Classification of Factual and Non-Factual Statements Using Adversarially Trained LSTM Networks

Daniel Obembe
Department of Computer Science and Engineering
The University of Texas at Arlington
daniel.obembe@mavs.uta.edu

Supervisor: Dr. Chengkai Li
Committee Members: Dr. Dajiang Zhu and Dr. Won Hwa Kim

Abstract

Being able to determine which statements are factual and therefore likely candidates for further verification is a key value-add in any automated fact-checking system. For this task, it has been shown that LSTMs outperform regular machine learning models, such as SVMs. However, the complexity of LSTMs can also result in overfitting (Gal and Ghahramani, 1997), leading to poorer performance as models fail to generalize. To resolve this issue, we set out to utilize adversarial training as a way to improve the performance of LSTMs for the task of classifying statements as factual or non-factual. In our experiment, we implement the adversarial training of an LSTM using the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) (Miyato et al., 2017) for the generation of adversarial examples. To implement adversarial training, we normalized the model inputs, which in this case are vector representations of words, called word embeddings. We also modified the loss function by adding a perturbation. In other words, we trained the neural network to correctly classify perturbed input values. We discovered that the adversarially trained LSTM outperforms the regularly trained LSTM on some performance metrics, but not all. Specifically, the adversarially trained LSTM shows increased precision with respect to sentences that are classified as check-worthy, and increased recall with respect to sentences which are classified as not being check-worthy.

1 Introduction

Recently, machine learning based approaches have been used to automate some of the tasks in the fact-checking process. One such task is the identification of factual claims from a collection of statements (Hassan et al., 2017). This task is critical, both because it is the entry-point for the fact-checking process and because of the high velocity and volume of content generated by information sources. Due to these factors, machine learning based approaches for the identification of factual claims have received a significant amount of interest from industry and researchers alike.

One class of models which have recently shown some promise in the identification of factual claims are Recurrent Neural Networks (RNNs) (Karadzhov et al., 2017), which are able to preserve information over multiple inputs from a sequence. More specifically, LSTMs (Hochreiter and Schmidhuber, 1997), which are a specialized form of RNN, as well as LSTM-derived architectures have proven useful for this task because of their ability to model long-term dependencies. For example, in (Jimenez and Li, 2018) we see the successful demonstration of an LSTM-based approach outperforming the then benchmark SVM-based approach (Hassan et al., 2015) for identifying sentences with salient factual statements.

However, due to the complexity of LSTMs, one issue faced in their training and usage is the tendency to overfit to training data (Gal and Ghahramani, 1997). One way to reduce the likelihood of this occurring is to collect larger and more representative datasets on which to train the model. While this can help, it may also be costly and in some cases, non-feasible. For the identification of factual statements, utilizing this method will involve not only collecting and processing a larger quantity of potentially relevant statements from diverse sources, but also finding individuals who will be willing to review and label this data. This is costly, both monetarily and in terms of manpower. Therefore, for this particular use-case, in which collecting more data is suboptimal at best, other methods are needed to train an LSTM based model that generalizes well to unseen data.

In this thesis, we discuss the applicability of adversarial training as one possible solution to the

creation of generalizable LSTMs for the identification of factual statements, since this method shows promise for use-cases in which the availability of sufficient representative training data is a constraint. We implement this approach using adversarial training techniques proposed by both (Miyato et al., 2017) and (Goodfellow et al., 2015).

In this study, the problem we are solving is the classification of sentences into two exclusive categories. These categories are:

- Non-Check-Worthy Sentence (NCS): These are sentences that either do not contain factual claims, or contain trivial factual claims that are unimportant and not of interest to the general public. Some of the types of statements that fall under this category are subjective statements, questions, declarations of emotion, and so on. Some example sentences are as follows:
    1. "I really enjoyed this pie."
    2. "This lake is cold and full of pebbles."
- Check-worthy Factual Sentence (CFS): These are sentences that are both factual and of interest to the general public. Examples of the kinds of statements that fall under this category are factual claims that rely on or make claims about statistical facts and statements that touch on historical information. Some example sentences are as follows:
    1. "The senator from Oklahoma voted for the energy independence bill."
    2. "Over 40% of people in America would be exposed to the novel coronavirus."

Given these two classes, the objective of this study is to use adversarial training as a means for improving the performance of an LSTM model in the classification of statements as being either factual or non-factual. To accomplish this objective, we modify the loss function by adding in two conflicting constraints, and we train the network to minimize this modified loss function. In the context of an LSTM model, the loss function is a function which maps the predictions to some real number. This number is used to evaluate the model's performance, and it is also used by the training algorithm to optimize the model during training. In this study we used a cross-entropy loss function (Janocha and Czarnecki, 2017). The constraints refer to modifications made to the loss function which penalize the model for misclassifying adversarially modified inputs. It is the addition of these constraints which makes the training method adversarial and which ideally would also result in the creation of a more robust model.

The main contribution of this study is that, even though other papers have used adversarially trained LSTMs for sentence classification (Miyato et al., 2017), this is the first time an adversarially trained LSTM has been used to identify factual claims.

2 Background

2.1 Recurrent Neural Networks

RNNs are sequential neural networks which model temporal dependencies via a hidden state. This hidden state is a function of both the current input and the previous input to a RNN cell. However, despite the ability of RNNs to capture temporal dependencies, one impediment to their continued utilization is their inability to model long-term temporal dependencies because of the vanishing gradient problem (Hochreiter, 1998).

This can be especially problematic for NLP tasks, where an ability to understand long-term dependencies between words at various, sometimes distant, positions in a statement is critical to correctly identifying the statement as being factual or non-factual.

2.2 LSTMs

LSTMs are a specialized form of RNN which were designed to resolve the vanishing gradient problem that frequently occurs with regular RNNs. They accomplish this by using a set of pairs of sigmoid layers and point-wise multiplications, where each pair is referred to as a gate. A typical LSTM has three such gates, with each time-step's hidden state, s_{t-1}, being passed through the gates to compute the hidden-state for the subsequent time-step, s_t.

By regulating how their cell states are updated, LSTMs provide a way to preserve information over many time-steps and thus to enable the modelling of long-term dependencies. However, they are not without flaws. LSTMs, due to their large size, can over-fit to training data, and this can negatively affect model performance. Adversarial training is being explored here as one poten-

tial method to counter this tendency towards overfitting.

The type of LSTM used in this thesis is a Bidirectional LSTM (called BiLSTM) (Graves and Schmidhuber, 2005). BiLSTM extends the typical LSTM architecture by stacking two LSTMs. One of the stacked LSTMs reads the input sequences in their original order, while the other one reads the input sequences in their reversed order (Schuster and Paliwal, 1997).

2.3 GloVe Embeddings

The sentences used in this study were encoded using GloVe word embeddings (Pennington et al., 2014). A GloVe model can be defined as a function which maps words into a semantically meaningful vector-space.

Due to the large amount of data required to train a good GloVe word embedding from scratch, pretrained word embeddings have been made available for download under the Public Domain Dedication and License. This is a secondary benefit of using GloVe and, for this study, we used a pretrained word embedding as the feature set for our experiments.

2.4 Related Works

LSTMs have an extensive literature as they have been successfully applied to a broad range of machine learning use-cases in which the processing of sequential data is required. In addition, there are previous studies that have looked specifically into the use of LSTMs for classification of text, such as (Lai et al., 2015). This thesis employs a similar approach, but with objectives similar to (Jimenez and Li, 2018), in that it constrains the domain of inquiry by looking specifically at the performance of LSTMs on the classification of statements with respect to their factuality.

With regards to the utilization of word embeddings to encode the text inputs, it is important to note that this method is now frequently used and is also the generally preferred method when using RNNs for text-based applications (Young et al., 2018). In particular, there has been research into the use of word embeddings to improve a neural network's performance when there is a scarcity of training data, e.g., (Qi et al., 2018). While this is a possible benefit of using word embeddings, this thesis focuses instead on using adversarial training to build a more robust model.

Finally, in recent years, there has been a significant amount of research into adversarial methods for training neural networks. This interest has been spurred by two main applications which are its utilization in Generative Adversarial Networks (Goodfellow et al., 2014) and the use of adversarial examples to make neural networks more immune to malicious attacks (Yuan et al., 2017). This thesis differs from these in that we pursue a different objective, using adversarial training solely as a generalization method.

3 Methodology

Although there are various methods for generating adversarial examples, Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) has become one of the more frequently used techniques. This is due to its being conceptually straightforward, as well as its being practical to implement. FGSM is also versatile, as it has been successfully used to generate adversarial examples for a broad range of machine learning models, such as CNNs (Yuan et al., 2018) and BIRT (Meng et al., 2020). Because machine learning libraries (including Tensorflow) do not yet have built-in classes or functions for the adversarial training of models, we constructed the pipeline for its implementation in this thesis. Within this pipeline we used FGSM to generate adversarial examples.

The main difference between an adversarial training pipeline and a normal machine learning pipeline is that two separate losses need to be calculated from the same initial input and used to create a composite loss, which is then used to optimize the model. Also, because the target loss is a composite loss, it is not possible to use TensorFlow's fit() method. To word this differently, the flow of information between the individual computational blocks that define the adversarial training process is atypical in comparison to the standard model optimization pipeline. One requirement for adversarial training is that an adversarial input, x_{adv}, needs to be generated. This was done by modifying the original word embedding in such a way as to maximize the probability of its being misclassified by the model. To obtain x_{adv}, the gradient of the BiLSTM's loss with respect to the model's input was computed. The signs of all the elements in the gradient vector were then extracted and multiplied by the perturbation constant, ϵ, to generate the adversarial vector, r_{adv}. This adver-

sarial vector was then added to the original word embedding to create the adversarial input. The equation for x_{adv} can therefore be represented as:

$$x_{adv} = x + \epsilon * Sign\left[\nabla_x Loss(\tilde{y}, y)\right] \quad (1)$$

Code snippets from the relevant functions used in computing the losses are shown in Figure 1.

```
def loss(model, x, y):
    y_hat = model(x)
    return loss_object(y_true=y, y_pred=y_hat), y_hat

def grad(model, inputs, targets):
    with tf.GradientTape() as tape:
        loss_value, y_hat = loss(model, inputs, targets)
    return loss_value, tape.gradient(loss_value,
    model.trainable_variables), y_hat

def adv_grad(model, inputs, targets):
    with tf.GradientTape() as tape:
        tape.watch(inputs)
        loss_value = loss(model, inputs, targets)
    gradient = tape.gradient(loss_value, inputs)
    return tf.sign(gradient)
```

Figure 1: Computing the adversarial gradient

Once the adversarial input was computed, it was then used to calculate the total loss. This was done by first computing the adversarial loss, $Loss(\tilde{y}_{adv}, y)$, and the regular loss, $Loss(\tilde{y}, y)$, and then summing the two losses to obtain the total loss.

The effect of this modification is that the term we add to the cost function is:

$$\begin{aligned} &\log p(y|x + r_{adv}; \theta) \\ &where\ r_{adv} = \arg\min_{r,||r||\leq\epsilon} \log p(y|x + r; \theta) \end{aligned} \quad (2)$$

in which r_{adv} is the perturbation to the input, θ is equal to the parameters of the classifier - which are held constant, and x is the input.

From this equation, we can see that r_{adv} is the change to the input (that is, the perturbation) which maximizes the probability that the neural network will misclassify the input. We can also infer from the equation that the adversarial perturbation can be computed in a way that is conceptually similar to how we compute weights when training a model, i.e., using gradient descent. To do so, we hold the model parameters θ constant and take the derivative of the output y with respect to the input x. The reverse of the resulting gradient is then the direction in which we must modify the input to maximize the change in the output, y. Additionally, in a fashion similar to how we implement gradient descent when training the model, we must set a bound on the extent to which we perturb the input in each iteration. This is done via the ϵ term in the equation, which is a tunable hyperparameter that determines the magnitude of the movement in the direction set by the gradient. To adversarially train the model, at each step we must train the model to be robust to perturbations using the above equation.

However, one difficulty is that minimization with respect to the adversarial perturbation is computationally intractable for most neural networks. To resolve this, Miyato (Miyato et al., 2017) suggests approximating the adversarial perturbation by linearizing the model around $\mathbf{x}$. The resulting equation is then:

$$\begin{aligned} r_{adv} &= -\epsilon \frac{\mathbf{g}}{||\mathbf{g}||_2} \\ where\ g &= \nabla_x \log p(y|\mathbf{x}; \Theta) \end{aligned} \quad (3)$$

which can the be more easily computed via backpropagation.

```
x = embedding_model(x_training)
regular_loss, _, y_hat = grad(model, x, y_training)
adversarial_gradient = adv_grad(model, x, y_training)
adversarial_x = x + epsilon*adversarial_gradient
adversarial_loss, _, _ = grad(model, adversarial_x,
y_training)
total_loss = regular_loss + adversarial_loss
```

Figure 2: Computing the adversarial and total loss

The model weights are then optimized using the total loss. For this study, the RMSprop optimization algorithm (Ruder, 2017) was used. However, it will be an interesting area of further exploration to evaluate the effects of using other optimization algorithms. Code snippets showing the computation of the loss are in Figure 2, and code snippets showing the computation for the gradient of the loss with respect to the model weights are in Figure 3.

```
# Retrieve the gradients w.r.t.loss.
grads = tape.gradient(total_loss, model.trainable_weights)

# Run one step of gradient descent.
optimizer.apply_gradients(zip(grads, model.trainable_weights))
```

Figure 3: Optimizing the model weights using the total loss

A simplified depiction of the algorithm for adversarially training BiLSTM is shown in Algo-

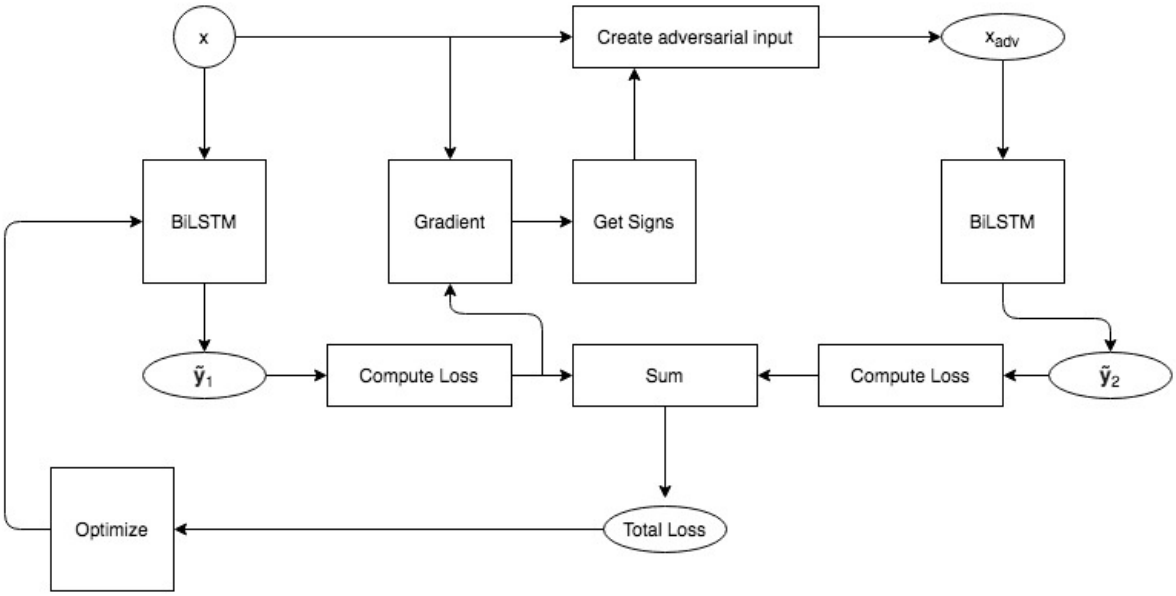

Figure 4: Our Implementation of the FGSM Adversarial training process

rithm 1. It is assumed in the algorithm that x is already a word embedding.

Algorithm 1: Adversarial Training

for *n epochs* **do**
- Call model on x: $y \leftarrow$ BiLSTM(x);
- Compute regular loss:
- $L_r \leftarrow Loss(\tilde{y}_1, y)$;
- Compute gradient of regular loss:
- $g_r \leftarrow$ Gradient(L_r,x);
- Extract signs from gradient:
- $s \leftarrow$ Sign(g_r) ;
- Create perturbation: $r_{adv} \leftarrow \epsilon * s$;
- Create adversarial input:
- $x_{adv} \leftarrow x + r_{adv}$;
- Call model on adversarial input:
- $\tilde{y}_2 \leftarrow$ BiLSTM(x_{adv});
- Compute adversarial loss:
- $L_{adv} \leftarrow Loss(\tilde{y}_2, y)$;
- Compute total loss: $L_T \leftarrow L_r + L_{adv}$;
- Extract model weights:
- $W \leftarrow$ BiLSTM.weights ;
- Compute the gradient of the total loss w.r.t the model weights:
- $g_{adv} \leftarrow$ Gradient(L_T,W);
- Update the model's weights using the computed gradient:
- BiLSTM.weights $\leftarrow$ BiLSTM.train(g_{adv}) ;

end

In addition, a schematic representation of the adversarial training algorithm is provided in Figure 4.

4 Results

We evaluate the performance of the adversarially trained BiLSTM (denoted Adv-BiLSTM) with regard to the task of classifying statements as being either factual or non-factual. To do this, Adv-BiLSTM is compared to a regular BiLSTM. It is also compared to an SVM, using the results from (Meng et al., 2020) that utilized the same dataset used for this thesis. Results from the same paper for a BIRT model (CB-BB) and an adversarially trained BIRT model (CB-BBA) are also shown for reference, as these are currently top-performing. We see the comparative performance metrics for the models in Table 1.

4.1 Dataset

This thesis uses the ClaimBuster dataset, which is a collection of 9674 sentences from presidential debates that were held between 1960 and 2012. This dataset is divided into 2 classes, namely non-check-worthy sentences (NCS) and check-worthy-factual sentences (CFS), where there are a total of 6910 sentences in the NCS class and 2764 sentences in the CFS class. The ClaimBuster dataset is a modified version of the dataset used in (Hassan et al., 2017). The main modification is that the original dataset had two classes besides the CFS class named UFS and NFS which were com-

	Precision		Recall		F1	
	NCS	CFS	NCS	CFS	NCS	CFS
SVM	0.8935	0.7972	0.9263	0.7240	0.9096	0.7588
CB-BB	0.9344	0.8149	0.9239	0.8379	0.9291	0.8263
CB-BBA	0.9335	0.8445	0.9386	0.8329	0.9361	0.8386
BiLSTM	0.9101	0.7975	0.9225	0.7450	0.9125	0.7710
Adv-BiLSTM (eps = 2)	0.8975	0.8013	0.9264	0.7351	0.9118	0.7666
Adv-BiLSTM (eps = 5)	0.8923	0.8296	0.9407	0.7135	0.9158	0.7676

Table 1: Results Table

bined to create the NCS class in this dataset. The UFS class contained factual but unimportant statements, and the NFS class contained non-factual statements.

To create the original dataset, transcripts were collected from all presidential debates occuring between 1960 and 2012. During this period (i.e. between 1960 and 2012), presidential debates were held consistently in all the election years except for 1964, 1968 and 1972, with variable counts of debates in each of the years, amounting to a total of 30 presidential debates. More information about the dataset can be obtained from (Arslan et al., 2020), and information about the specific version of the dataset used in this study, including the details of its differences from the original corpus, is available from (Meng et al., 2020).

4.2 Models

The models being evaluated are listed below:

SVM: To set a benchmark, we consider the results of the SVM obtained by (Meng et al., 2020) on the same dataset. This model was constructed using the LinearSVC class in Scikit-learn. Its feature vectors are a combination of tf-idf vectors, part-of-speech vectors, and sentence length. The results from this model are displayed alongside the results from other models.

CB-BB: This is one of ClaimBuster's recently implemented and top performing models. It is a BIRT model, and is included here both to serve as a reference and to place the performance of BiLSTM and Adv-BiLSTM in context with respect to the state-of-the-art. More information about this model (and its adversarially trained variant, below) can be obtained from (Meng et al., 2020).

CB-BBA: This is an adversarially trained replica of the above BIRT (that is, CB-BB) model.

BiLSTM: The Bidirectional LSTM (or, BiLSTM) model used for this study is a TensorFlow 2.0 LSTM implementation that takes word embedings as inputs. In this thesis, GloVe word embeddings of dimension 300 have been used as the input representations. The hidden dimension of BiLSTM is also 300, and between the LSTM and output layers there is a dropout layer with a dropout of 0.5. Binary cross-entropy is used as the loss function for the training model, and TensorFlow's RMSprop optimizer is used as the training algorithm for optimizing model weights.

Adv-BiLSTM: The adversarially trained BiLSTM is structurally identical to BiLSTM but differs in its training algorithm. The adversarial training method is described in full detail in Section 3.

4.3 Adversarial training study results

To evaluate the performance of Adv-BiLSTM relative to BiLSTM and SVM we consider their precision, recall and F1 scores. The first thing noticed from the experiments is that Adv-BiLSTM outperforms the baseline SVM model. Establishing this fact is important because, had it been otherwise, we would have strong evidence against using adversarial training for this task. In comparing BiLSTM to Adv-BiLSTM, however, it is less clear which one performs better as neither model scores higher on all the metrics. With regard to the differences between the models, we observe that the precision of the adversarially trained model on the CFS class is higher than the precision of BiLSTM. Also, the highest precision for this

class occurs when $\epsilon = 2$, with a precision 0.8296 for Adv-BiLSTM as compared to 0.7975 for BiLSTM. This is encouraging because it demonstrates that, on the statements classified as check-worthy, Adv-BiLSTM is more likely to be correct. Adv-BiLSTM (with $\epsilon = 2$) also has the highest recall with respect to the NCS class, where it has a recall of 0.9407 as compared to an NCS recall of 0.9225 for regular BiLSTM. This demonstrates that the adversarially trained model is capturing a larger share of non-check-worthy statements. One business case benefit of this, for example, is that Adv-BiLSTM can help fact-checkers save on the time and cost of needlessly checking statements which are in reality not checkworthy.

Although discussing the details of the BIRT models (CB-BB and CB-BBA) is beyond the scope of this thesis, it will be of benefit to point out an interesting parallel in the effect of adversarial training on the model performance of both BiLSTM and BIRT. With respect to model precision, the adversarially trained BIRT model (CB-BBA) has a higher precision than the regularly trained BIRT model (CB-BB) on the CFS class, but a lower precision on the NCS class. With respect to recall, CB-BBA has a higher recall than CB-BB on the NCS class, while an inverse relationship holds on the CFS class. As demonstrated in the previous paragraph, these same relationships are seen to hold in the comparative metrics of BiLSTM and Adv-BiLSTM.

4.4 Areas of further study

As it stands, Adv-BiLSTM does not clearly outperform BiLSTM across all metrics, which would have been the ideal outcome for a strong case to be made in its favor. It will therefore be interesting to explore what additional modifications can be made to improve its overall performance. One observation is that the hyperparameter, ϵ, has a noticeable effect on the model's performance. Therefore, one possible area of further inquiry is to do a more thorough study on the effects of varying this hyperparameter. Additionally, for this thesis, all characteristics of the base BiLSTM were held constant during adversarial training. The motivation behind doing so was to ensure that results are directly comparable. It is conceivable, however, that adversarial training may require modifications to the base model to be most effective. For example, it may perform better with a different loss function, or it may require that the model's dimensions be changed. These are all potentially interesting areas of further inquiry.

References

Fatma Arslan, Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2020. A benchmark dataset of check-worthy factual claims.

Yarin Gal and Zoubin Ghahramani. 1997. A theoretically grounded application of dropout in recurrent neural networks. *NIPS*.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *ICLR*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *NIPS*.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *IJCNN*.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Marke Tremayne. 2017. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. *CIKM*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Katarzyna Janocha and Wojciech Marian Czarnecki. 2017. On loss functions for deep neural networks in classification.

Damian Jimenez and Chengkai Li. 2018. An empirical study on identifying sentences with salient factual statements. *IJCNN*.

Georgi Karadzhov, Preslav Nakov, Lluis Marquez, Alberto Barron-Cedeno, and Ivan Koychev. 2017. Fully automated fact checking using external sources. *RANLP*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. *AAAI*.

Kevin Meng, Damian Jimenez, Fatma Arslan, Jacob Daniel Devasier, Daniel Obembe, and Chengkai Li. 2020. Gradient-based adversarial training on transformer networks for detecting check-worthy factual claims.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *ICLR*.

Jeffrey Pennington, Richard Socher, and Chistopher D. Manning. 2014. Glove: Global vectors for word representation. *ACL*.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation. *NAACL*.

Sebastian Ruder. 2017. An overview of gradient descent optimization algorithms.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE*.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE*.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xialin Li. 2017. Adversarial examples: Attacks and defenses for deep learning.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2018. Adversarial examples: Attacks and defenses for deep learning.