Political Point of view classification

Teodor Rustad Kvisberg & Even Balto
Economics - Business Analytics / 4. Year
teodorrkl@stud.ntnu.no evenbal@stud.ntnu.no

Abstract

This study explores how text data from reddit posts can be used to predict if the user has a conservative or radical political point of view. By applying different methods such as Bayes classifier and Recurrent neural networks we aim to construct a model that can accurately classify users' political view. The findings from this study will contribute to how language from social media platforms can reveal someone's political preferences, and consequently work as tool for political analysis and targeted communication.

1 Introduction

This report is a part of the subject *TDT4310 Intelligent Text Analysis and Language Understanding* at the Norwegian University of Science and Technology. When deciding on which topic to apply our newly adopted knowledge we quickly knew that we wanted to include social media in some way. Social media has become a huge part of most people lives, and a place where people like to voice their opinion. With big sports events, political elections and other happenings, people create debates online on a variety of different platforms on a daily basis. The debates leave a lot of data online and a big opportunity for analysis.

For political parties, understanding how people react to their political measures are crucial. Especially, understanding how their target group react is essential. By getting insight in such data can political parties better adjust and form their strategies. Developing a classification model that can classify political preferences can be used to find the people in the target group and works therefore well as a base model. To find out how different measures are welcomed in different targets groups one can apply a sentiment analysis after extracting the relevant people. However, this report will only focus on developing a good base classification model that can classify people into interesting target groups.

Developing text classification models is a well explored research area. Jang et al. (2020) found that including an attention mechanism into a recurrent neural network (RNN) architecture helped improve the performance. While this was tested on relative longer reviews with an average length of 238 words, found other results for his short text data setDeng et al. (2021) . He found that Bi-LSTM and CNN models performed better than the attention based Bi-LSTM-CNN model. This report will look at how similar model structures performs on relative short text social media posts.

The report will be structured as follows: First, we will explain relevant context and terminology in the background section. Then we move on to the related work section, where we will discuss common classification approaches to social media texts. Next, will the architecture and methodology be presented in section four before showing our results in the following section. Finally, will we discuss our results and make a conclusion in the last sections.

2 Background

In this section will we go through the theoretical basis of our work. The section will work as a baseline for understanding the models presented later in the report.

2.1 Naïve Bayes

Naïve Bayes is probabilistic machine learning method based on Bayes' theorem that is based upon the assumption of independency between different features. The method is known for its simplicity and is widely used in, among other things, text classification.

The concept of Naïve Bayes revolves around determining the probability of a document belonging to a specific class based on its feature values. When applying this concept on text classification, features are typically the words present in a document. By utilizing Bayes' theorem, we can determine the posterior probability of a class given a document. Anote (2023)

Because of the simplicity of a Naïve Bayes approach, it is often used a baseline model when evaluating more advanced models. Consequently, have we created a Naïve Bayes model to work as a baseline model. This will be presented with our models in a later section.

2.1.1 Term frequency-inverse doc frequency (TF-IDF)

TF-IDF is a method that is used for evaluating the importance of a word in a document compared to a collection of documents. This results in a score/weight based on how often the word occurs in a document (term frequency) and how unique it is across the document collection (inverse document frequency). TF-IDF weights is power tool when performing classification tasks based on its ability to identify important words for different classes.

2.1.2 N-grams

N-grams consist of possible sequences of n amount of words in a piece of text. They are quite useful when aiming to capture contextual information and relationships between words. For instance, when looking at bigrams (2-grams) for the sentence "Language modelling are fun" we are looking at "Language modelling", "modelling are", "are fun" as bigrams.

N-grams can particular be useful for text classification because of its ability to encode information about word order and context. In our report we will utilize the strengths of n-grams in our baseline Naïve Bayes model.

2.2 Recurrent Neural Network

A recurrent neural network (RNN) is network that uses sequential data. It is commonly used in typical NLP tasks such as language translation, speech recognition, image capturing, sentiment analysis and classification.

The RNNs distinguishes themselves by their memory. As inputs, a RNN model uses both the current input data, x, and a hidden state, h, based on their memory. This continues in a sequential way. In comparison, traditional deep neural networks see inputs and outputs as independent of one another. IBM

Additionally, the RNN differ from feedforward networks by using backpropagation through time (BPTT). Although the principles of BPTT are the same as traditional backpropagation, by training itself from calculating errors, it differs in how it calculates errors. While BPTT sums errors at each time step, the traditional backpropagation do not sum errors because the parameters are not shared across layers.

Through the process of BPTT, the RNNs can experience to types of gradient problems, exploding gradients and vanishing gradients. Exploding gradients occurs when the model weights grow so large that they end up being represented as NaN. On the opposite side, vanishing gradient problem occurs when the gradient becomes so small that it stops updating the parameters. This means that the algorithm stops learning.

To avoid these gradient problems different solutions can be implemented into the RNN architectureIBM. Examples of this are regularization techniques, ReLU activation functions or LSTM and GRU cells. In our models we have decided to include a LSTM layer.

2.2.1 Long Short-term Memory (LSTM)

The LSTM is a RNN architecture designed in 1997 that uses two different paths when making predictions. The first path is for long-term memories and the other for short term memories.

The LSTM consist of a memory unit, that itself consist of four different feedforward neural networks. These neural networks perform different memory management operations and decides which information that will be kept in the memory.

The first stage in the LSTM unit determines what percentage of the long-term memory that is remembered and is called the forget gate. The second and third stage combines the short-term memory and the input to determine how we should update the long-term memory. This stage is called the input gate. The final stage determines how we should update the short-term memory and is called the output gate. How the different stages work is illustrated in figure below. Ottavio Calzone (2022)

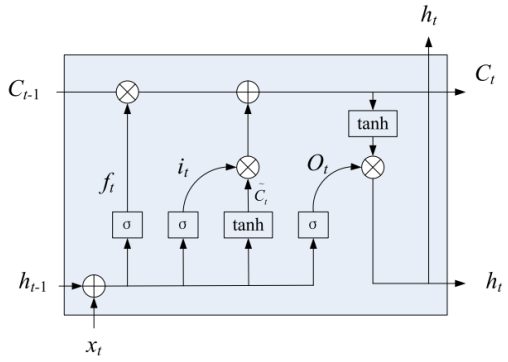

Figure 1: Shows the LSTM architecture. Deng et al. (2021)

Bi-directional Long Short-Term Memory (Bi-LSTM)

Bi-LSTM is a another RNN-architecture that builds upon the regular LSTM architecture. However, unlike the traditional LSTM network that processes input sequences in single direction, Bi-LSTMs processes sequences in a forward and backward direction simultaneously. In practice, this means that for every time step, the network considers both past and future contexts before predicting.

By leveraging information from both past and future contexts, Bi-LSTMs can achieve higher accuracy compared to traditional LSTMs and is the reason why we have chosen to use this in our models.

2.2.2 Convolutional Neural Networks (CNN)

Even though CNN is best known for their effectiveness in handling visualizations and images, it has in recent years gained a breakthrough within text classification.

The CNN architecture consists of different layers; convolutional layers, pooling layers and fully connected layers. When applied on text the network ends up treating the text as an image, and consequently the words as pixels.

When an CNN is integrated within a RNN model, it creates an architecture that utilizes the strengths of both models. While the CNN excels at extracting local features, the RNN has its strength in modeling sequential information. The result is a model that can benefit from complementary strengths and improved performance. Vijay Choubey (2020)

2.2.3 Additive Attention in Seq2Seq Models

Additive attention, initially introduced for machine translation by Bahdanau et al. Bahdanau et al. (2016), marked a significant advancement in deep learning for natural language processing (NLP). It addresses the drawbacks of traditional sequence-to-sequence (seq2seq) models that use fixed-length vectors from the input sequence. These fixed-length encodings compromise the model's performance on longer or unfamiliar sequences by challenging the neural network's ability to preserve accuracyNikhil Agrawal (2020).

The use of additive attention in encoder-decoder models, such as RNNs, allows for dynamic information retrieval during decoding Galassi et al. (2021). Additive attention generates a context vector at every output step, providing a focused method for processing input sequences. This context vector, a weighted sum of the input features, is shaped by weights calculated through a trainable compatibility function. This process enhances the model's ability to identify and use the most pertinent information effectively.

The attention layer enhances the network's ability to concentrate on important parts of the input, leading to improved performance in complex NLP tasks.

2.2.4 GloVe

An embedding layer is important for a language model ability to predict. One of the biggest advantages with the embedding layer is that it opens up for transfer learning. Usually, training models demands a lot of computational power. Transfer learning allows you to upload pre-trained weights instead of training them yourself.

GloVe is an open-sourced pre-trained model developed at Stanford and is commonly used for NLP tasksPennington et al. (2014). The model is based upon unsupervised learning to create a vector representation for words. It uses a global approach by analyzing the whole corpus at the same time. This approach allows GloVe to catch meaningful patterns and relation between words even though they do not appear in the same sentence.

Summarized, utilizing transfer learning to benefit from pre-trained GloVe weights creates a good starting point for our network models.

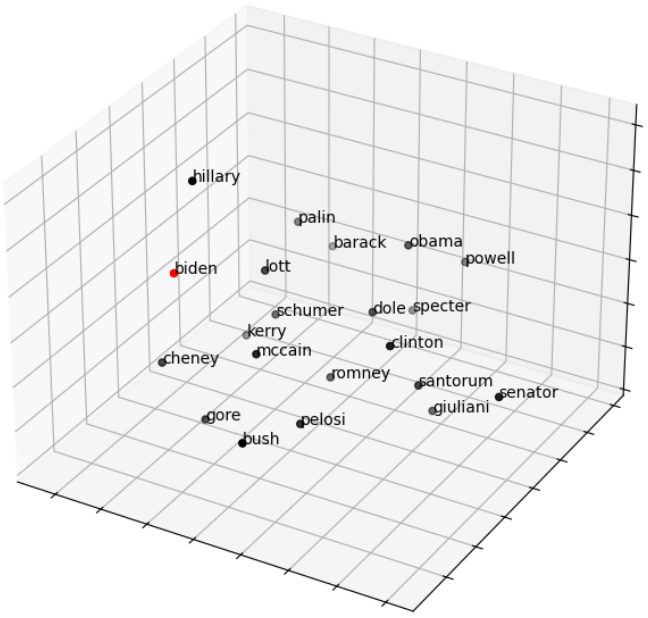

Figure 2: Word representation in vector space with GloVe

2.3 BERT

"In 2017, Vaswani et al. released the paper *Attention is All You Need*, which introduced the transformers architecture that underpins large language models (LLMs). Transformers are composed of two main parts: an encoder and a decoder. BERT (Bidirectional Encoder Representations from Transformers) is an encoder-only model pre-trained on language tasksDevlin et al. (2019). BERT processes words in a sentence simultaneously, leveraging its bidirectional capability to understand each word in the context of all other words in a sentence. This contextual understanding enables BERT to effectively integrate information across the entire input sequence. BERT models can be fine-tuned for specific tasks such as text classification by adding an additional output layer, making them adaptable for various NLP applications."

At the core of BERT is the self-attention mechanism that determines the relevance of various features (words or combinations of words) within the input. This architecture includes several layers of multi-head attention, allowing the model to simultaneously process different types of information from various positions. The self-attention mechanism, supported by multiple heads, enables the model to compute and understand all these contextual relationships in one go, without having to process words in a strict sequential order as previous models like LSTM.

3 Related Work

The task of extracting meaningful information from big data, and especially from sources like social media and websites, is a popular and well explored field. However, the chase for even higher accuracy scores and savings in computational power is still a hot topic in the NLP environment.

Jang et al. (2020) explored how LSTM, CNN, multi-layer perception (MLP) as well as hybrid models performed on a binary classification task compared to their proposed attention based Bi-LSTM-CNN model. Their results show how their proposed model outperforms both the CNN and LSTM models as

well as hybrid models. Further in their report, they conclude that the proposed model can work as an alternative solution to long-term dependency- and data-loss problems.

Later research has explored different approaches. Among them, Zhou (2022) examined how one could use TF-IDF scores to extract the most relevant features from a text, before feeding it into a CNN-LSTM model. The proposed model was then compared to CNN, LSTM and LSTM-attention models, using both short text and long text datasets. The article finds that when extracting the most important features with TF-IDF scores before running an RNN-architecture, the models achieve acceleration and parameter reduction. The consequence was a relatively small loss in accuracy for short texts and almost none for longer texts. This opens up for a interesting soloution when computational power is a problem.

Deng et al. (2021) explored another model for text classification. Building on previous models, they added a gating mechanism, making their provided model a attention-based Bi-LSTM fused CNN with gating mechanism model (ABLG-CNN). Their findings are that Bi-LSTM and CNN-based models works better for short text classification, but that an ABLG-CNN model helps enhance the performance on long-text classification.

After the research process, we have seen that Bi-LSTM and CNN models are heavily used and popular for text classification. The models are usually applied on news articles. Therefore, we thought it would be interesting to explore how effective similar models works on short text social media posts, more precisely Reddit posts.

4 Architecture

This section will firstly introduce the dataset used for this report, before we will discuss the model structure.

4.1 Dataset

The dataset used in this report was downloaded from Kaggle GAJARE (2022). The dataset contains different features about around 13 000 reddit posts, including a labeled political point of view as either Liberal or Conservative. The political view is distributed with 65% liberals and 35% conservatives.

Total	Training	Validation	Test	Min length	25 %	50 %	75 %	Max length	Words (Avg)
12,854	8,998	1928	1928	2	10	14	23	5719	48

Figure 3: Descriptiv about the Dataset

When splitting the dataset into training, validation and test sets we choose to 70%, 15%, 15% split, leaving the model to train on about 9 000 posts. An another interesting point is the relative low word average, that is also heavily influenced by a high outlier of 5719 words. This means that the models are working with relative short texts. The reddit posts has a mean length of 47 words, but 75 per cent of the posts has a length of 23 words. We chose a sequence length of 20 to accommodate the majority of sequences. This length was also selected for computational efficiency.

4.2 Methodology

Inspired from the article Jang et al. (2020) on how they presented that their proposed attention model helped improve the original LSTM-CNN model, will this report use a similar approach when comparing

different models.

Firstly, a Naive Bayes model will be created utilizing TF-IDF scores and both unigrams and bigrams. This model will work as a baseline when comparing the different RNN models.

Then a simple Bi-LSTM model will be introduced as a starting point for our RNN models. Moving on, the next model will look at how a Bi-LSTM-CNN architecture can help improve the performance before adding a third model that also includes an attention mechanism for improved performance. Finally, a comparison will be made against a transformer model to evaluate the performances against each other. For this report we will use BERT for comparison.

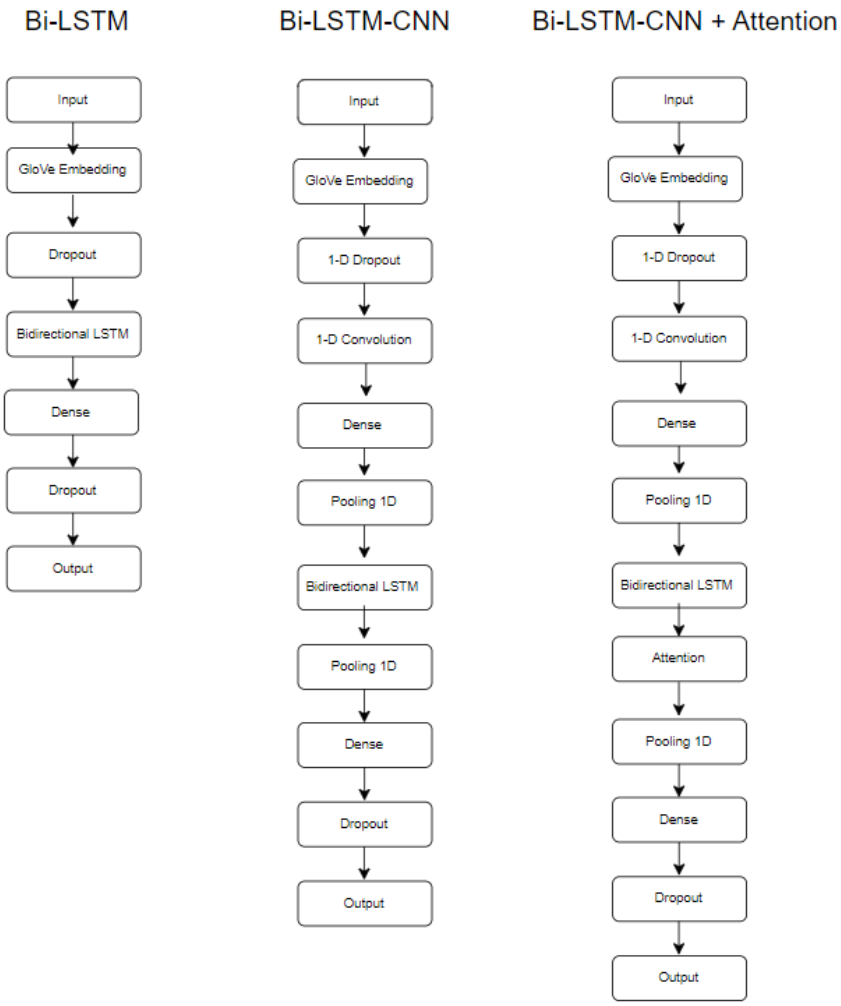

Figure 4: Architecture of our different RNN models - made with draw.io

5 Experiments and Results

In this section we will look at the pre-processing, at the different experimental set-ups, compare them, and look at how they perform against both our baseline model and BERT model, as well as each other.

5.1 Pre-processing

Started by looking for empty rows and creating a column "All text" which contains all the relevant text information from the posts. Abbreviations has also been fixed, U.S and U.S.A are transformerd to the tokens "USA" and "US". Next step in the tokenizing process is lower-casing, removing stopwords and apllying the spacy lemmatizer to create tokens. Preprocessed text are saved as a new column "Processed".

"Political Lean" is our label column which we will one-hot-encode because we want to have a binary case. If the label is equal to 1 then the post is liberal or if the post is equal to 0 it is conservative.

5.2 Experimental Setup

The Naïve Bayes baseline model was trained and evaluated using a pipeline approach. The pipeline firstly consists of a TF-IDF vectorizer that is configured with n-gram range (1, 2), meaning that model considers both unigrams and bigrams in the input text data. Based on the n-grams the pipeline extract the most unique features for each post. The vectorizer gives TF-IDF scores for each features in each sequence, these scores are relative scores based on the whole corpus. The next step is the binarizer which one-hot-encodes all the the features, based on presence or absence. Binarization simplifies the model's input by reducing the variability in feature representations. The final step of the pipeline is the Multinomial Naive Bayes classifier.

When looking at the setup for our experimental setup for our Bi-LSTM models, we refer to figure 4. First we used the tokenizer from tensorflow to encode and create padded sequences. with our chosen length, unknown words as <*OOV*> tokens. We can see that all of these models start with an initial embedding layer where we have utilized transfer learning to load pre-trained weights from GloVe. The setup then introduces a dropout layer at 20% to reduce the chance of overfitting and increasing the generalization performance. This is then followed up by CNN including a convolutional layer, a dense layer and a pooling layer which combined helps the model to capture only the most important local information/features. Note that the CNN are only included in two of the three models. After this, all of the models if followed up with a Bi-LSTM layer with a reccurent dropout on 0.1. Then we introduce an attention layer for the only our last model with a following pooling layer, before all of the models are followed up by a dense, dropout and another dense output layer. For attention we uses the ADDITIVEATTENTION layer from keras.

All the dense layers uses a ReLu activation function. We use a sigmoid function in the last layer, and the model is compiled with binary-crossentropy as the loss function, Adam-optimizer and accuracy set as metric. The optimal learning rate for the optimizer was found through hyperparameter tuning.

The amount of output nodes in every layer as well as amount of epoch and batch size are visualized in the figure below.

Model	Embedding	Dropout	I-D Convolutio	Dense	1D Pooling	Bi-LSTM	Attention	1D Pooling	Dense	Dropout	Dense	Epochs	Batch Size
Bi-LSTM	100	0.2				160			92	0.1445	1	20	32
Bi-LSTM-CNN	100	0.2	64	86	100	160		160	92	0.1445	1	20	32
Bi-LSTM-CNN + attentio	100	0.2	64	86	100	160	160	160	92	0.1445	1	20	32

Figure 5: Experimental setup RNN models

To finetune the hyperparameters in the LSTM-CNN model we used a library in Python called *optuna*. We set the limit for 50 runs and the adjustable hyperparameters where both dense layers, the LSTM layer, the second dropout rate and the learning rate. The objective was to maximize the validation accuracy. Each run had a batch size of 64 and 10 epochs. We saved each trial and adjuster furthermore to see more improvements.

5.3 Experimental Results

The final results for our models are presented in the following figure

Model	Naive Bayes	Bi-LSTM	Bi-LSTM-CNN	Bi-LSTM-CNN + attention	BERT
Loss-Value		0.5504	**0.4998**	0.5209	1.824
Accuracy	**0.7765**	0.7496	0.7662	0,7657	0.7651
F1-score	**0.8469**	0.8089	0.8334	0.8253	0.8232
Recall	**0.943**	0.8085	0.8924	0.8449	0.8346
Presicion	0.7685	0.8091	0.7817	0.8066	**0.8121**

Figure 6: Experimental results

Figure 6 shows that our baseline Naïve Bayes model outperforms our Bi-LSTM models, and even the transformer model BERT. Looking closer at the different Bi-LSTM models, one can observe that the Bi-LSTM-CNN model scores better that both the simple Bi-LSTM model and the more complex Bi-LSTM-CNN + attention model. A further evaluation and discussion on why we have gotten these results will be made in the next section.

6 Evaluation and Discussion

6.1 Evaluation

The models in this report have shown that when working with short text data, complexity might not be the most important element. The fact that the inclusion of an attention mechanism made our performance worse, shows exactly this. Our baseline Naive Bayes model further supports this by outperforming our various Bi-LSTM models as well as the BERT model in terms of accuracy, F1-score and recall. The high F1-score and recall of Naive Bayes indicate that the model is not only capable of identifying the true cases but also correctly classifying a high proportion of them

If we isolate our Bi-LSTM models an interesting observation is how the CNN layer helps improve performance on every metric except precision. In the context of our classification problem where 1 is liberal, a model with this trait might have a tendency to predict more cases of liberals then what is actually the case. This seems fair when our dataset consists of 65 per cent liberal posts. The Bi-LSTM-CNN also shows a significant reduction in test-loss compared to the Bi-LSTM model. This might be because the convolution layer has the ability to capture local features and combinations of words that can be relevant for the classification. The CNN shows a important factor when working with relative short texts, but the high proportion of liberal classification, shows a biased tendency towards the most representative class.

Adding an attention mechanism to the CNN-LSTM did not give any improvements to the models validation accuracy, but the precision went up. The precision is on the other side not higher than the more simpler BI-LSTM model without CNN. High precision indicates that the attention layer helps classifying posts correctly, but in this case the model needs more fine-tuning. This can also be justified by the performance of the BERT which is made up of multi-head attention layers to help identify relevant features for classification. BERT has the best precision of all the models, and this indicates that different attention layers can help precision. The reason for BERT not being that accurate can be because it has a very high loss function, and the model needs more fine-tuning for this specific task in order to work well.

6.2 Discussion

Although the Bi-LSTM models performed poorly on the data set chosen for this report, it can still work well at other/longer types of social media posts. When Jang et al. (2020) did a similar classification experiment on longer text data (IMDB movie reviews), they concluded that their proposed model (attention based Bi-LSTM-CNN) worked best. So, if a similar experiment to this report was made on longer social media texts, like Twitter for example, one could achieve better results for these models.

In context with this, the sequence length used in this report is a relevant factor. As previously talked about, due to limited computational efficiency, we choose a relative low sequence length for our models at 20. This was also tested in the early stages when we worked with a smaller LSTM model. Increasing this to a length that covers around 90% of the data would help our models improve because the LSTM are made for handling longer sequences and remembering key features for predictions. The Naive Bayes classifier had more data to work with since we used shorted sentences for the LSTM-models and the BERT. Therefor the Naive Bayes had more features to use for classification.

Based on a PCA analysis where we visualized the three most explainable dimensions and by testing, we foind that GloVe embeddings provided the highest accuracy for our case. We tested against Googles *Word2Vec* embeddings. GloVe's ability to capture both global statistics and local context within the corpus proved more aligned with our goals of classifying social media posts, which often contains nuanced language and contextual meanings

Another interesting implementation to these models is to add part of speech (POS) tags as feature into our Bi-LSTM models. This could help the models gain more information about a word's role in a sentence. This can be especially useful when looking at social media data with emoji's and special characters. However, with the reddit posts used for this report, emoji's and special characters was not a issue.

Improving the validation accuracy of the BERT even further could be a interesting task, since it shows potential for being precise but because of the high loss the model needs more fine-tuning. The model is made for more complex tasks like text generation, and can be too complex for small datasets. The multi-head attention in BERT are also built for handling longer sequences of various lengths in parallel. If we had kept the lengths of the sequences, we could maybe have benefitted from the models complexity.

Having access to more computational power can lead to higher batch sizes and embedding dimensions and more effective training of the networks. Batch size is the number of training examples used to calculate the gradient during one iteration of model training. Larger batch sizes can lead to more stable and reliable gradient estimates, which can improve the training process by making it smoother and less susceptible to the noise of individual data points. Larger embedding dimensions can also catch up on more nuances in the language which might lead to more accurate results. Its also necessary to have more computational power if we want to tune the BERT model.

7 Conclusion and Future Work

In this report we have explored how different Bi-LSTM models performs on classifying reddit posts with a binary outcom, and compared them against both a baseline Naïve Bayes model as well as a transformer BERT model. The purpose of this report was to examine how different RNN-architectures performs on short-text data, like reddit posts.

The results showed us that a relatively simple Naive Bayes model outperformed different RNN-

architectures that in total included both Bi-LSTM layers, CNN and attention mechanisms. Additionally, the Naïve Bayes outperformed the BERT model as well. A conclusion can be made that to complex models is not optimal when working with so small text data as we have done it this report. Simpler models, as the Naïve Bayes, seems to be preferable as they firstly are easier to train and secondly better and generalizing.

For future work it would therefore be interesting to take a closer look at the Naïve Bayes model and try to test and further develop it. Additionally, if more time were available, the time would have been spent on following the original plan to include both part of speech tagging and dependency parsing in our Bi-LSTM models. We believe that the introduction of both features would help our models capture more complex and syntactic patterns and relations.

References

Anote. Naive bayes: A bag of words approach to text classification", 5 2023. URL https://anote-ai.medium.com/naive-bayes-bag-of-words-approach-a-powerful-text-classification-technique-8ccfb07ac2be.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

Jianfeng Deng, Lianglun Cheng, and Zhuowei Wang. Attention-based bilstm fused cnn with gating mechanism model for chinese long text classification. *Computer Speech & Language*, 68:101182, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

NEEL GAJARE. Liberals vs conservatives on reddit [13000 posts]. https://www.kaggle.com/datasets/neelgajare/liberals-vs-conservatives-on-reddit-13000-posts, 2022. Accessed: 12.03.2024.

Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4291–4308, 2021. doi:10.1109/TNNLS.2020.3019893.

IBM. What are recurrent neural networks? URL https://www.ibm.com/topics/recurrent-neural-networks.

Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang-ug Kang, and Jong Wook Kim. Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism. *Applied Sciences*, 10(17):5841, 2020.

Nikhil Agrawal. Understanding attention mechanism: Natural language processing, 1 2020. URL https://medium.com/analytics-vidhya/https-medium-com-understanding-attention-mechanism-natural-language-processing-9744ab6aed6a.

Ottavio Calzone. An intuitive explanation of lstm, 2 2022. URL https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Vijay Choubey. Text classification using cnn", 7 2020. URL https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9.

Hai Zhou. Research of text classification based on tf-idf and cnn-lstm. In *journal of physics: conference series*, volume 2171, page 012021. IOP Publishing, 2022.