language pack and with a simple `if` statement, one can check if a word is a stop word and remove it from the dataset.

After this preprocessing step we reduced our vocabulary to 96000 words. Even though the vocabulary itself did not get reduced much in overall, the word count was reduced massively.

To give an example: Without removing stop words, the dataset for "Der Standard" had a total word count of approx. 1,8 million words. After removing stop words, it got reduced to 887635. So the amount of words in the dataset got reduced by half.

3.6 Labeling

Now we concentrate on preparing the dataset for use with the FastText framework by Facebook.[3] This framework and how it is used for this solution will be explained in Section 4.

To prepare our data for the framework we had to add labels for every article. They were in the following form:

- `__label__krone`
- `__label__derstandard`
- `__label__diepresse`

These tags were added to inform the learning algorithm that an article belongs to a certain newspaper. The fastText library also allows to include more than one label for further classification but in our case this feature is not needed since we just want to classify the authoring newspaper of an article.

3.7 Finishing Touches

To finish the preprocessing for the framework we put each article together with its label on one line of a text file. Each line then represented a single article and its corresponding tag. The whole set of articles then was split into training and test set with a separate script file. The training set was used to let our algorithm learn how to classify the given newspapers and consisted of 3000 articles per newspaper. The test set was used to test the performance of the learned model and consisted of 900 articles from every newspaper.

3.8 Further Improvement Attempts

In the end we tried to improve the performance and word count of our dataset even further with another preprocessing step.

We split the articles into sentences because we did not want our algorithm to decide on the length of the article but on the characteristics of the language used. Unfortunately, this step turned out to decrease our test result and we reversed that change.

4 LEARNING PHASE

Once the data was acquired and preprocessed, a classifier had to be trained on the given data. For this task we used a text classification and word representation tool called FastText developed by the Facebook Research Group.

FastText can be used for two major scenarios: First, it can be trained on labeled data and after that the model can be queried to which label the text in question most likely belongs. The second major aspect of FastText is to let it learn details about word representations. This way, details about synonyms and hidden language structures can be learned which in turn may improve results for subsequent text classification problems.

FastText was designed by Facebook to work with big sets of data with several classes each and therefore is highly optimized to deliver acceptable model training times. This optimization is accomplished by utilizing hierarchical structures instead of flat ones where possible and sharing common information across labels as much as possible too. [4]

For this project, FastText was trained on labeled data where the label describes the newspaper from which the article was crawled. FastText supports multi-classification but this was not needed here as each article belongs to exactly one source newspaper. The FastText authors provide generic pre-trained models for more than 150 languages including German, but these were not used as those models do not contain information about our classes - the newspapers. Instead, our own model was trained using a supervised approach.

The supervised learning can be fine-tuned by adjusting the learning rate, the number of n-grams to be used and the epoch. The epoch defines how often each sample is processed during the training phase. According to the documentation the default value for the epoch is rather small so that each sample is seen only four times. [5] The higher the epoch value, the more often each sample is considered during the training. The learning rate specifies how much the model should be adjusted during each iteration. The suggested rate is between 0.1 and 1, a value of zero would mean no learning at all. Another important setting for the training is the number of words considered at once - the n-grams. The suggested range for this setting is between 1 and 5. This is especially important if the semantic of word combinations should be considered.

To find the ideal combination of these parameters we iterated over the possible combinations and found out that a low learning rate of 0.1, a large epoch of 100 and using bi-grams delivered the best results. Due to the size of the dataset this takes quite some time. To improve FastText's training performance another loss function was set. Instead of the regular softmax function a hierarchical softmax function was set as suggested in the documentation as performance tweak. [5] As no major performance reduction was noticed by this tweak we used that for all iterations. A possible reason for the good performance of bi-grams is the fact that this covers word combinations which are influenced by the writing style of the individual author.

5 PREDICTION PHASE

This section will cover the classification process of the machine learning algorithm.

	Die Presse	Kronen Zeitung	Der Standard
Die Presse	276	97	227
Kronen Zeitung	103	448	49
Der Standard	265	46	289

Table 1: Confusion matrix for the validation dataset.

After the model was trained the resulting precision and recall values for the testing dataset could be extracted. Since we only classify our articles by one label, the values for precision and recall are equal in every case. The results we achieved with our main implementation were a precision/recall value of 56%.

The computed confusion matrix for our test dataset can be seen in table 1. The matrix shows us that we can differentiate between "Kronen Zeitung" and the other two newspapers pretty well but "Die Presse" and "Der Standard" do not manifest any major differences. The model fails to differentiate between those two newspapers. The newspapers offer as much wrong classification as there are right classifications. This would be as good as picking the newspaper randomly.

This raises the question if it is even possible to classify any newspaper with the same amount of advanced language. To exhibit this problem further research would be necessary.

6 CONCLUSION

To assess the quality of sources of journalistic work gains importance in times of rising popularity of terms like 'Fake News'. Furthermore, unquoted reuse of such work can be observed on blogs. By inspecting features like writing style, average sentence length, the vocabulary used and others it might be possible to find out the original source of an article.

6.1 Approach

This project tried to inspect if well-known Austrian newspapers can be distinguished. To do so, articles from "Der Standard", "Die Presse" and "Kronen Zeitung" were crawled, preprocessed and fed into a text classifier.

To crawl the articles a framework called Scrapy was used. To increase the likelihood that the articles cover the same events it was tried to fetch articles from April 2018 only. For "Kronen Zeitung" this was not possible as they neither offer an article search page nor an archive.

To improve the quality of the dataset a few preprocessing steps were applied before feeding the data to the classifier. During this preprocessing special characters were removed, lemmatization of words was applied and common German words were excluded as these are not beneficial for the classification. For this task, the Python framework spaCy was used which already provides a German model which was trained on Wikipedia articles. The model is used to recognize stop words and gives the base information needed for lemmatization. During this step the amount of data which has to be fed into the classifier was reduced dramatically which in turn improved the performance of the subsequent steps.

For the actual learning a tool developed by the Facebook Research Group called FastText was used. The available pre-trained models were not taken but instead the acquired labeled data of the newspapers built the basis to train a new model. For this, the best combination of parameters was determined in an iterative manner. The parameters consist of a low learning rate, a high epoch which means that individual samples are considered more often and bi-grams were used instead of uni-grams. This way the order of words is important. It is noteworthy that the removal of stop words had a direct influence on the bi-grams generated at this step.

6.2 Results

Most of the results of table 1 show us that a classification of different newspapers is possible. Not all papers can be differentiated but we can tell if the source of an article originates in a boulevard newspaper or in a more sophisticated newspaper.

6.3 Future Work

This project considers the vocabulary used in an article as well as the direct neighbors of words. To improve the accuracy of the classifier, more general aspects of news articles could be considered too like average length of articles, sentences or even words. Furthermore, it could be beneficial to generate a vocabulary set which is common to a newspaper but not common in general. This way, if such vocabulary is found it would be a strong indicator for a given label.

Another aspect not elaborated in detail in this work is the structure of sentences. The tool used for lemmatization (spaCy) can classify tokens to nouns, verbs, adjectives and so on. At first sight, this worked reasonably well, so by looking in detail how sentences are structured, hints regarding the author of the article could be extracted as well.

REFERENCES

[1] Scrapy developers. 2018. Scrapy Tutorial. https://doc.scrapy.org/en/latest/intro/tutorial.html.
[2] eurotopics. 2017. Dominanz des Boulevards. https://www.eurotopics.net/de/149419/oesterreich-dominanz-des-boulevards.
[3] facebookresearch. 2018. Library for fast text representation and classification. https://github.com/facebookresearch/fastText/tree/master/python.
[4] Facebook Inc. 2018. fastText. https://research.fb.com/fasttext/
[5] Facebook Inc. 2018. Text Classification - fastText. https://fasttext.cc/docs/en/supervised-tutorial.html
[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press. https://doi.org/10.1017/CBO9780511809071
[7] spaCy developers. 2018. Code examples - spaCy. https://spacy.io/usage/examples.
[8] Daniel Tunkelang. 2017. Stemming and Lemmatization. https://queryunderstanding.com/stemming-and-lemmatization-6c086742fe45