First-Derivative Saliency (Explainability) Method for a Text Perturbation Attack

Janaki Viswanathan
Department of Computer Science
Universität des Saarlandes
Saarland, 66123 Saarbrücken
s8javisw@stud.uni-saarland.de

Abstract

Neural networks have been used in various NLP tasks to achieve impressive results when compared to most of the machine learning models. However, when it comes to explaining the model or rationalising the output, it has always fallen short when compared to machine learning models. To address this, Li et al. (Li et al., 2015) proposed many ways of visualizing and understanding neural models in NLP. The first derivative input saliency method measures how much each word has contributed to the final decision. This helps in identifying important words which influences the output. Furthermore, as an attack method, we replace those important words using GloVe embeddings to flip the classes showing how efficiently the first derivative input saliency method identifies the important words as well as how vulnerable neural networks could be. This article is written as part of the "Explainability Methods for Neural Networks" seminar at Saarland University that took place in WS2020/2021 (EMN). The article primarily focuses on the paper "Visualizing and Understanding Neural Models in NLP" by Li et al. (Li et al., 2015).

1 Introduction

Neural networks are being widely used in several domains due to their powerful capacity for modelling complex input patterns. However, it is applied in a black box manner making it difficult for humans to understand the model (Samek et al., 2017). A well-trained model could identify patterns in data which are otherwise inaccessible to human experts. On the other hand, it is also crucial to understand the rationale behind the model before implementing it in high-risk scenarios such as healthcare or autonomous driving. Verifying the robustness of a model helps in understanding if the model has learnt something meaningful from the data or not. To explore the robustness of neural networks, adversarial attacks are widely being used.

Since the availability of pre-trained embeddings, neural networks have been widely used in the Natural Language Processing (NLP) domain. Several explainability methods have been introduced to understand the NLP models.

One such way to rationalise the output is considering *compositionality* of the text. Compositionality is the principle that the meaning of one whole sentence is determined by the meanings of small phrases or words which constitutes that sentence (Wik). By understanding the compositionality of a text, we can identify which phrases or words of the text influences the output making it easy to explain the final decision. This convinces people to trust the model more.

Recently, due to the success of the deep neural networks, several deep learning models are provided as a cloud based service to the users. Though these are provided as a black-box model without sharing any information about the model or its architecture with the end users, they are vulnerable to many attacks. The basic requirement to carry out such an attack is to identify what influences the output the most and tweak the input carefully to flip the label.

Many of these methods have been explored in the image domain and have been successful. One of the famous methods to perturb the input data to be misclassified by the model was introduced by Goodfellow et al.,(Goodfellow et al., 2014b).

However, due to the discrete nature of the text data and the presence of an embedding layer to represent it, it has been difficult to apply the gradient-based methods to the NLP domain to generate adversarial samples (Li et al., 2020a). Two important factors to be noted while carrying out such an attack on text input are the following:

- The perturbed text should be grammatically correct and preserve the context;
- The perturbed text should be semantically consistent with the original text while misleading the classification model.

In this paper, we will explore how the first-derivative input saliency method - an explainability method for NLP introduced by Li et al. (Li et al., 2015) could be used to attack a text classification model.

2 Related Work

A brief description of several explainability methods and attack methods have been described in this section.

2.1 Explainability Methods

A wide range of techniques and methods have been proposed to explain neural networks. It can be broadly classified into three types:

- Neural visualization techniques - to visualize each layer and basic units
- Modelling methods - which quantifies how much each feature roughly influences the output
- Explanation generating methods - the short explanations being generated helps in rationalising the final decision

Several neural visualization methods have been explored in vision, especially for Convolution Neural Networks (CNNs or ConvNets). It involves mapping different layers of the network back to the original image input to interpret how each unit contributes to the final decision. In NLP, words are analogous to units and hence word vectors rather than single pixels are basic units.

Word embeddings could also be used to visualize and understand the model. Embeddings could be projected into a two dimensional space to see how similar words cluster together (Faruqui and Dyer, 2014). Recently, an open sourced python library named Ecco created by Jay Alammar (Ecc) helps in creating interactive visualizations to explore what the NLP language model has learnt.

Apart from visualizing the neural units to understand what has been learnt, many modelling methods have been introduced to understand NLP models. LIME is a method proposed by Ribeiro et al. (Ribeiro et al., 2016) which is an additive feature attribution method. The LIME method interprets individual model predictions based on locally approximating the model around a given prediction. It can be summed up in four steps (Lim):

- Generate data points around the instance that has to be explained
- Use the machine learning classifier built using the actual data set to predict classes for the newly generated data points
- Compute distances between the instance being explained and each generated data points and compute weights(importance) of the generated instances
- Use the generated data set, its predictions and their importance to fit a simpler and interpretable linear model

In 2017, Lundberg et al. (Lundberg and Lee, 2017) proposed a unified approach to interpret model predictions named as SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. SHAP is a unified measure of feature importance. It combines insights from some already available additive feature attribution methods such as LIME (Ribeiro et al., 2016) and DeepLIFT (Shrikumar et al., 2016) and approximates it to calculate the SHAP value.

By combining several of these methods, SHAP has the following properties (1) Local accuracy (2) Missingness (3) Consistency which are otherwise not possessed by the methods on their own. However, SHAP consumes a lot of time and computational power and assumes that the features are all independent of each other which is not ideal for text data.

Generative Explanation Framework (GEF) was proposed by Liu et al. (Liu et al., 2018) for NLP models. The GEF helps in building trustworthy explainable text classification model capable of explicitly generating fine-grained information for explaining their predictions. They consider a task of rating a product based on a review text.

The GEF framework consists of an Encoder-Predictor architecture to predict classes and a Generator to generate fine-grained explanations. However, since the output from them are independent of each other, they propose using an Explanation Factor.

The Explanation factor is measured using three different probabilities calculated from a single input - (1) predicted probability from the Encoder-Predictor model, (2) classified probability from a pre-trained classifier which takes the fine-grained explanation as input where the classifier is pre-trained on the original golden explanations and (3) golden probabilities obtained by passing the golden explanations as input to the pre-trained classifier.

The Explanation Factor is then measured as follows:

$$\text{EF(S)} = |\tilde{p}_{classified} - \tilde{p}_{gold}| + |\tilde{p}_{classified} - \tilde{p}_{pred}| \quad (1)$$

Liu et al. (Liu et al., 2018) also propose Minimum Risk Training to minimize the expected loss which is a sum of the classification loss and explanation generation loss multiplied by the explanation factor.

2.2 Attack Methods

To verify the robustness of a model, several attack methods have been proposed. An evasion attack happens when a machine crafted or manipulated data which appears to be unaltered for humans is fed as an input to a model, for example a classifier model and it throws the classifier off. One of the famous adversarial attack methods - 'fast gradient sign method' was introduced by Goodfellow et al. (Goodfellow et al., 2014a) (Goodfellow et al., 2014b).

The Fast Gradient Sign Method (FGSM) is a method to generate adversarial examples by adding a small amount of carefully calculated noise to the input image to flip the label. The noise is calculated as:

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \quad (2)$$

where θ represents the parameters of the model, x the input to the model, y the output class associated with x and $J(\theta, x, y)$ is the cost used to train the neural network.

While training a neural network using backpropagation, the gradient of the loss function with respect to the weights are calculated to tweak the weights to minimize the loss. In FGSM, the η is measured as the sign of the gradient of the loss function with respect to the input. By taking the gradient with respect to the input, we will be able

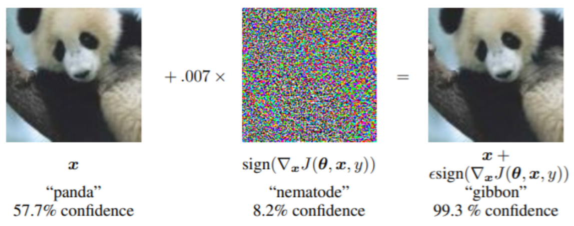
$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 1: The figure demonstrates how the output label has been flipped from 'panda' to 'gibbon' with high confidence just by adding a small carefully calculated noise to the input. Figure from (Goodfellow et al., 2014b))

to understand how to change the input to increase or decrease the loss function.

A scalable parameter ϵ is used to bound the amount of change to the input ensuring that the addition of noise is not easily identified by humans.

In Figure 1, on adding the calculated η to the input image, the class label has flipped from 'panda' to 'gibbon' with as high confidence as 99.3%

A lot of such methods have been proposed in the vision domain. However, transferring these ideas to text has been challenging because of the discrete nature of text data.

Some of the attack methods in the NLP domain are based on heuristics. For example, one of the data augmentation methods - Back Translation could be used. It involves translating a text to another language and translating it back to its original language. There is a very high possibility that there will be at least a little change in the text while maintaining the semantic consistency of the text given the nature of text data. However, most of the heuristic methods suffer from the problem of semantic inconsistency.

Another method to attack an NLP model is to replace the important words with a word having a similar meaning is thesaurus-based substitution. WordNet could be used to obtain a word with similar meaning and POS tag.

Recently, an attack method in the NLP domain - BERT-Attack was introduced by Li et al. (Li et al., 2020b). It uses BERT embeddings to attack a model trained using BERT embeddings. Using BERT embeddings to attack a model helps in preserving the semantic consistency of the model.

In the next section, we will discuss how FGSM by Goodfellow et al. (Goodfellow et al., 2014a) (Goodfellow et al., 2014b) and the first-derivative saliency method proposed by Li et al. (Li et al.,

2015) share the same ideas of using first derivative of the loss function with respect to the input to achieve two different goals.

3 Input Saliency Method

The first-derivative input saliency method is a strategy inspired by the back-propagation strategy in vision (Erhan et al., 2010). By calculating the first derivative of the loss function with respect to the input, it helps in measuring how much each input unit contributes to the final decision.

To understand and rationalise the model in NLP, it is helpful to understand which words influence the output the most. This could be measured by calculating the direction where the function responds the strongest.

For a classification model with embeddings E for input words associated with an output class label c, we obtain a score $S_c(e)$ where

$$S_c(e) \approx w(e)^T e + b \tag{3}$$

where $w(e)$ is the derivative of S_c with respect to the embedding e.

$$w(e) = \frac{\partial(S_c)}{\partial e}|_e \tag{4}$$

The absolute value of the derivative could be interpreted as the measure of how much one particular dimension of the word embedding contributes to the output class. The saliency score is thus calculated as:

$$S(e) = |w(e)| \tag{5}$$

By obtaining a saliency score for each word of the input text, we can identify the important words that influence the output. This helps in a better understanding and interpretability of the model.

Both FGSM and the input saliency method uses the first derivative of the loss function with respect to the input to achieve different objectives.

The former method is to attack a neural network and the latter to identify and visualize important words which influence the output. The former perturbs the data by adding the calculated noise back into the input which leads to flipping the output label. However, the latter identifies the important units which contributes to the output to explain and understand the neural models.

In the next section, we will see a text perturbation method which uses pre-trained embeddings to replace these important words to attack a text classification model.

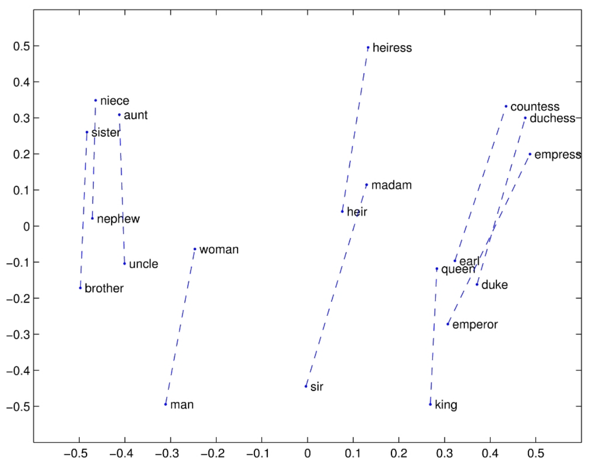

Figure 2: GloVe vectors of the words ‘man’ and ‘woman’; ‘king’ and ‘queen’ are closer to each other in the vector space. Figure from (Glo))

4 Text Perturbation Attack

The Fast Gradient Sign Method (FGSM) discussed in Section 2.2 adds the gradient back to the input in order to flip the class label. This is possible for continuous data like image input. However, for discrete data such as text, this is not a possibility.

With the introduction of Global Vectors for word representation (GloVe) (Glo) vectors, we can use the same idea roughly to text data. GloVe vectors are pre-trained word embedding vectors for words. The GloVe vectors possess a property that the vectors for words with similar meanings are represented closer to each other in the vector space.

With this property observed in GloVe vectors, we can replace the important words obtained from the input saliency method with similar words from the pre-trained GloVe word embedding. The similarity could be measured using one of the similarity metrics such as euclidean distance.

One of the famous examples to understand this property in GloVe is described in the Figure 2. The vectors corresponding to words ‘king’ and ‘queen’; ‘man’ and ‘woman’ which are similar are represented closer to each other in the vector space. Additionally, when the vector representing ‘woman’ and ‘king’ are added together, it will be roughly equal to the vector representing the word ‘queen’.

Assuming that it is a white-box model where the model parameters and model architecture are available for the attacker, they can calculate the first-derivative saliency scores for the considered input text and attack the model by replacing the words with similar words which can be obtained using the pre-trained GloVe embeddings.

5 Dataset

The explainability method and the attack method has been applied on a text classification task more precisely on a sentiment analysis task. The IMDb dataset available in the PyTorch library has been used to carry out the experiment.

The original dataset consists of 25000 training samples and 25000 test samples. A 70:30 split was made in the train set to create a validation set of 7500 validation samples and 17500 training samples. The original test set was retained. It is a binary sentiment classification dataset with labels 'pos' and 'neg' referring to positive and negative review texts.

6 Experiment and Results

The input review text was tokenized using 'spacy' library (spa) available in Python which tokenizes text based on rules specific to each language. The pre-trained word embedding GloVe - Global Vectors for word representation which was trained on 6 billion tokens and has 100-dimensional vectors was used to initialize word embeddings. Words with similar semantic meaning are represented closer in the vector space by the GloVe vectors (Glo).

6.1 Text Classification Model

The IMDb dataset consists of review text as input and sentiment labels as output. Unlike images with fixed-sized input and fixed-sized output, the NLP tasks can have varied-sized input and even varied-sized output. In the considered task - text classification, we only have to deal with varied-sized input. Furthermore, text data are inter-dependent and needs to be treated as sequences to understand the context of the entire text. Recurrent Neural Networks (RNNs) allow to operate over sequences of vectors.

With the review text as an input and the sentiments as an output, a bidirectional Long Short-Term Memory (biLSTM) model was built. Long Short-Term Memory (LSTM) is a type of recurrent network which overcomes the long-term dependency problems in RNN. A bidirectional LSTM helps in capturing the context and meaning of a sentence better leading to a better sentiment prediction. Furthermore, multi-layer RNN was used with two biLSTM layers.

With multiple bidirectional LSTM layers, we will have several parameters in the model which might lead to overfitting. To address overfitting,

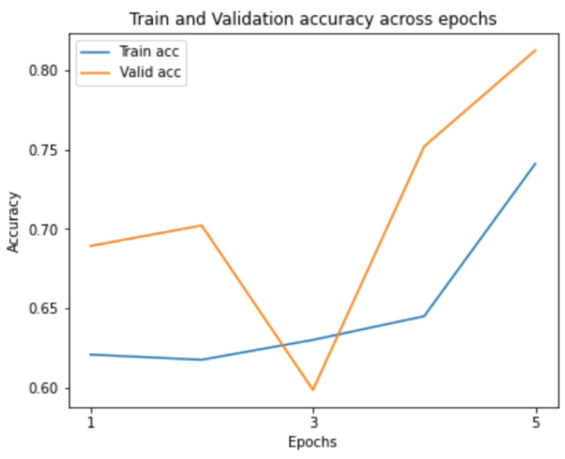

Figure 3: Accuracy for the Train and the Validation set over five epochs trained using a biLSTM model with Dropout as a regularizer.

Dropout regularization method was used. Dropout randomly sets 0 to some neurons during a forward pass. The Dropout regularization method is empirically found to improve the performance of a model.

Adam optimizer was used to train the text classification model. Unlike Stochastic Gradient Descent (SGD) which uses a same learning rate to update the parameters, Adam optimization derived from the phrase 'adaptive moments' adapts the learning rate for each parameter, giving parameters that are updated more frequently lower learning rates and parameters that are updated infrequently higher learning rates.

For the loss function, binary cross entropy with logits was used. A binary cross entropy loss function is used on a bounded prediction output. The output is bounded between 0 and 1 by using a sigmoid or logit function.

To evaluate the performance of the model, accuracy was used and the model was trained for 5 epochs.

From Figure 3 and Figure 4, we can observe that for the training data set, the accuracy has steadily increased for each of the epoch and the loss has steadily decreased for each of the epoch. For the validation set, we see that there is a sudden increase in the loss and decrease in the accuracy at the third epoch. However, by the end of five epochs, the loss is significantly lower than the training loss and the accuracy is significantly higher than the training accuracy.

For the train set, an accuracy of about 81% was achieved. And for the test set, an accuracy of about

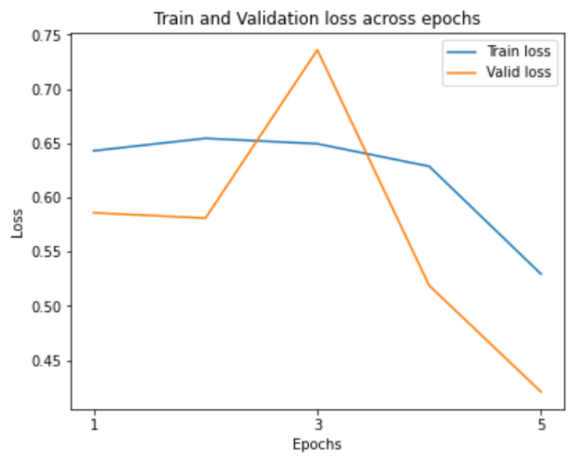

Figure 4: Loss for the Train and the Validation set over five epochs trained using a biLSTM model with Dropout as a regularizer.

85% was achieved.

6.2 Input Saliency Method

Given an input text, the trained text classifier model was used to calculate the score. Once the score was calculated, the first derivative of the score with respect to the input was calculated and its absolute value was taken to measure saliency of each word.

The saliency of each word represents how important it is to influence the output sentiment label. The words with high saliency scores are important words which when perturbed could flip the label.

Once the saliency scores for each of the word is obtained, a heat map could be used to represent, visualize and understand the effect of each word on the output. Through visualization, we can understand how the neural model has learnt the compositionality of the whole text to achieve the task.

Example-1:
Input: "The movie is fantastic I really like it."
Output: "pos"

In Figure 5, we can see the saliency heatmap of the above positive example. Each row corresponds to the saliency score for each word in the sentence and each grid represents each dimension. The darker the shade of blue it is, higher the influence of the corresponding word to the output. We can see that the words 'fantastic' and 'really' which are positive have been picked up as words of high importance or high influence over the output.

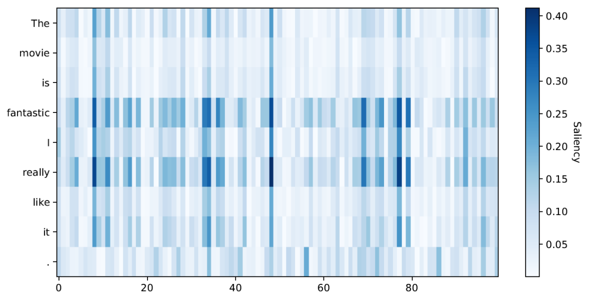

Figure 5: Saliency heatmap for 'The movie is fantastic I really like it.'. Each row corresponds to saliency scores for the correspondent word representation with each grid representing each dimension.

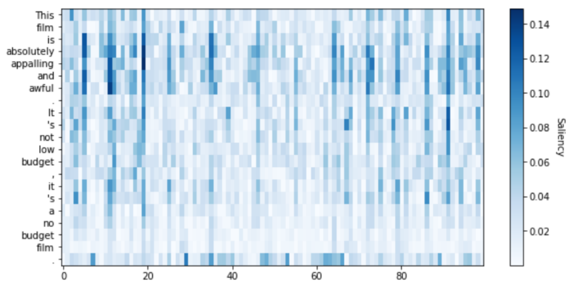

Figure 6: Saliency heatmap for 'This film is absolutely appalling and awful. It's not low budget, it's a no budget film.'. Each row corresponds to saliency scores for the correspondent word representation with each grid representing each dimension.

Saliency scores:
('fantastic' : [0.4587])
('really' : [0.3538])

Example-2:
Input: "This film is absolutely appalling and awful. It's not low budget, it's a no budget film."
Output: "neg"

In Figure 6, we can see the saliency heatmap of the above negative example. The words 'absolutely', 'appalling', 'awful' which are words with negative sentiment have a high saliency score when compared to the other words.

Saliency scores:
('appalling' : [0.1490])
('absolutely' : [0.1422])
('awful' : [0.1395])

6.3 Text Perturbation Attack

The text perturbation for the 'important' words identified by the first-derivative saliency method was done by replacing it with a similar word from GloVe embedding. The open-source library Gensim was used to obtain a list of most similar words. GloVe embeddings trained on wiki data was used to obtain similar words. The following table lists the original words and its corresponding similar words obtained from the GloVe embeddings:

Original word	GloVe word
fantastic	amazing
excellent	quality
exceptional	extraordinary
beautiful	lovely
bad	worse
film	movie
horrible	terrible
appalling	deplorable
awful	horrible
disappointment	frustration

Table 1: Examples of most similar words obtained from Glove embedding

Replacing words such as 'is' or 'the' might not be semantically consistent. Hence, only words which are not stop words were replaced. Also, punctuation marks were not replaced. Stopwords from the nltk package in python was considered.

If the text is too long, a hyper-parameter could be used to control the amount of perturbation while still flipping the class. For example, one can replace only 30% of the text which is not a stopword or a punctuation mark with a similar word from the GloVe embedding to attack the model while still retaining the semantic consistency . The examples considered here were short and hence it wasn't implemented.

For all the sentences that were tested for analysis - some of which have been listed in Table 2, the sentiment of the text flipped on replacing the important words identified by the first-derivative input saliency method with a similar word obtained from the pre-trained GloVe word embedding.

True output	Original text	Perturbed text
'neg'	This film is **absolutely appalling** and **awful**. It's not **low budget**, it's a no **budget film**.	This movie is **obviously deplorable** and **horrible**. Ithas not **higher** spending, ithas a no **spending movie**.
'pos'	The **movie** is **fantastic** I **really like** it.	The **film** is **amazing** I **something even** it.
'neg'	This **film** is **horrible**!	This **movie** is **terrible**!
'pos'	This **film** is **excellent**	This **movie** is **quality**
'neg'	I wish I knew what to make of a movie like this. I must say that the scenes on the beach struck me as so stereotypical in so many ways. The film devolves into a succession of visual displays and not too much else. A **disappointment**.	I wishes I why what to making of a movies even this. I should believe that the scene on the beaches hit me as so stereotype in so some how. The movie devolving into a dynastic of imagery display and not too even nobody. A **frustration**.

Table 2: Example of word replacement with GloVe replacement

7 Discussion

From the analysis done in Table 2, we can observe that the methods work perfectly for short sentences but it fails to make much sense for longer sentences.

First-derivative input saliency method has a strong theory behind it but it can only be used when the model architecture and model parameters are known. It is assumed that it is a white-box model to use this method. If we would like to calculate saliency score to attack a model which is available online as a service through cloud-based services, we cannot do so without knowing the information about the model.

For this reason, one has to resort to other methods of identifying important words which does not require the information of the model. For example, one can use inclusion and elimination method to identify important words for a black-box model as such.

For an input sentence: 'This film is excellent!', we can pass all combinations of the sentence without one word - 'film is excellent!', 'This is excellent!' and so on and observe how the output probability or class label gets influenced. The words which has high influence will be considered as the important words.

In Table 2, we can also observe that the adjective replacements preserve the semantic consistency of the sentence whereas for words which act as verbs, the replacement word suggested by GloVe does not preserve the semantic consistency on replacement.

As a replacement strategy, WordNet could be used based on the POS tags. For replacing a word with a POS tag of ADJ, another word with the same POS tag could be chosen. Additionally, we can also limit ourselves to only replacing certain POS tags such as ADJ / NOUN to retain the semantic consistency of the text.

Another embedding which captures the context of the text - BERT embedding could also be used. This will preserve the meaning of the sentence while being grammatically correct.

We could also use a constraint based strategy which constrains on the BERT sentence embedding to check the consistency of the text after replacing important words with similar words from GloVe embedding.

8 Conclusion

In this report, it was discussed along with results how the explainability method proposed by Li et al. (Li et al., 2015) - the First-derivative input saliency method could be used along with a replacement strategy using GloVe word embedding could act as an attack mechanism for a text classification task.

It was observed that the saliency method could be used for only white-box methods and the GloVe replacement strategy performs well for words with ADJ POS tags and only for short sentences. Attack methods for a black-box model and replacement strategies for input text which are longer and with other POS tags were also discussed.

References

Ecco: An open-source python library that creates interactive visualizations allowing you to explore what your nlp language model is thinking. https://www.eccox.io/.

Explainability methods for neural networks (ws 2020/2021). https://sites.google.com/view/emnn-ws-2020/home#h.s5oampi1wpep.

Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/.

spacy. https://spacy.io/usage/linguistic-features#tokenization.

Wiki: Principle of compositionality. https://en.wikipedia.org/wiki/Principle_of_compositionality.

Youtube: Interpretable machine learning with lime - how lime works? 10 min. tutorial with python code. https://www.youtube.com/watch?v=vz_fkVkoGFM.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014a. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014b. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020a. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Hui Liu, Qingyu Yin, and William Yang Wang. 2018. Towards explainable nlp: A generative explanation framework for text classification. *arXiv preprint arXiv:1811.00196*.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.