
University of Applied Science

# Project: NLP

Task 1: Sentiment Analysis on Movie Reviews

Project Report

Author: Slavka Fersch

Matriculation Number: 92106749

Study-branch: Applied Artificial Intelligence B.Sc.

Tutor: Anna Androvitsanea

Delivery Date: October 24, 2023

## Contents

## List of Figures

## List of Tables

## 1 Introduction

In this work, the implementation of sentiment analysis for movie reviews will be discussed. To retrieve the sentiment, methods of Naive Bayes, Support Vector Machine (SVM) and Deep Learning will be used. In the beginning, the used dataset will be examined in detail. In Chapter 3, the attempted and conducted preprocessing steps will be covered. The methods of Text Vectorization will be discussed in Chapter 4 and Chapter 5 will examine the initialized approaches along with visualizations of the training results. The resulting test metrics of the utilized methods will be discussed and compared in Chapter 6. Limitations and future work is covered in Chapter 7 and a conclusion is presented at the end of the paper.

## 2 The Dataset

The dataset used in this project is known as the *Large Movie Review Dataset* provided by the Stanford AI Lab (Maas et al., 2011) and is publicly available. It contains 50,000 movie reviews along with binary sentiment polarity labels, besides another 50,000 unlabeled documents. For this project the latter was discarded, since supervised learning methods had to be implemented.
According to the authors of the dataset, there is a maximum of 30 reviews per movie to reduce correlation between ratings. The given test and training sets contain a disjoint set of movies, to prevent distortion by memorizing movie-unique terms. Hence, the segmentation of training set and test set were not manipulated to ensure an unadulterated result.
For developing the models the training set was further split into a training set and a validation set to evaluate the performance of the different algorithms implemented in this project. The test set provided by the original authors only was used as a final evaluation of the generalization of the developed models. This resulted in 25,000 files with 20,000 samples for training and 5,000 samples for validation.
There is several other information in the dataset, like the Uniform Resource Locators (URL) to the respective reviews of the Internet Movie Database (IMDb), already-tokenized bag of words (BoW) features and expected ratings for the tokens, computed by Potts (2010), which will not be further considered in this report.
Furthermore, the reviews of the dataset contain knowledge not only about a positive or negative sentiment, but also about the ranking of the movies. One consideration taken into account was to use this data as additional classification information, but was later discarded due to the lack of generalization the model has shown. Additionally, an adaption of another dataset with the implemented algorithms would have been more difficult to implement.

## 3 Preprocessing

After loading the training data, the overall structure of the reviews were investigated to decide for further cleaning approaches of the samples and decrease the complexity of the datasets. To reduce the unimportant information of the data, several attempts were implemented, along with regular expressions to remove certain patterns. Among the first glance, there were several samples containing a Hypertext Markup Language (HTML) tag for bold text, namely $<b/>$. Since it was not certain if there were other HTML tags present in the samples, all tags with this format were removed.

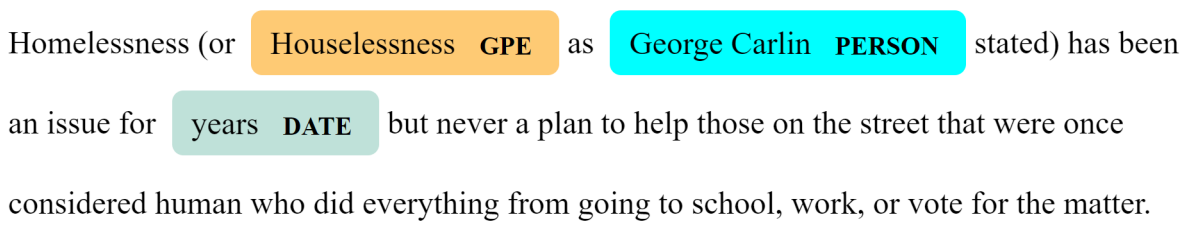
In the English language word sequences expressing negations are often concatenated into a single word, like for example *will not*, is known to be abbreviated with *won't*. Since these negations can influence the accuracy of a sentiment analysis model, the sequence *n't* was substituted with the term *not*.

### 3.1 Part of Speech Tagging and Named Entity Recognition

While evaluating the highest TF-IDF values, which will be introduced in Chapter 4, of the overall content, it appeared that the model may be biased by the names of actors and film directors as well as the movie titles. These entities were not clearly extractable with methods like regular expressions, as they were not located in the same position of the reviews or had a special format, Named Entity Recognition (NER) and Part of Speech (PoS) Tagging were tested on the dataset to filter and remove these terms and increase the ability of the models to generalize.
For this, the PoS Tagging module of spaCy library (Honnibal & Montani, 2017) was initialized first and all tokens with the tag *PROPN* were extracted in a sample of twenty test files. Although this approach looked very profitable at first glance, the overall dataset still contained much unnecessary information which might lead in a lack of generalization of the later implemented algorithms.
Therefore, also the NER functionality of spaCy was tested on the training dataset. With these module, it was possible to clearly visualize the content of the samples and which entities might be unnecessary and therefore should be discarded.
While there were many different scenarios tested, from removing only single entities up to removing all identified entities, the decision was made to remove only entities with the tag of *PERSON*. This lead to remaining some names of persons in the dataset which could not be identified by the algorithm, along with the movie titles. But since the other approaches tend to remove too much words which would not be labeled as an entity by a human, while keeping some other identifiable entities in the samples, this procedure was considered as a suitable trade-off, as most of the names could be identified in the samples under review. Figure 1 shows an illustration of one of the samples with positive sentiment in the training dataset with missing identification of terms like *school*, wrong identification of the term *Houselessness* as *Geopolitical Entity* and correct identification of the named entity *PERSON*.

**Figure 1**

*Example of Named Entity Recognition.*




### 3.2 Stemming, Lemmatizing and Stopword Removal

In the same sequence of the aforementioned preprocessing approach, stemming and lemmatizing were tested, along with removing so called stop words.
With stemming one obtains a root word, which does not necessarily have to result in a actually existing

word. The advantage of this method is that all parts of speech are reduced to their root form, which would allow a considerable reduction of the text corpus and therefore clean up the data set. On the other hand, some words lose their meaning and this might reduce the accuracy of the results. Although this method was tested, it was discarded at an early stage because the meaning of some words could no longer be identified. Therefore, Lemmatizing was used, where a root word is extracted into the respective dictionary form, thus retaining its meaning and making the results better interpretable in terms of natural language (Kulkarni, 2019, p. 54).
Two commonly known modules were tested to achieve the best result, the WordNetLemmatizer of the nltk library and the lemma module of spaCy. While the former had a faster computing time, considering all parts of speech and reducing the text to lemmas was more simple to implement with spaCy, since in that library it is already considered that different parts of speech have to be lemmatized according to different rules. Hence, lemmatizing with the spaCy library was implemented.
The text was further cleaned by removing stop words. For this purpose, the english stop words of the nltk library were used and extended iteratively with words that did not contain any significant information but had a high score of BoW and TF-IDF, like the words *movie* or *film*, which can be considered obsolete since the dataset is about movie reviews only. Also, words that may improve the accuracy of the later implemented models were removed from the set of stop words, like *no*, *not*, *nor* and *again*. To improve the computational effort, the list of stop words was transformed into a set.

Finally, all samples were transferred into lowercase, tokens with single characters only were removed and only letters were retained in order to remove punctuation.
All aforementioned preprocessing steps were conducted on all text files of the training and testing dataset and stored into Comma-separated values (csv) files. Only later, when using the tensorflow library, all preprocessed text files were additionally saved into new directories, since that library only allows customized preprocessing steps to a certain amount.

## 4 Text Vectorization

To enable computer algorithms to process text data, the text has to be converted into numerical values. For this, the approaches with BoW and TF-IDF were used for the machine learning methods, while for the deep learning model the module *TextVectorization* of tensorflow was used as input layer. Both methods will be further discussed in Chapter 5.
The method of Bag of Words (BoW) refers to whether a word occurs in a document and can be used either binary or additive. Although it does not consider the order of the words in a document neither their semantic, it is a simple method to convert texts into numerical data (Singh, 2022, p. 192).
One major drawback of BoW is the basic blunt count of all occurring words, even if they occur only once in the entire corpus, which leads to highly sparse vectors. Term Frequency-Inverse Document Frequency (TF-IDF) tackles this problem and provides an more informative approach. It relates the occurrences of a word compared to the number of documents it appears in, therefore normalizing the frequency the word occurs in the corpus based on all other documents (Zong et al., 2021, p. 35).
The TextVectorization module of tensorflow on the other hand is considered as a layer for neural networks, which transforms strings into vocabulary indices. The algorithm determines the frequency of individual string values and creates a vocabulary from them (Martín Abadi et al., 2015).
In this work, all described text vectorization methods were used and tested. While the machine

learning methods showed better performance with TF-IDF and therefore that vectorization method was used on the final application, for the deep learning approach several settings for the TextVectorization layer were tested.

## 5 Sentiment Analysis

Sentiment analysis is a NLP task, which is a ”form of opinion mining where algorithms detect sentiments expressed about features of products, services, etc.” (Loukanova, 2021, p. 163).
Since the datasets reviews were labeled and considered to be *positive* and *negative* only, the algorithms were implemented as a supervised binary classification task. Several algorithms were tested, in this chapter three of them will be discussed: Naive Bayes, Support Vector Machine and Deep Learning.

In the beginning, the models were tested with only subsamples of the dataset and only in the subsequent steps the full dataset was used to evaluate the models and tune the hyperparameters.
Since the computing time of the SVM algorithm increased with the number of input features, the maximum features of the vectorizers were set to 2000. This led not only to faster computing, but also to better results on the other machine learning methods.

### 5.1 Naive Bayes

Among the different naive bayes algorithms the library scikit-learn provides, the Naive Bayes classifier for multivariate Bernoulli models, in the following named as BernoulliNB, showed the best accuracy on the validation set. This Bernoulli model considers only the presence or absence of terms in documents and does not take the term frequency into account (Aggarwal, 2022).
To develop an optimal model, several parameters were conducted. Models were build with different ranges of n-grams along with different vectorizer and hyperparameters and compared based on the resulting accuracy on the validation set, as shown in Table 1.

**Table 1**

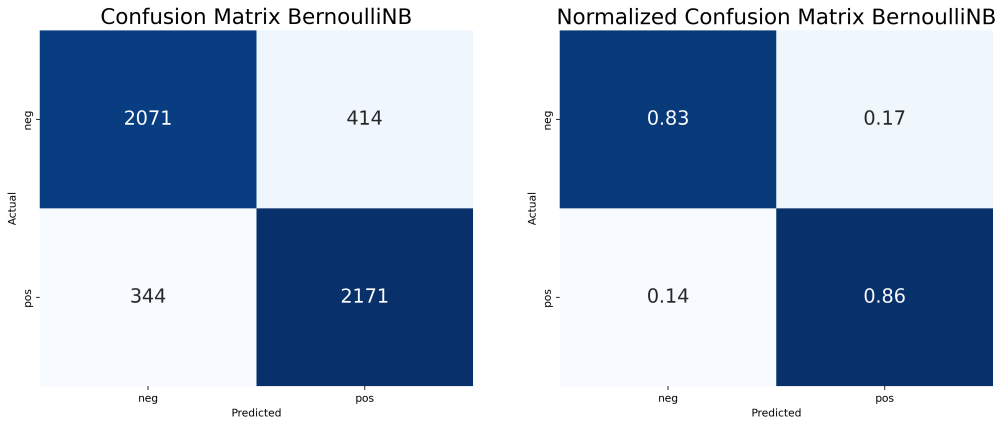
*Comparison of validation accuracy for BernoulliNB with different n-grams (smoothing).*

| Vectorizer \ n-grams | unigram | unigram & bigram | bigram |
|---|---|---|---|
| BoW | 0.848 | 0.844 | 0.774 |
| TF-IDF | 0.848 | 0.844 | 0.774 |

While the choice between the text vectorization methods did not impact the prediction accuracy, the comparison of different n-grams showed that unigrams result in the highest accuracy. These parameter were tested with and without smooting, which did not yield different results.
The confusion matrix of the validation data with the BernoulliNB model with smoothing, computed on unigrams vectorized with TF-IDF is shown in Figure 2.

**Figure 2**

*Confusion Matrices of BernoulliNB with raw counts (left) and normalized (right).*




### 5.2 Support Vector Machine

Support Vector Machines (SVM) solve classification problems by determining a optimal hyperplane which seperates the input features by their classes and thereby maximize the ability to generalize. They can be used to solve linear and nonlinear problems by mapping features to a higher dimensional space with the aid of kernel functions (Abe, 2005).
Like with the BernoulliNB algorithm, several options were tested with the SVM module of the scikit-learn library - more exactly the Support Vector Classification module. Table 2 shows the resulting accuracy with different n-grams and vectorizers on the validation dataset.

**Table 2**

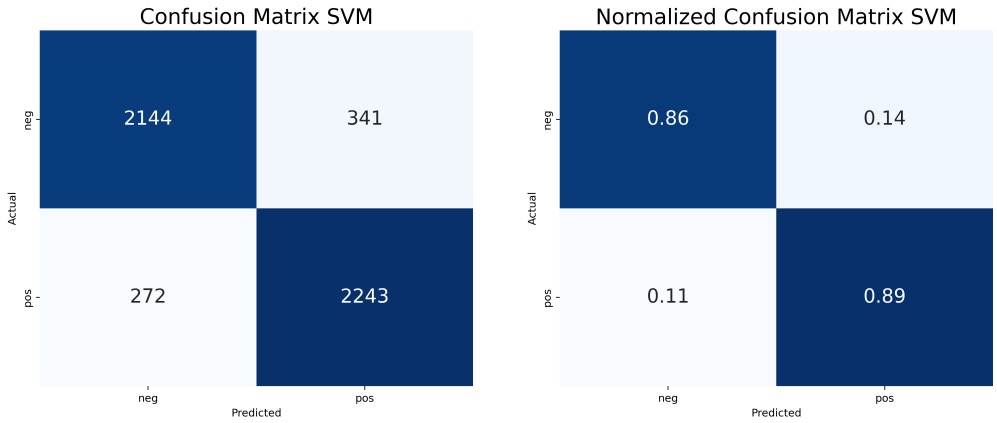
*Comparison of validation metrics for SVM with different n-grams.*

| Vectorizer \ n-grams | unigram | unigram & bigram | bigram |
|---|---|---|---|
| BoW | 0.861 | 0.866 | 0.770 |
| TF-IDF | 0.875 | 0.877 | 0.785 |

The algorithm further allows the usage of different kernel functions. In this projects, mainly the linear and the rbf kernel were investigated. While the linear kernel function showed a better computing efficiency, the accuracy was better with the use of the rbf kernel, therefore the later was kept for further testing.
Another hyperparameter for this approach is regularization, which adds additional terms to the optimization function to improve the ability of generalization. The models provided the best performance with a regularization parameter of 0.9. Since the SVM model showed a slightly better performance with TF-IDF vectorized data with uni- and bigrams, these values were used for further considerations. The confusion matrix of the validation data with the SVM model with uni- and bigrams vectorized with TF-IDF, the rbf kernel function and a regulation parameter of 0.9 is shown in Figure 3.

**Figure 3**

*Confusion Matrices of SVM with raw counts (left) and normalized (right).*



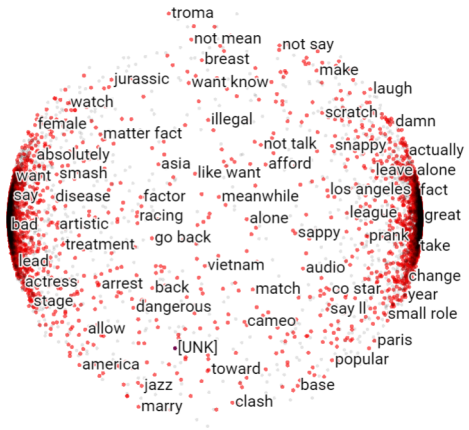

### 5.3 Deep Learning

There are several open source deep learning methods available for sentiment analysis. Due to time restrictions and more complex implementation than the aforementioned machine learning methods, there had to be made a decision which one to use.
In this project, word embeddings from the tensorflow library were used to generate a deep learning neural network for sentiment analysis. For this, different architectures were build and compared along with the validation set.
Word embeddings provide an efficient dense representation of text data where similar words have a similar encoding. An embedding can be considered as a dense vector of floating point values. One important hyperparameter tuned for this approach is the dimension of the embeddings, where a higher dimension can capture more fine-grained relationships between words, but takes more data to learn (Martín Abadi et al., 2015).
Since the library works with a slightly different approach than the scikit-learn libraries mentioned above, the datasets were preprocessed with the approach described in Chapter 3 beforehand and stored into a new directory. Therefore, the TextVextorization of tensorflow was implemented without further preprocessing on runtime.

Again, different hyperparameters were tested on this approach: The n-gram range was applied with unigrams only, unigrams and bigrams and bigrams only, where the option with unigrams and bigrams showed the best accuracy. The vocabulary size was tested with 2,000, 10,000, 15,000 and 20,000, with the highest evaluation accuracy on a size of 15,000. The sequence length was initialized with 100, 500 and 1,000 where a value of 500 resulted in the highest accuracy on the validation data in the training process. The dimension of the embedding layer was tested along with values of 5, 8, 10 and 20, where a dimension of 8 improved the evaluation metrics.
For tuning the hyperparameters and visualizing the result during the development phase, the weights and metadata, i.e. the vector representation of the vocabulary were stored and plotted with the aid of the Embedding Projector of tensorflow, one example is shown in Figure 4.

**Figure 4**

*Example of Word Embeddings visualized with Embedding Projector.*




The following architecture of the neural network shows the one with the best performance among all implemented and was used as the final model:

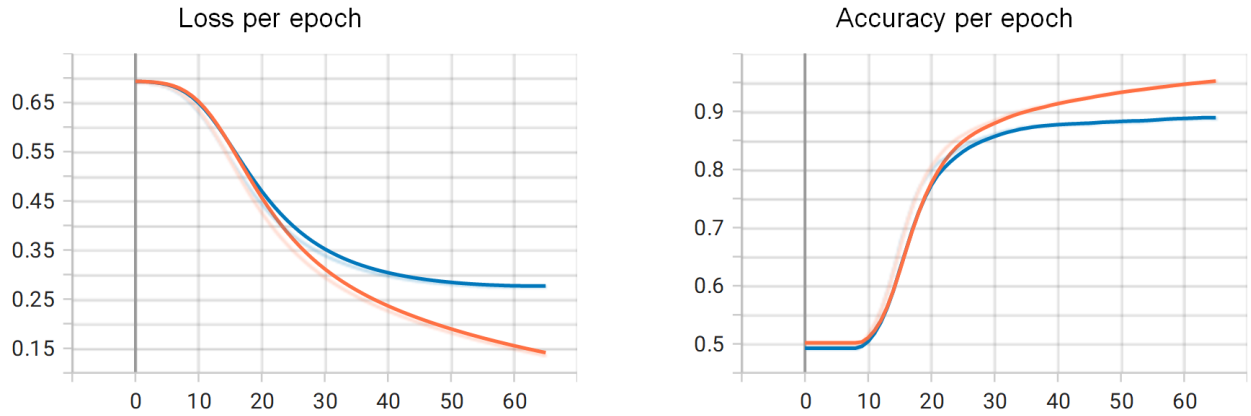
- Text Vectorization with sequence length of 500 and vocabulary size of 15,000
- Embedding Layer with 8 dimensions, vocabulary size of 15000
- Global Average Pooling (1D) for generating fixed size of word sequences
- Dense Layer with 16 nodes and ReLU activation function
- Dense Layer with 1 node

The model was compiled with BinaryCrossentropy as loss function and Adam as optimizer. The training process was designed for 100 epochs, with the option of early stopping after the first 15 epochs, since first training procedures showed that the validation accuracy decreased in many cases after around 30 epochs, but also needed around 10 epochs tuning in with an increasing sequence length.
Since there were only two classes predictable, a simple Output Layer was used without an activation function specified. With this setup, negative prediction results are classified as negative sentiment, while positive results are classified as positive sentiments.
The resulting metrics with loss and accuracy per epoch of the final model are shown in Figure 5.

The training process of the neural network terminated with a training accuracy of 0.954 and loss 0.137 and validation accuracy of 0.889 and loss of 0.277, which can be considered as sufficient comparing with the machine learning results described above. In order not to generate overfitting on the training samples, the early stopping at epoch 66 is considered appropriate.

**Figure 5**

*Training metrics per epoch of the Neural Network with Loss (left) and Accuracy (right).*




*Note.* Metrics of Training Set in orange, Metrics of Validation Set in blue.

## 6 Results

The supervised machine learning models discussed showed a significant better performance on the validation dataset after using the preprocessing steps mentioned in Chapter 3 and resulted in an increased accuracy of up to 15 percent.
For the neural network with embedded layer, on the other hand, the accuracy only increased by around five percent, but differently than published by Camacho-Collados and Pilehvar (2017), the model performed better when trained on preprocessed data. This might be due to the fact that in this project stopword removal and part-of-speech tagging was conducted. However, this assumption requires further research, which is not addressed in this report.

After adapting the algorithms to a suitable accuracy on the validation dataset, the generalization performance was tested on the test dataset. As already mentioned in Chapter 2, the test dataset can be considered as disjoint of the training data. Therefore this approach can be considered as a suitable method to evaluate the ability of the models to predict new, unknown data.

**Table 3**

*Comparison of metrics on the test dataset.*

| | Bernoulli | SVM | Deep Learning |
|---|---|---|---|
| Accuracy | 0.848 | 0.880 | 0.887 |
| Precision | 0.842 | 0.872 | 0.886 |
| Recall | 0.860 | 0.891 | 0.889 |
| F1 Score | 0.850 | 0.881 | 0.887 |

Table 3 shows the metrics of the three discussed methods on the test dataset. The values are rounded to the fourth decimal place. While all algorithms show a relatively good performance on unknown data

with a accuracy of more than 84 percent each, the deep learning model showed the best results in summary. Therefore, for further predictions, this model was stored to enable its deployment.

The confusion matrices of the three approaches with raw counts and normalized are shown in Appendix A.1. Overall, all of the developed models can be considered having a good generalization ability and are therefore appropriate solutions for conducting sentiment analysis on new, unseen movie reviews.

### 6.1 Framework for Prediction of new User Data

The output of the system should be a binary classification indicating whether the sentiment towards the movie is positive or negative.
In order to fulfill this task, a simple python file was created which is callable through a command line interface. First the pretrained deep learning model is loaded, then the user is asked to enter a review and then the entered text is preprocessed according to the steps described in Chapter 3, since this is considered to increase the classification ability of the model. The sentiment predicted will be displayed via the command line. While this implementation is rather simplistic, the project report focused on the implementation of a sentiment analysis model itself and therefore is considered sufficient.
The implementation code along with the results can be retrieved from
https://github.com/slayvi/NLP_project

## 7 Limitations and Future Work

The presented models all show good performance on new, unseen data. However, these models could be further optimized, which is not part of this project report due to time constrains. One possible approach is to consider subjectivity and objectivity of the movie reviews. The library textblob (Loria, 2018) differentiates these settings. With considering the subjectivity and objectivity of the reviews a model could weight objective reviews more than subjective ones to increase the overall objectivity of the predictions.
Another approach which could have been used test several more deep learning methods. There is a wide variety of libraries serving LSTM model or even transformers, like BERT (Devlin et al., n.d.). However, since the achieved performances of the models can be considered sufficient, there were no further deep learning methods implemented. Future work could include a comparison between several different deep learning algorithms.
One limitation for enabling a direct comparison between the discussed machine learning methods and the deep learning approach was the usage of different libraries, which resulted in different training and validation test splits. Therefore, these model could result in different performances with the use of the respectively other data. But since the exactly same test set was used on all approaches, this shortcoming can be considered secondary.
Since the models were trained on movie reviews only, they may perform poor on review data which is not movie related. In order to use the developed models on other data content, transfer learning might be necessary.
While the implementation of the user interface was kept rather simplistic for this project, in future work it could be deployed on the world wide web, for example by the use of cloud providers with the aid of Docker (Merkel, 2014).

## 8 Conclusion

This work examined sentiment analysis on movie reviews. After analyzing the dataset, the text was preprocessed with different approaches to enable a high generalization ability on the developed machine and deep learning models. The text data was then vectorized in different ways to enable the further training of these models. The respective results on the validation data were discussed and the metrics of three different approaches were compared on the basis of the test data. The deep learning model, which was considered as the model with the highest performance, was then used to predict new user data. In the end of this paper, limitations of the developed models and potential future work were discussed.

## Bibliography


Abe, S. (2005). *Support vector machines for pattern classification*. Springer London. https://doi.org/10.1007/1-84628-219-5

Aggarwal, C. C. (2022). *Machine learning for text*. Springer International Publishing.

Camacho-Collados, J., & Pilehvar, M. T. (2017). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. https://arxiv.org/pdf/1707.01780.pdf

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (n.d.). Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/pdf/1810.04805.pdf

Honnibal, M., & Montani, I. (2017). Spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing.

Kulkarni, A. (2019). *Natural language processing recipes: Unlocking text data with machine learning and deep learning using python*. Apress.

Loria, S. (2018). Textblob documentation. *Release 0.15*, *2*.

Loukanova, R. (Ed.). (2021). *Natural language processing in artificial intelligence: Nlpinai 2020* (Vol. Volume 939). Springer.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. http://www.aclweb.org/anthology/P11-1015

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, . . . Xiaoqiang Zheng. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux journal*, *2014*(239), 2.

Potts, C. (2010). On the negativity of negation. *Semantics and Linguistic Theory*, *20*, 636–659.

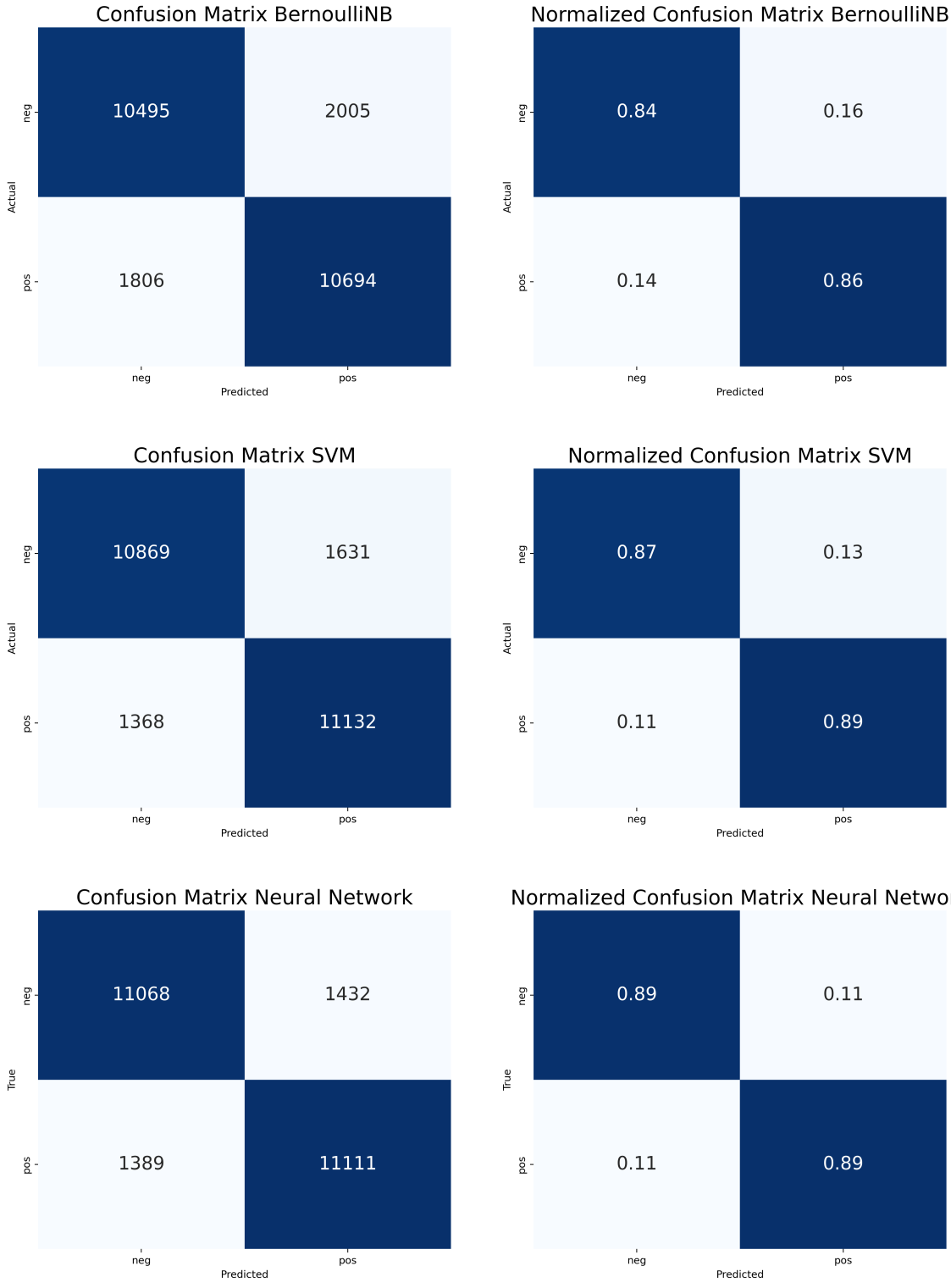
Singh, P. (2022). *Machine learning with pyspark: With natural language processing and recommender systems* (2nd). Apress.

Zong, C., Xia, R., & Zhang, J. (2021). *Text data mining* (1st). Springer.

# A Appendix

## Figure A.1

*Confusion Matrices on the Test Set for Bernoulli Naive Bayes, SVM and Deep Learning (from top to bottom).*




.