Building sentimental classifiers and robustness testing
Group 2
CS4360 Natural Language Processing

Chandran Nandkumar
5692520
C.NANDKUMAR
@student.tudelft.nl

Laura Holvoet
5841755
L.E.Holvoet
@student.tudelft.nl

Harmen Kroon
4230833
H.M.Kroon
@student.tudelft.nl

Abstract

This paper presents a comparative study of sentiment classification models and adversarial attacks. We explore three different models: a feed-forward neural network, a bidirectional recurrent neural network (RNN) with soft-attention, and a self-attention model. Our primary aim is to investigate the effect of a gradient based adversarial attack algorithm on each of the models. The success rates of these attacks and the distribution of perturbation positions are used to draw conclusions on the robustness of said models. Visualizations of word embeddings selected for perturbation using Principal Component Analysis (PCA) provide insights into the models' decision-making process. This research contributes to understanding sentiment classification models and their vulnerabilities to adversarial attacks, guiding the development of more robust models.

1 Introduction

Sentiment analysis is arguably one of the most important and researched text classification tasks in the field of Natural Language Processing. The primary goal is to predict the sentiment polarity of a given text, typically as positive or negative. This task is crucial in numerous applications, such as social media monitoring, product reviews, and customer feedback analysis. It can also be applied to detect specific feelings or emotions, urgency or intentions. The advent of deep learning has significantly advanced the field, with various neural architectures being proposed for sentiment classification (Zhang et al., 2018).

The task of sentiment classification is far from trivial, since people use various different ways to express their thoughts and emotions and they can sometimes be masked or expressed through sarcasm or irony. Sometimes the input sentence does not contain the explicit sentiment or the apparent sentiment is not what the speaker meant to convey. Sometimes words associated with negative emotions can actually be used to express a positive concept, e.g. 'Break a leg'. Thus, the model can't simply rely on the presence of certain words to classify an utterance, it must be able to extract more complex information from context.

In this paper, we delve into the exploration of three distinct neural architectures for sentiment classification: an Embedding model, a Recurrent Neural Network (RNN) model with soft-attention, and a Self-attention model. All models utilize the same pre-trained GloVe-100 embeddings (Pennington et al., 2014) and are trained and evaluated on the SST-2 dataset (Socher et al., 2013). The aim of this project is to study the performance of three sentimental classifiers with different architectures and to adversarially attack these classifiers to verify and rank them based on their robustness.

The adversarial attacks on these classifiers will be performed using the gradient-based HotFlip algorithm (Ebrahimi et al., 2018). Adversarial attacks have been shown to be an effective way to probe the robustness and vulnerabilities of machine learning models, and our study extends this line of research to sentiment classifiers. We restrict our attacks to perturb only one word in the input and exclude words with strong sentimental preference as adversarial candidates.

We report the classification performance of all models, the success rate of adversarial attacks, the distribution of perturbation positions, and the comparison of attention distribution and perturbation positions for the soft-attention and self-attention models. Furthermore, we visualize the adversarial candidates from the models using Principal Component Analysis (PCA), providing an intuitive un-

Code can be found at `https://github.com/lauraholv/NLP_project` derstanding of the adversarial landscape.

In this paper, the following research questions will be addressed:

- Which one of the three analysed sentiment classifiers is the most sensitive to adversarial attacks?
- Is there a direct correlation between the complexity of a model and its robustness?
- What are the reasons behind the difference in performance of all three models?

Our work contributes to the ongoing discourse in NLP by providing insights into the performance and robustness of different sentiment classification architectures under adversarial attacks. We believe our findings will be valuable for researchers and practitioners aiming to develop more robust and reliable sentiment classifiers.

2 Related Work

Sentiment classification, a key task in Natural Language Processing (NLP), has been extensively studied over the years. Traditional methods relied on hand-crafted features and machine learning algorithms, as demonstrated by (Pang et al., 2002). However, the advent of deep learning has shifted the paradigm towards neural architectures.

2.1 GloVe & Feedforward Neural Networks

The use of word embeddings has been a significant advancement in NLP. These embeddings, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), capture semantic and syntactic relationships between words and have been widely used in various tasks, including sentiment classification. GloVe has been shown to better as a word embedding tool due to its ability to capture both local and global information about the textual input and has thus been used for this study.

To train a model using these embeddings, a common method is to incorporate them into a Feed-Forward Neural Network. (Bengio et al., 2003) proposed the use of a simple feedforward neural network to first learn word embeddings and then the probability function for word sequences. The network architecture consists of an input layer with one-hot encoded word inputs, a linear projection layer for the word embeddings, a hidden layer with a hyperbolic tangent function, and a softmax classifier output layer. However, after further research, it was found that using an NGram CNN model would be a better fit for our research based on the work of (Çano and Morisio, 2020).

2.2 Fast Gradient Sign Method - Adversarial attack

Fast Gradient Sign Method (FGSM) is a popular technique for generating adversarial examples, initially introduced by Goodfellow et al. in the context of deep learning models for image classification (Goodfellow et al., 2015). The premise of FGSM is to leverage the gradients of the model's loss function with respect to the input data to create a perturbation that, when added to the original input, significantly impacts the model's output. This method is noted for its simplicity and efficiency, as it only requires one forward and backward pass through the model to generate adversarial examples (Papernot et al., 2017). Further advancements in adversarial attacks have built upon FGSM, proposing iterative variants that apply the FGSM multiple times with a small step size, which often results in adversarial examples of higher quality (Kurakin et al., 2016). Despite the introduction of more complex methods, FGSM remains widely used due to its simplicity and the insights it provides about model vulnerabilities

2.3 Bi-directional RNN and Soft-attention

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), have been extensively used for sentiment classification due to their ability to capture long-range dependencies in text. The classic RNN is used to collect context from previous timesteps in the hidden layer and by combining the hidden state h_{t-1} and the current input x_t the new hidden state h_t can be computed. (Schuster and Paliwal, 1997) invented a way to independently collect context from both temporal directions in their bi-directional implementation of the RNN and combine often through concatenation. In order to efficiently use the correct information a mechanism that learns to align by soft searching for most relevant positions is introduced by (Bahdanau et al., 2014). The output of a bi-RNN is weighted on each timestep depending on the previous state and the current input. A byproduct of attention is that the weights reflect what linkage the model "sees". The term soft attention is coined by (Xu et al., 2015) where they add an intermediate tanh step before the softmax.

2.4 Self-attention

The attention mechanism is a technique used in Deep Learning applications to encode and convey long term information and dependencies that can not be retained by other hidden vector representations. Introduced in the Transformer model by (Vaswani et al., 2017), this model has also been applied to sentiment classification. This mechanism computes the representation of each word based on all other words in the input, allowing for more flexible and context-aware representations. Attention mechanisms can be parallelized since attention scores are computed independently between tokens and across different positions. This property significantly improves the computational efficiency and allows for reduced training and inference times, especially when working with longer sequences. (Lin et al., 2019) demonstrated the effectiveness of self-attention models in sentiment classification tasks.

The attention mechanism uses 3 weight matrices W_q , W_k and W_v to compute three linear transformations of each input vector for the three different parts of self attention. It maps a query and a set of key-value pairs to an output. The output vector is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Self attention can be given greater power of discrimination by combining several self-attention mechanisms each with different query, key and value weight matrices, called attention heads. For an input vector each attention head will produce a different output vector. This allows to account for the fact that a word can mean different things to different words in the sentence.

2.5 HotFlip: White - box Adversarial Examples

In recent years, adversarial attacks on NLP models have gained attention. The HotFlip algorithm, proposed by (Ebrahimi et al., 2018), is a gradient-based method that flips the characters of the input text to change the model's prediction.This model, contrary to previous work that focused on creating adversarial examples without any explicit knowledge of the model, i.e. in the black box setting, uses complete knowledge of the model to develop worst case attacks that can reveal larger vulnerabilities of the model. The HotFlip algorithm generates modified text samples that aim to mislead the classification model into producing an incorrect output. This model relies on an atomic flip operation that swaps one token for another, based on the gradients of the one-hot input vectors. HotFlip identifies the character in the input text that has the highest absolute gradient value and flips it to another character that maximizes the gradient. This process aims to perturb the input text in a way that increases the loss and moves the predicted class label closer to the desired target class. HotFlip can be applied both at word and at character level. (Li et al., 2019) presented TextBugger, a general attack framework for generating adversarial texts, demonstrating its effectiveness, evasiveness, and efficiency. (Cheng et al., 2020) studied the problem of crafting adversarial examples for sequence-to-sequence models, proposing a projected gradient method combined with group lasso and gradient regularization. (Zhou et al., 2019) proposed a novel framework, learning to discriminate perturbations (DISP), to identify and adjust malicious perturbations, thereby blocking adversarial attacks for text classification.

3 Approach

Our research methodology is divided into two primary sections: the development of sentiment classifiers and the execution of adversarial attacks on these classifiers.

3.1 Sentiment Classifiers

Our first objective is to construct three sentiment classifiers with distinct architectures. We begin by setting up our environment, which includes downloading the Stanford Sentiment Treebank 2 (SST-2) dataset and loading the GloVe embeddings. We create a mapping from words in our corpus to their corresponding GloVe embeddings and set up our PyTorch environment.

Embedding Model The first classifier is an Embedding Model, a simple feed-forward neural network implemented in PyTorch. The input to this model is the sum of the GloVe embeddings of all words in the sentence, effectively creating a sentence-level representation. The model consists of a fully connected layer that maps this input to the output sentiment label. We train this model on our training data and evaluate its performance on the validation/test data.

The first main question which had to be answered was - what architecture of the embedding

Sl no.	Model	Training Time	Accuracy
1	Feedforward	8min 52sec	80.85%
2	Bi-RNN	38min 23sec	82.57%
3	Bi-RNN + Soft-Attention	50min 52sec	84.29%
4	Multi-head attention	1h 20min	77.18%

Table 1: Each model with their training time and baseline test accuracy

model must we use to increase accuracy and performance of our model. (Çano and Morisio, 2020) goes in depth with respect to different architectures for a feedforward neural network using the 'Ngram-CNN' with a filter length of 1,2 and 3(Larger values of lengths did not improve results.). Thus this model was reproduced and a similar architecture was adopted for sentiment analysis. Figure 1 shows the architecture incorporated.

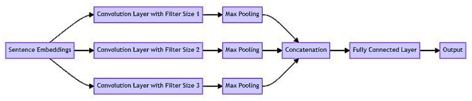
Sentence Embeddings
Convolution Layer with Filter Size 1
Convolution Layer with Filter Size 2
Convolution Layer with Filter Size 3
Max Pooling
Max Pooling
Max Pooling
Concatenation
Fully Connected Layer
Output

Figure 1: Architecture of the NGram CNN model

Bi-RNN Model with Soft-Attention The second classifier is a bidirectional RNN model with a soft-attention mechanism. The input to this model is the GloVe embeddings of the words in the sentence. The RNN processes these word embeddings and produces a sequence of hidden states. An attention mechanism is then applied to these hidden states, generating attention weights that reflect the importance of each word in the context of the sentence. A weighted sum of the hidden states, guided by these attention weights, forms the sentence-level representation. The architecture used is shown in Figure 2

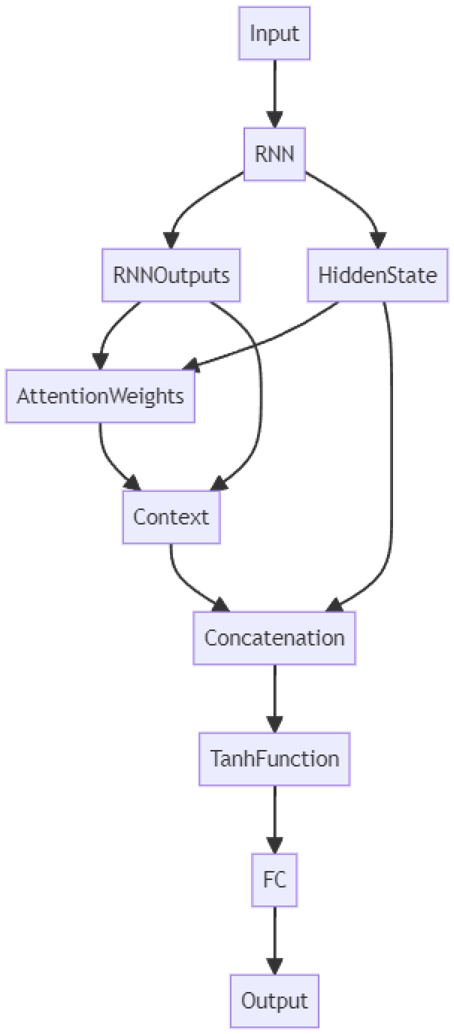

Figure 2: Architecture of the Bidirectional RNN model

Self-Attention Model The third classifier is a Self-Attention Model. This model also takes the GloVe embeddings of the words as input, but instead of using an RNN, it uses a self-attention mechanism to determine the representation of each word based on all other words in the sentence. This model also incorporates positional encoding to capture the order of words in the sentence. After training, we evaluate the performance of this model. The architecture of the same can be seen in Figure 3

3.2 Training

Embedding Model The sentiment classification model was trained on the Stanford Sentiment Tree-

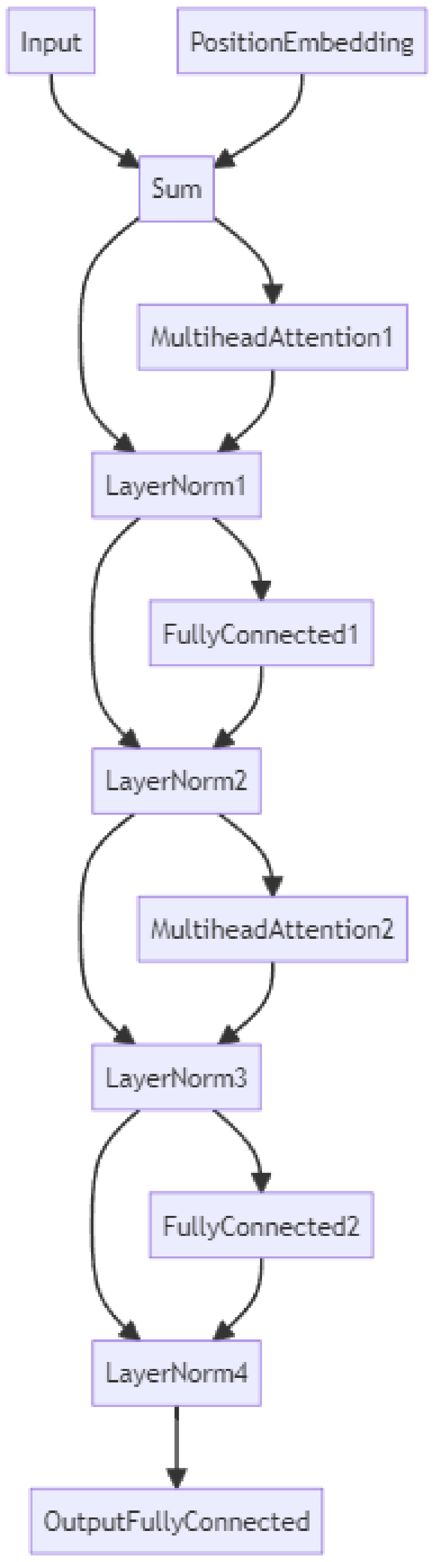

Figure 3: Architecture of the Multihead attention model

bank (SST2) dataset and GloVe embeddings. In this setting a basic NGram CNN feedforward model where the filter lengths were taken as 1, 2 and 3. The model takes the input of the sum of word embeddings of given sentence as input and provides an output based on the predicted sentiment. Tuning the hyperparameters using grid search for the model showed that training for 15 epochs proved best without overfitting on the training data. An Adam optimizer with a learning rate of 0.001 was used in conjunction with the Cross-Entropy loss function. The final model had a training time of 8 minutes 52 seconds in the Google Colab CPU runtime.

RNN Model with Soft-Attention In this setting a basic model with a single soft-attention layer that takes the input of a single layer bi-directional LSTM is used. Tuning the hyperparameters for the model showed that training for 15 epochs proved best without overfitting on the training data. An Adam optimizer with a learning rate of 0.001 was used in conjunction with the Cross-Entropy loss function. The final model had a training time of 50 minutes in the Google Colab CPU runtime.

Self-Attention Classifier In this setting a self attention model was implemented. During the training and hyperparameter tuning process of the classifier, the Adam optimizer with a learning rate equal to 0.0001 proved to have the best results. The Cross-Entropy loss was chosen as the loss function. The model was trained for 45 epochs, which took around 1h20min in total on a CPU runtime in Google Colab.

3.3 Adversarial Attacks

The second part of our methodology involves implementing adversarial attacks on the classifiers developed in Section 3.1.

Fast Gradient Sign Method (FGSM) Fast Gradient Sign Method (FGSM) is an efficient technique used for generating adversarial examples, specifically designed for deep learning models. The FGSM leverages the gradient of the loss function with respect to the input data to create small perturbations which, when added to the original input, can significantly impact the model's output.

For the word embedding model, FGSM had to be used since the sum of the word embeddings were used as input to the model rather than the independent word tokens unlike the other models.

FGSM is particularly relevant due to its simplicity and effectiveness. The nature of FGSM perturbations, small but strategically calculated, aligns with the practical constraint in the problem domain, which allows only minor alterations to the input. Moreover, FGSM serves as an effective tool to test and improve the robustness of the model against adversarial attacks.

The FGSM generates adversarial examples as follows:

$$X_{adv} = X + \epsilon \cdot sign(\nabla_X J(\theta, X, y)) \quad (1)$$

In this equation, X_{adv} represents the adversarial examples, X is the original input, ϵ is a small constant ensuring that the perturbations are small, $sign(\nabla_X J(\theta, X, y))$ is the sign of the gradient of the loss function J with respect to the input data X, and θ represents the model parameters. The term $sign(\nabla_X J(\theta, X, y))$ provides the direction in which to change X to increase the loss, thus leading the model to misclassify X_{adv}.

HotFlip Adversarial Attack We implement the HotFlip algorithm, a gradient-based method for generating adversarial examples. In the context of our task, HotFlip is used to flip one word in the input sentence in order to change the classifier's prediction. The word to be flipped is chosen based on the gradient of the classifier's output with respect to the input embeddings. Thus requiring only one forward and one backward pass. The adversarial token is not evaluated with another forward pass, but instead is determined by an approximation of the change in loss found through the following vector:

$$\nabla_{[\hat{\mathbf{x}}-\mathbf{x}]}\mathcal{L}_{adv} = \nabla_x \mathcal{L}_{adv}^T \cdot [\hat{\mathbf{x}} - \mathbf{x}] \quad (2)$$

The largest vector is then chosen by evaluating the tokens against the token space and taking the largest increase in loss which can be derived with:

$$\underset{1 \le i \le n, \hat{w} \in \mathcal{V}}{\text{argmax}} \; -1 \cdot [\hat{\mathbf{w}} - \mathbf{w_i}]^T \nabla_{\mathbf{w_i}} \mathcal{L}_{adv} \quad (3)$$

For efficiency the gradient of the original word $\nabla_{\mathbf{w_i}}$ is used in a dot product with its own word embedding $\mathbf{w_i}$ and a matrix vector multiplication with the embedding matrix $\mathcal{V}$. Then they are subtracted and negated because the adversarial should have an opposing gradient than the original word; finally, the maximal value is chosen as our adversarial token. In practice the token with the largest gradient is chosen as word to perturb which is then considered against a quarter of the GloVe embedding matrix resulting in a fast approximation of flips.

Alongside the gradient search a set of constraints is taken into account; the exclusion of a portion of the vocabulary up for perturbation and adversarial choice and the limitation of only one word perturbation per attack. The list of unwanted tokens consist of punctuation, stop words and strong sentimental words , which are all left unconsidered in in our algorithm.

Perturbation and Evaluation We generate adversarial examples for each of our classifiers using the implemented HotFlip algorithm. We then measure the attacking success rate for each model, defined as the percentage of adversarial examples that cause the classifier to make an incorrect prediction. We also report the distribution of the perturbation positions across all models.

Attention Distributions and Perturbation Positions For the models with soft-attention and self-attention, we analyze how the perturbed word affects the attention distribution. We visualize the attention weights before and after the perturbation. We also compare the perturbation position with the attention distribution to understand if the models pay more attention to the perturbed word or not.

Visualization of Adversarial Candidates We gather the embeddings of the words selected for perturbation (i.e., the adversarial candidates) for each model. We then use Principal Component Analysis (PCA) to reduce the dimensionality of these embeddings to 2 dimensions. We plot these 2D points, using different colors for the different models. This visualization helps us understand if there are patterns in the words that are chosen for perturbation by the different models.

4 Results

In this section, the results of the adversarial attacks are presented and analyzed.

4.1 Embedding Model

Before any adversarial attacks, the model achieved a remarkable accuracy of 80.85% on the test data. This demonstrates the effectiveness of the chosen architecture in correctly identifying sentiment in

Lists are provided by the Natural Language ToolKit package https://www.nltk.org/

Model	Test Accuracy	Accuracy on adversarial test set	Percentage Accuracy Drop
Feedforward	80.85%	52.64%	34.89%
Bi-RNN + Soft-Attention	84.29%	48.74%	35.55%
Multi-Head Attention	77.18%	31.94%	55%

Table 2: Comparison of model accuracy on the original test set and on the adversarial test set

text, indicating a good understanding of the semantic relationships between words in sentences.

However, to test the model's robustness against adversarial attacks, we used the Fast Gradient Sign Method (FGSM). FGSM is a simple yet effective method to generate adversarial examples, which adds small but intentional perturbations to the input data aiming to cause misclassification. The attack demonstrated the model's vulnerability to adversarial examples, as the accuracy dropped significantly to 52.64% on the adversarially perturbed test data.

Figure 4 shows that the highest frequency of adversarial example positions are in the lower end. This follows logically since reviews obtained from the SST2 dataset are of different lengths. This makes it highly likely that the first few words have higher frequency given a large number of shorter reviews rather than longer reviews where the word responsbile for the highest gradient is on the tail end of the sentence.

To visualize the adversarial examples in a lower dimensional space, we conducted Principal Component Analysis (PCA), reducing the dimensions to two. This analysis revealed a pattern but no discernable information could be extracted from it as seen in 5. The first two principal components accounted for approximately 16.81% of the total variance in the adversarial examples, with the first and second components explaining 12.97% and 3.84% of the variance, respectively. This indicates that the adversarial perturbations introduced by FGSM have a widespread and diverse impact across different dimensions of the word embeddings, affecting a broad spectrum of semantic features in the data.

While the model shows strong performance in sentiment classification tasks under normal conditions, its performance significantly degrades under adversarial attacks. This highlights the need for further research on the robustness of NLP models against adversarial perturbations, and the importance of developing strategies for defending against such attacks.

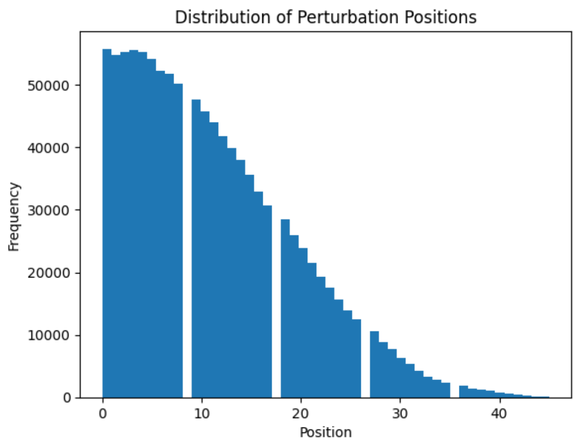

Figure 4: Distribution of perturbation positions

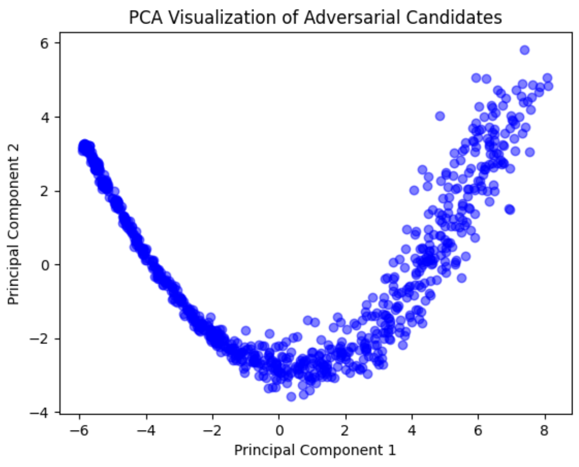

Figure 5: PCA visualisation of adversarial candidates reduced to two dimensions. Two dimensions were selected as the second dimension already rendered only 3.8% of the information contained in the embeddings

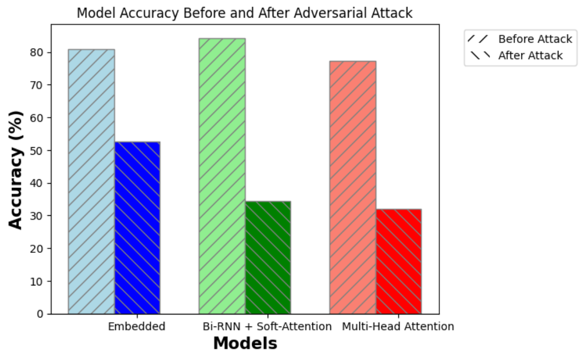

Figure 6: Accuracy of the three models before and after adversarial attack

4.2 Bi-RNN Model with Soft-Attention

The Bi-RNN extends on the Feed Forward Embedding model with the three elements that aid in context preservation and text classification: an LSTM considering hidden states based on history, the bi-directionality considering history from two temporalities and the soft-attention layer providing focus on the significant input tokens. All these additions did prove an almost 3.5 percentage point accuracy increase to 84.29%, at the cost of a sixfold increase in training time.

The creation of adversarial data on this model was ultimately a matter of minutes using the first order Taylor approximation. Letting the model run on this newly created adversarial validation set gave an accuracy of only 34,52%. This is a showcase of the effectivity of the HotFlip algorithm in creating adversarial sentences and at the same time a look under the hood of the soft-attention mechanism as its largest gradients were targeted in the attack. A distribution of the perturbed position does show - see figure 7 - that lower indices are chosen more often, which is in line with the dataset distribution.

Looking at the attention distribution a clear focus on highly sentimental words is observed spread out over the entire length of the sentence. This indicates that the soft-attention mechanism does well in attending to positions wherever they appear in the sentence.

The HotFlip attack on this model is also visualized in a lower dimensional space through a PCA as can be seen in figure 9. The variance of the first component is 20.60% and that of the second component is 9.48% showcasing that the HotFlip attack is able to create even more impact than the FGSM method.

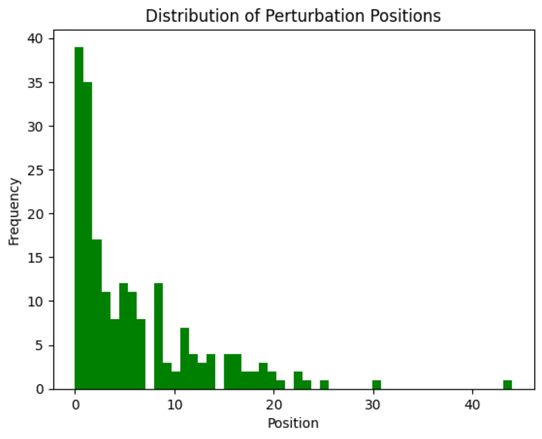

Figure 7: Distribution of perturbed positions in RNN HotFlip

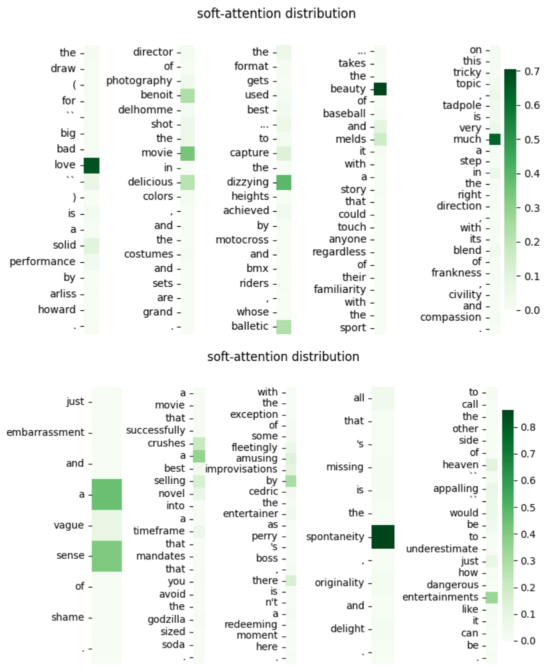

Figure 8: Attention distribution of bi-RNN Soft-Attention model

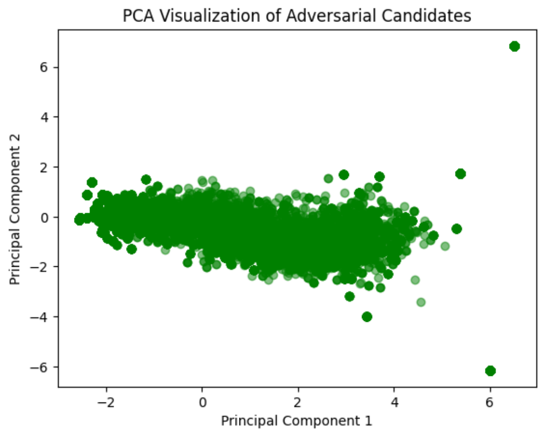

Figure 9: PCA visualisation of adversarial candidates reduced to two dimensions

4.3 Self-Attention based model

In order to perform adversarial attacks on the self-attention based classifier, the HotFlip algorithm was applied on the gradients and inputs of this model. The attacks, however, proved to be unsuccessful, so we decided to investigate the reasons for these results.

The first problem was encountered upon the creation of the adversarial test set, which consists of 72 sentences taken from the original test set. We noticed that the new words that were being generated were always the same two words. Furthermore, upon plotting the distribution of the perturbed positions, illustrated in Figure 10 it appeared that only the first words in the sentence were being flipped (namely the first 5 words, while the test set also contained longer sentences, with a maximum sentence length of 53 tokens). This led to a deeper analysis of the gradients that were being computed with respect to the input and of the attention distribution in the model's weights.

Figure 11 shows the attention distribution of the self attention model:

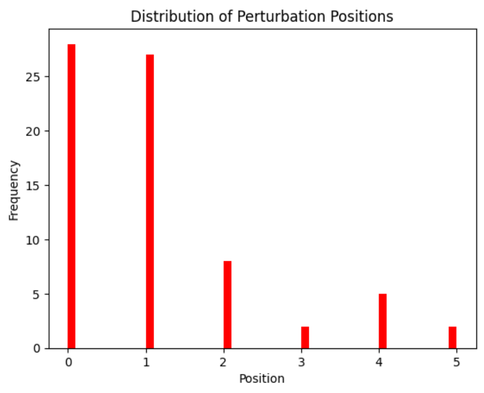

Figure 10: Distribution of perturbed positions in Self-Attention HotFlip

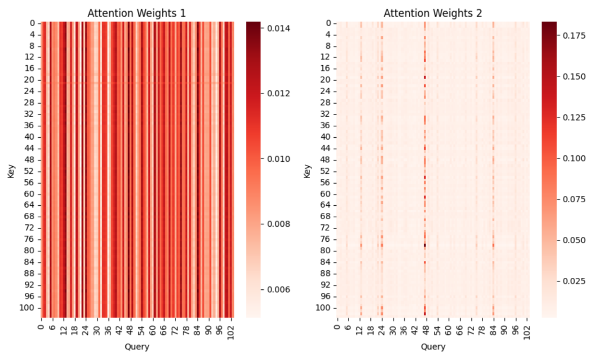

Figure 11: Attention distribution of Self-Attention model

In this heatmap it is possible to see that the distribution does not correspond to a typical attention distribution (e.g. Figure 12), that contains a more prominent diagonal, indicating that for each token more attention is being paid to the surrounding words. In this case, however, the vertical lines show that some embedding dimensions receive much higher weights than others. This clearly shows that the attention mechanism is not exploited correctly in the model.

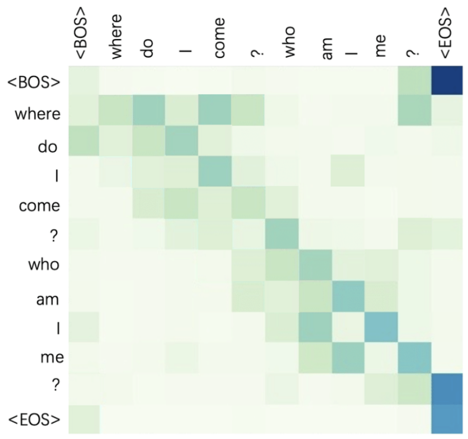

Figure 12: Typical Attention distribution (Zhan et al., 2022)

Several hypotheses have been raised to explain this behavior:

- The model architecture was not implemented correctly, thus the model was unable to learn any sentence structure or word dependencies.
- Incorrect inputs were being passed to the model. The inputs given to this model were the GloVe embeddings of the words in the sentence. Since GloVe embeddings already capture semantic similarities between words, this might have prevented the self-attention mechanism from learning any additional relationships between the words in the sentence.
- Lastly, an existing PyTorch implementation of the multi-head self-attention layer was used to build the model. While this might have initially accelerated the process of building and training the model, it made it more complicated to look into what was happening "under the hood" later on during the process, namely when it was necessary to obtain the gradients

of the weights with respect to the input. Obviously, it is not the existing implementation itself that is incorrect, but it is rather the use thereof that was not fully correct.

Despite the complications explained above, the adversarial test set was used to attack the classifier. The resulting accuracy on the adversarial dataset was equal to 31.97%.

This outcome was not foreseen due to the initial results obtained using the classifier. A test accuracy of 77% indicated reasonably good performance, far from state-of-the-art, but still quite good considering the simple architecture and short training times of the model.
Due to time constraints, it was not possible to conduct a deeper analysis of the results and to use them to improve the current model or to build and train a new model.
This outcome, however, still is a valuable learning experience. First and foremost it shows that it is important to critically analyze a model and its outputs at every step of the way and not to base one's evaluation solely on the final results such as accuracy. Second of all, it shows that it can be more useful, especially for learning purposes, to build a model up from the very beginning instead of using a readily available implementation in order to have a better understanding of the algorithm and of the variables in play.

5 Conclusion and Future Work

In this paper, we have investigated the robustness of different models against adversarial attacks in the context of sentiment analysis tasks. Our experiments employed three types of models: a feedforward model, a Bi-RNN with Soft-Attention model, and a Multi-Head Attention model. All models were trained on the same dataset and then subjected to adversarial attacks using the Fast Gradient Sign Method (FGSM) (embedded model)and Hot Flip adversarial attacks (bi-RNN and self attention).

The feedforward NGram CNN model had an initial test accuracy of 80.85%, which dropped to 52.64% after the adversarial attack, marking a 34.89% decrease. The Bi-RNN with Soft-Attention model exhibited an even larger performance drop. Initially, it achieved an accuracy of 84.29%, but after the adversarial attack, it decreased significantly to 34.52%, corresponding to a 49.77% drop. Finally, the Multi-Head Attention model, which initially had an accuracy of 77.18%, saw its performance decrease to 31.94% post-attack, indicating a substantial accuracy drop of 55%. Thus the NGram CNN model was suprisingly, the most robust to adversarial attacks while the self attention based model was the most affected. Based on our results, model complexity does not positively improve robustness of the model to adversarial attacks as the computational time and resources for bidirectional RNN and self attention were significantly higher.

Speculated reasons for results - We believe the main reasons for the results are because of the following reasons -

- Using FGSM for embedded model and hot flip for the other models - FGSM is a simpler model and disrupts the sentences much lesser than Hot Flip as mentioned by (Ebrahimi et al., 2018). The primary reason this was done was due to constraints in the problem statement but using different adversarial attacks was not experimentally sound.

- CNNs are known to capture local and position-invariant features in the input space, which could have contributed to its resilience against adversarial attacks. The Bi-RNN and Multi-Head Attention models might be more sensitive to slight perturbations in sequential data, leading to larger drops in accuracy. This suggests that, under adversarial attacks, the ability of CNNs to capture local, translation-invariant features can be advantageous compared to architectures that focus on the global sequence order.

The recommendations for future work are as follows -

- Use FGSM for all or perform hotflip on a simple LSTM model or embedding model that takes the word tokens as input and not the sum of word embeddings

- Ensure the architectures are hyperparameter tuned perfectly till the highest achievable level of accuracy and performance is achieved. This ensures the same adversarial attack is fairly affecting the best version of the model.

- Perform research on state of the art models for sentiment analysis using bidirectional RNN and self attention to get strong architecture recommendations.

The evident performance degradation under adversarial attacks across all models tested accentuates the vulnerability of the tested NLP models. This reinforces the need for adding adversarial robustness into model design considerations and motivates the need for more research in defensive mechanisms against such attacks. The variations in accuracy drops between different architectures also suggest that the model's structure influences its resilience to adversarial attacks, which is a topic to be considered when building models for sentiment analysis in human machine domains like robotics, chatbots and more.

Acknowledgments

This project was TA'd by LiJun Jiu (L.Lyu@tudelft.nl)

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(6):1137–1155.

Erion Çano and Maurizio Morisio. 2020. A data-driven neural network architecture for sentiment analysis. *CoRR*, abs/2006.16642.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6321–6328.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *Proceedings of the 2019 Network and Distributed System Security Symposium*.

Zichao Lin, Ming Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2019. A self-attention based model for sentiment classification. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 4422–4431.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

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

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Jiaao Zhan, Qian Chen, Boxing Chen, Wen Wang, Yu Bai, and Yang Gao. 2022. Non-autoregressive translation with dependency-aware decoder.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4).

Yichao Zhou, Junjie Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4862–4872.