NLP Project Report

Gabrijel Radovcic
Eduard von Bothmer
Alexandru Balan Temocico
Tin Josip Curik
Gunesh Ozmen Bakan

Abstract

This study aims to classify emotions in text data, a crucial task in Natural Language Processing with wide-ranging implications across digital communications, mental health analysis, and personalized services. In this paper a variety of models is utilized, ranging from traditional machine learning approaches to advanced deep learning architecture. The research also included preprocessing and augmenting data to deal with dataset imbalances, contributing to the robustness of the models. The key outcomes of the study analyzed the performance of different models in emotion detection, offering valuable insights into their effectiveness. The research emphasizes the crucial role of data preparation and shows the potential of advanced models in dealing with the complexity of emotions in text data.

1 Introduction

Understanding emotions in text is a critical task in the field of Natural Language Processing. It helps in many areas like understanding customer reviews, analyzing social media, and studying mental health. Yet classifying these emotions is not a simple task.

In this report, we look at numerous methods to overcome this problem, ranging from simpler models like Logistic Regression, Support Vector Machines, and Naive Bayes, to more complex models like Recurrent Neural Networks, Convolutional Neural Networks, and Tiny BERT.

Our study used a dataset from Kaggle called the "Emotions Dataset for NLP", which included text entries labeled with one of six emotions: anger, fear, surprise, joy, sadness, and love. We prepared the data by cleaning it, simplifying the text, and encoding the labels. Because the dataset was uneven in terms of the number of entries for each emotion, we also used techniques like data augmentation to balance it out.

The following sections will address the models we used, how they were trained, and how well they performed in classifying emotions.

2 Related work

2.1 Logistic Regression

Logistic regression is a model that has been widely used in Natural Language Processing for emotion detection tasks because of its ability to handle imbalanced and high-dimensional data [12] [5]. It has been used for emotion recognition in short texts because of its ability in successfully modelling class probability distributions [6]. however, it is a model that should only be used as a baseline for emotion detection because it struggles when dealing with complex patterns and long-term dependencies in the data.

2.2 Support Vector Machine

Support Vector Machines are another model that is used for emotion detection in Natural Language processing because of their ability to work in high dimensional spaces [3] since they can handle linear as well as non-linear problems. Nevertheless, SVMs are also struggling when dealing with complex patterns and long-term dependencies, making them another good baseline model to compare with more complex ones for this task.

2.3 Naive Bayes

Naive Bayes classifiers are another family of models that are used for Natural Language Processing tasks because of their simplicity and efficiency [9] [6]. These models are also able to handle high-dimensional data for emotion classification [10] and have also been improved by using techniques like TF-IDF to represent the data [11]. however, as it is the case for Logistic regression, these models also are limited by long-term dependencies in the text data. This means that they are also (like Logistic Regression) a good baseline model for emotion detection.

2.4 Recurrent Neural Networks (RNN)

Recurrent Neural Networks are another powerful model used in the field of Natural Language Processing, having the important ability to model sequential information, an important factor for giving actual context to words. RNNs have the capacity to memorize previous information over time, making the model suitable for tasks where the output is depending on previous data. A common approach is to use Recurrent Neural Networks for sentiment analysis as they can memorize long sequences of words and this is the reason why they perform better for such tasks than simpler methods, using, for example, a bag of words and not taking the context into consideration. LTSM variant of RNNs is usually used for big data sets, helping the model to keep longer sequences in memory, than a normal RNN [15]. On the other hand, GRU is another variant of the RNN, which is simplified when compared with LTSM [2]. This will make the model easier to train and implement, being beneficial in cases where the training data is limited.

2.5 Convolutional Neural Network

Convolutional Neural Networks have been increasingly used in Natural Language Processing for emotion detection since they are able to better capture contextual information from the text data [8]. Moreover, researchers have been trying to improve their performance by adjusting the window of text considered, which resulted in significant improvement in emotion classification tasks [7]. CNNs are, however, also limited by long-term dependencies. Nevertheless, it is possible to increase their performance by creating so-called hybrid models by coupling them with an RNN [14].

2.6 Tiny BERT

Tiny BERT is a compact version of the BERT model [4] and has been developed to deal specifically with sentence classification (since the general BERT model is used to process text of considerable size). It has been widely used in emotion classification tasks in natural language processing because of the learning framework that characterizes it [13]. It is a very compact model, which means that it can be easily deployed, but at the same time it is able to have a performance that can compete even with the larger models (for instance the original BERT model). Specifically it has been proven that a large model doesn't necessarily perform better than it

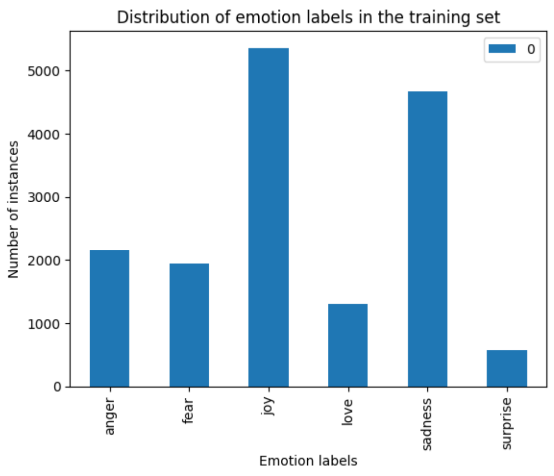

Figure 1: Distribution of emotion labels in the training set

[1].

3 Data

This report summarizes the work conducted on the "Emotions Dataset for NLP" obtained from Kaggle. The dataset includes text entries, each labeled with a distinct emotion from the following: anger, fear, surprise, joy, sadness, and love. The data was preprocessed and prepared for results. This involved removing stop words and lemmatization (reducing words to their root form), which decreased data sparsity. Additionally, basic word preprocessing techniques were used, like converting text to lower case, and removing punctuation and special characters. During the data exploration phase, an imbalance was spotted in the dataset that could potentially impact model performance. To avoid this, data expansion was later conducted to augment the existing dataset, increasing its volume and then the robustness of the models trained on it. This can be better verified by looking at 10.

4 Methods

4.1 Data preparation

The process of preparation began with Label Encoding, which transformed the emotion labels from categorical to numerical values. Next, text data was converted into a numerical format using Count Vectorization and TF-IDF Vectorization (for Logistic Regression, SVM, and Naive Bayes). Pre-trained GloVe word embeddings of size 300 were utilized for the more advanced models (RNN and CNN). This involved building a map of words to indices and a matrix where each row corresponded to a word vector representation. Furthermore, because of the imbalance in the dataset, data augmentation was performed. For this, a new dataset was utilized, called GoEmotions, which was created by Google. It features an extensive labeling system for emotions, which can be aggregated into categories from the original Kaggle dataset. In the end, this made the data being more balanced. It is worth noting that the augmented dataset was only used when training the RNN for the second time.

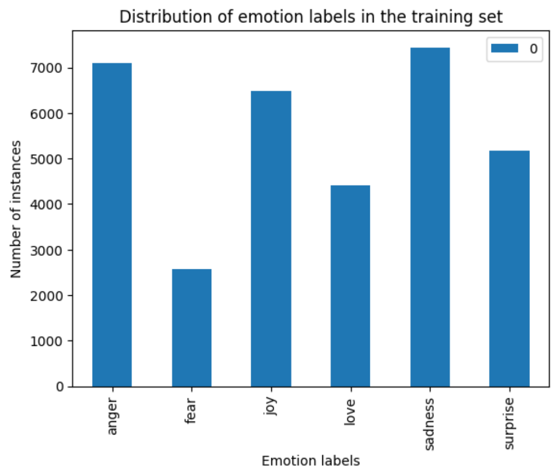

Figure 2: Distribution of emotion labels in the training set after data augmentation

4.2 Logistic Regression

Logistic Regression was one of the models used in this study. Its parameters were optimized using a grid search approach. This systematically tested different hyperparameter combinations to find the best configuration for the model. This process was conducted separately on the Count Vectorization and TF-IDF Vectorization datasets. The grid search yielded the best parameters for the logistic regression model, which afterwards were used for model training.

4.3 Support Vector Machine

Support Vector Machines were another one of the models implemented for the emotion classification task. Two SVM models were trained using the 'linear' kernel, one on the Count Vectorization dataset and the other on the TF-IDF Vectorization dataset. Both SVM models were fitted with the class weight parameter set to 'balanced', tuning the weights inversely proportional to class frequencies in the input data. This is useful for dealing with imbalanced classes.

4.4 Naive Bayes

The Naive Bayes algorithm was also implemented for emotion classification. Specifically, the Multinomial Naive Bayes variant was utilized, which is suitable for classification with discrete features, such as text data expressed as word counts or TF-IDF scores. Again, two Multinomial Naive Bayes models were trained, one each on the Count Vectorization and TF-IDF Vectorization datasets.

The algorithm's smoothing parameter, 'alpha', was set at 0.1 to minimize the impact of unseen features in the test data on the model's predictive performance.

4.5 Recurrent Neural Network

This study utilized a Recurrent Neural Network with a Gated Recurrent Unit for emotion detection in text data. The model architecture consisted of an Embedding layer, a Bidirectional GRU layer, and a Dense layer. The Embedding layer transformed the vectorized words into dense vectors. The Bidirectional GRU layer processed sequence data in both directions, capturing temporal dependencies. The Dense layer, with a softmax activation function, outputted the probability distribution over the emotion categories. This architecture takes advantage of the strengths of RNNs in handling sequential data, such as text. This model has been used on both the original dataset as well as the augmented one.

4.6 Convolutional Neural Network

The CNN model also followed a subsequent structure, with the following layers:

- The embedding layer utilized in the CNN architecture was analogous.

- A 1D convolutional layer with a defined number of filters and kernel size, using a ReLU activation function.

- A GlobalMaxPooling1D layer was introduced to reduce the dimensionality of the output from the Conv1D layer.

- A dense layer with a set number of units, using a ReLU activation function.

- A dropout layer with a certain dropout rate to prevent overfitting.

- A final dense layer with a set number of units and a softmax activation function, identical to the one in the GRU model.

The model was compiled and trained similarly to the GRU model but incorporated an early stopping callback to prevent overfitting.

4.7 Tiny BERT

For the Tiny BERT model the training began by tokenizing training, validation and test sets with the Tiny BERT tokenizer. Subsequently, after converting the emotion classes into integer values via a LabelEncoder, the datasets for model training and evaluation were created by using the encoded sentences with their labels. Finally the model was instantiated: a Tiny BERT model for sequence classification, optmized using the Adam optimizer and Tensorflow's SparseCategoricalCrossentropy as loss function. The training was performed for seven epochs and the model's accuracy is then evaluated on validation and test datasets to assess its performance.

5 Results

Performance was evaluated based on the F1 macro score, which calculates the F1 score for each class independently and then computes their average. This scoring method treats all classes equally, not respectively of their imbalance. The models' performance was assessed on the training, validation, and test sets, and a detailed classification report was generated for the test data.

5.1 Logistic Regression

	precision	recall	f1-score
accuracy			0.89
macro avg	0.83	0.86	0.84
weighted avg	0.89	0.89	0.89
Test F1 macro			0.8422747973604164

Table 1: Results of logistic regression on data vectorized with Count Matrix

	precision	recall	f1-score
accuracy			0.86
macro avg	0.82	0.81	0.81
weighted avg	0.87	0.86	0.86
Test F1 macro			0.8470907906226007

Table 2: Results of logistic regression on data vectorized with TFIDF Vectorization

5.2 Support vector machines

	precision	recall	f1-score
accuracy			0.88
macro avg	0.83	0.86	0.84
weighted avg	0.89	0.88	0.88
Test F1 macro			0.8418535849804712

Table 3: Results of SVM on data vectorized with Count Matrix

	precision	recall	f1-score
accuracy			0.85
macro avg	0.80	0.85	0.81
weighted avg	0.87	0.85	0.86
Test F1 macro			0.8409496743858802

Table 4: Results of SVM on data vectorized with TFIDF Vectorization

5.3 Naive Bayes

	precision	recall	f1-score
accuracy			0.80
macro avg	0.76	0.71	0.73
weighted avg	0.79	0.80	0.79
Test F1 macro			0.7270716402449441

Table 5: Results of Naive Bayes on data vectorized with Count Matrix

	precision	recall	f1-score
accuracy			0.76
macro avg	0.82	0.58	0.62
weighted avg	0.78	0.76	0.74
Test F1 macro			0.6226059012173246

Table 6: Results of Naive Bayes on data vectorized with TFIDF Vectorization

5.4 GRU

5.4.1 Original dataset

	precision	recall	f1-score
0	0.93	0.93	0.93
1	0.88	0.92	0.90
2	0.96	0.95	0.95
3	0.83	0.84	0.84
4	0.97	0.96	0.97
5	0.74	0.77	0.76
accuracy			0.93
macro avg	0.89	0.90	0.89
weighted avg	0.93	0.93	0.93

Table 7: Results of the GRU on original dataset

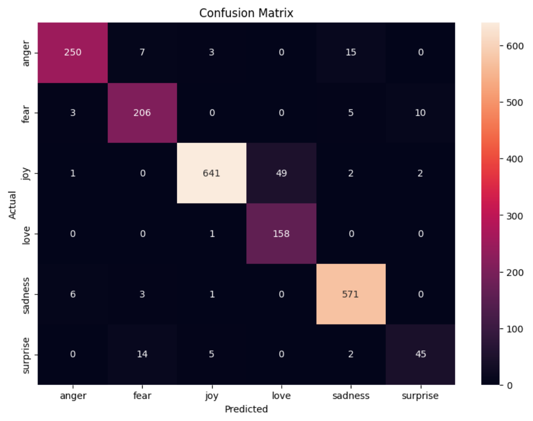

Figure 3: Confusion matrix for the results for the GRU on original dataset

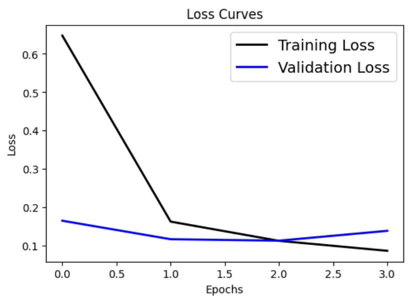

Figure 4: Loss curves for the GRU on original dataset

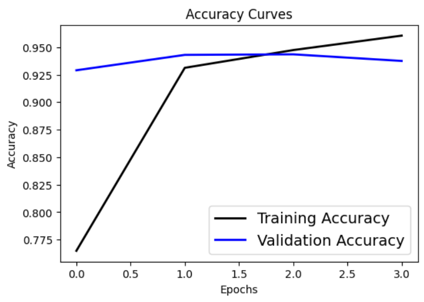

Figure 5: Accuracy curves for the GRU on original dataset

5.4.2 Augmented dataset

	precision	**recall**	**f1-score**
0	0.94	0.91	0.92
1	0.92	0.92	0.92
2	0.96	0.95	0.95
3	0.82	0.85	0.83
4	0.96	0.97	0.97
5	0.75	0.86	0.80
accuracy			0.93
macro avg	0.89	0.91	0.90
weighted avg	0.94	0.93	0.93

Table 8: Results of the GRU on augmented dataset

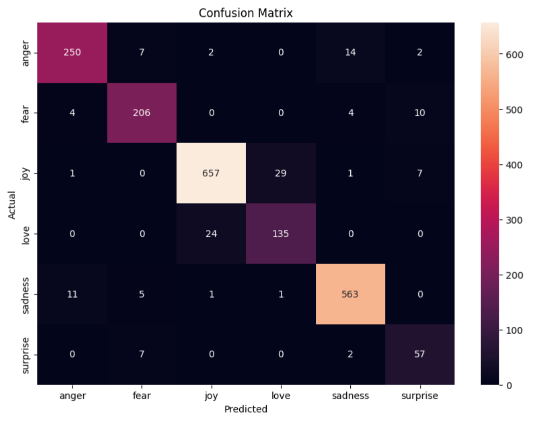

Figure 6: Confusion matrix for the results for the GRU on augmented dataset

5.5 Convolutional Neural Network

	precision	**recall**	**f1-score**
0	0.96	0.90	0.93
1	0.90	0.90	0.90
2	0.93	0.96	0.94
3	0.83	0.81	0.82
4	0.94	0.97	0.96
5	0.88	0.68	0.77
accuracy			0.93
macro avg	0.91	0.87	0.89
weighted avg	0.93	0.93	0.92

Table 9: Results of the CNN model

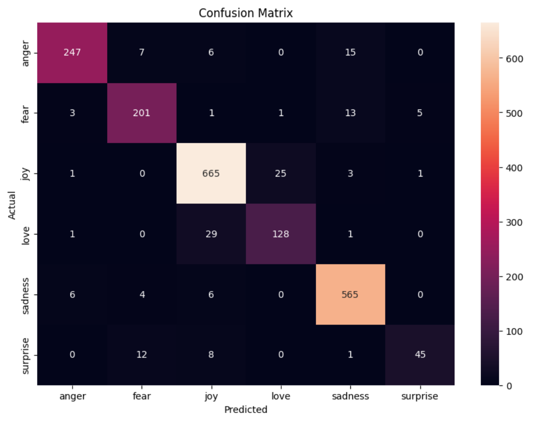

Figure 7: Confusion matrix for the results of the CNN model

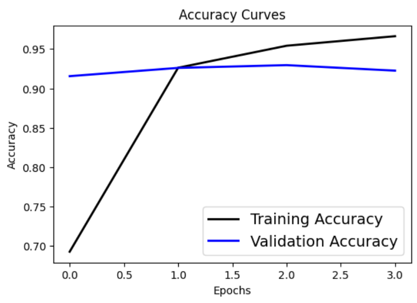

Figure 8: Accuracy curves for the CNN model

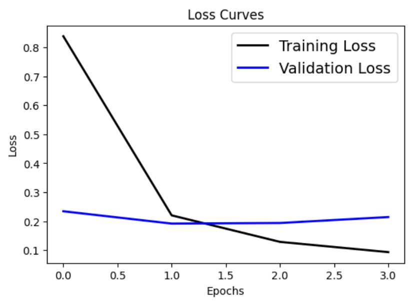

Figure 9: Loss curves for the CNN model

5.6 Tiny Bert

	precision	recall	f1-score
0	0.91	0.87	0.89
1	0.87	0.88	0.88
2	0.93	0.92	0.92
3	0.75	0.82	0.78
4	0.93	0.94	0.94
5	0.71	0.76	0.74
accuracy			0.90
macro avg	0.85	0.86	0.86
weighted avg	0.90	0.90	0.90

Table 10: Results of the Tiny BERT model

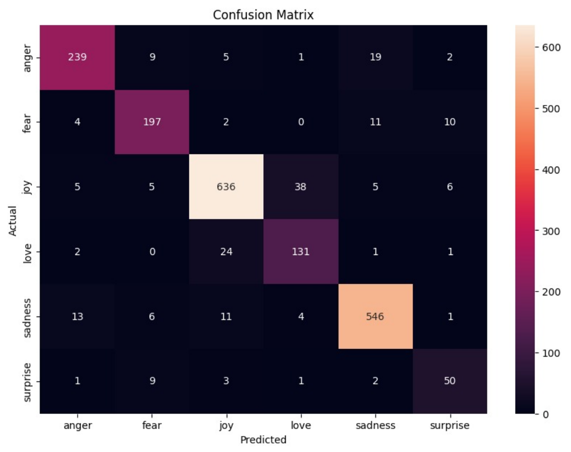

Figure 10: Confusion matrix for the results of the Tiny BERT model

6 Discussion

As can be seen in the Results section, all three baseline models showed significant performance discrepancies between training, validation and test sets (specifically, they all overfit the data), which means that the models are not able to well generalize the data. This issue motivated the use of more complex models such as neural networks, which give the possibility of changing additional parameters such as dropout and regularization, to better fit the data.

Regarding the vectorization technique, TF-IDF performed better for Logistic Regression and Support Vector Machine, even though the difference in performance is not significant, while Naive Bayes had an increased performance with the Count Matrix.

The research also highlighted the issue caused by the imbalanced dataset since all models had a better performance on classes 2 (joy) and 4 (sadness) with respect to the F1 macro score. As can be seen in the previous section where the structure of the dataset is discussed, the class labeled as "surprise" contained fewer examples than the others; this was then reflected in the overall performance of the models.

Overall, when comparing the performance of the baseline models, both Support Vector Machine and Logistic Regression had an F1 score of 84-85%, significantly outperforming the Naive Bayes model, which only reached 62%

For the more complex models, the one with best performance was the Bi-directional GRU which is still very close to the CNN. Surprisingly enough, the Tiny BERT model performed worse than expected. This could be caused by a number of reasons, such as the dimension of the data: BERT is normally used to work on text data with considerable size and Tiny BERT is a more compact version that was developed to handle text of smaller size. Nevertheless, the size of the sentences in the dataset that was used for this research might be too short, enabling other models, such as the Bi-directional GRU, to better represent the sentences without losing information, thus outperforming a model such as Tiny BERT.

The best performing Bi-directional GRU model has also been retrained on the expanded dataset, which increased the total amount of data by 50%. Surprisingly, there was no significant difference in performance compared to the model trained on the original dataset.

These findings motivated an analysis of the test set instances that posed a challenge for the classifiers: the following are examples of instances where incorrect predictions were made, keeping in mind that these sentences have been tokenized.

- In the first example, the sentence was ['feel', 'bit', 'stressed', 'even', 'though', 'thing', 'going', 'fun']. The model predicted the label ['sadness'], while the actual label was ['anger'].
- In the second case, the sentence was ['feel', 'like', 'paradise', 'kissing', 'sweet', 'lip', 'make', 'feel', 'like', 'dive', 'magical', 'world', 'love']. The model predicted the label ['love'], while the actual label was ['joy'].
- In the third example, the sentence was ['friend', 'dropped', 'frog', 'neck']. The model predicted the label ['fear'], while the actual label was ['anger'].
- The last example had the sentence ['feel', 'heart', 'tortured', 'done']. The model predicted the label ['fear'], while the actual label was ['anger'].

It is clear that because of bad labeling there is a crucial loss of information which prevents the models from reaching a higher performance since also a human reader would struggle in classifying sentences like the ones listed above.

Furthermore, as can be seen from the confusion matrices in the previous section, a considerable amount of missclassification happened when the models tried to distinguish between sentences labeled as "joy" and the ones labeled as "love". What in fact happens in other studies, is that normally the two labels are aggregated in one single class for this specific reason: it is difficult even to a human reader to clearly distinguish the line between the two, thus it cannot be expected from a classification model to be able to discriminate between the two, simply because often they coincide. Thus what is ultimately believed, by looking at the result of this study, is that the dataset itself contains a conspicuous amount of misslabeled data: specifically, some sentences labeled as "joy" might actually be part of the class "love" and vice versa, thus the models are not able to generalize the difference between the two classes. Specifically for the implemented models, between the two classes CNN missclassified around 50/141 (35.5%), the GRU 50/129 (38.7%) and Tiny BERT 62/201 (30.8%). Thus it can be seen that the latter model is able to better distinguish between these two classes, but then it struggles when classifying the other ones, which led to it being outperformed by the two former models. Furthermore, analyzing the results of the GRU on the augmented dataset, similar results were achieved, thus proving that this issue cannot be addressed by solely increasing the dimensions of the data. For this reason it may be useless to try to reach a higher accuracy since these models might already be reaching their highest level of accuracy given these datasets and preprocessing techniques.

Furthermore, the examples previously listed were not anomalies, they were instead easily found among the first instances that were incorrectly classified. This further justifies the limited accuracy that the models were able to reach (even though the results are still very promising).

References

[1] Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. *Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics*. 2021. arXiv: 2110.01518 [cs.CL].

[2] Kyunghyun Cho et al. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. 2014. arXiv: 1409.1259 [cs.CL].

[3] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20 (1995), pp. 273–297.

[4] Xiaoqi Jiao et al. "TinyBERT: Distilling BERT for Natural Language Understanding". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 4163–4174. DOI: 10.18653/v1/2020.findings-emnlp.372. URL: https://aclanthology.org/2020.findings-emnlp.372.

[5] Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[6] Dan Jurafsky and James H Martin. *Speech and language processing. Vol. 3*. 2014.

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. *A Convolutional Neural Network for Modelling Sentences*. 2014. arXiv: 1404.2188 [cs.CL].

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[9] David D Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval". In: *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings 10*. Springer. 1998, pp. 4–15.

[10] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques". In: *arXiv preprint cs/0205070* (2002).

[11] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 1. Citeseer. 2003, pp. 29–48.

[12] Sebastian Raschka. *Python machine learning*. Packt publishing ltd, 2015.

[13] Iulia Turc et al. "Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation". In: *CoRR* abs/1908.08962 (2019). arXiv: 1908.08962. URL: http://arxiv.org/abs/1908.08962.

[14] Wenpeng Yin et al. *Comparative Study of CNN and RNN for Natural Language Processing*. 2017. arXiv: 1702.01923 [cs.CL].

[15] Yong Yu et al. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computation* 31.7 (July 2019), pp. 1235–1270. ISSN: 0899-7667. DOI: 10.1162/neco_a_01199. eprint: https://direct.mit.edu/neco/article-pdf/31/7/1235/1053200/neco_a_01199.pdf. URL: https://doi.org/10.1162/neco%5C_a%5C_01199.