International Journal of Computer Applications (0975 - 8887)
Volume 178 - No.50, September 2019

Toward Mitigating Adversarial Texts

Basemah Alshemali
College of Computer Science and Engineering
Taibah University, KSA
College of Engineering and Applied Science
University of Colorado at Colorado Springs, USA

Jugal Kalita
College of Engineering and Applied Science
University of Colorado at Colorado Springs, USA

ABSTRACT

Neural networks are frequently used for text classification, but can be vulnerable to misclassification caused by adversarial examples: input produced by introducing small perturbations that cause the neural network to output an incorrect classification. Previous attempts to generate black-box adversarial texts have included variations of generating nonword misspellings, natural noise, synthetic noise, along with lexical substitutions. This paper proposes a defense against black-box adversarial attacks using a spell-checking system that utilizes frequency and contextual information for correction of nonword misspellings. The proposed defense is evaluated on the Yelp Reviews Polarity and the Yelp Reviews Full datasets using adversarial texts generated by a variety of recent attacks. After detecting and recovering the adversarial texts, the proposed defense increases the classification accuracy by an average of 26.56% on the Yelp Reviews Polarity dataset and 16.27% on the Yelp Reviews Full dataset. This approach further outperforms six of the publicly available, state-of-the-art spelling correction tools by at least 25.56% in terms of average correction accuracy.

Keywords

Adversarial text, Spelling correction.

1. INTRODUCTION

Deep neural networks (DNNs) have gained popularity in image classification [17], object recognition [28], and malware detection [26]. DNNs are also a popular option to solve many tasks in speech recognition [13], voice synthesis [33], and natural language processing [31].

Many use cases of DNNs are life crucial, raising significant safety and security concerns. Studies have shown that DNNs are vulnerable to carefully-designed input samples that can trick even high-performing models. Small perturbations to the inputs can mislead such networks into making wrong and potentially harmful decisions.

Szegedy et al. [32] first exposed this vulnerability of DNNs in the context of image classification, by showing that small perturbations to the inputs can cause deep learning classifiers to completely change their output on a given image. They also showed that the same perturbation can transfer, fooling a variety of network classifiers. This vulnerability has been shown in the computer vision [11] [25], speech [1] [5], and NLP domains [6].

Two kinds of attacks have been proposed in NLP: Black-box and white-box. A black-box attacker is only able to query the target model and manipulate its input samples by testing and observing the model's output [3]. In the white-box setting, the attacker who generates adversarial examples, possesses complete knowledge of the target model such as the model's parameters, along with the architecture, training method, input features, and, in some situations, the training data as well [20].

Researchers have used the fast gradient sign method (FGSM) [11] for NLP as a white-box attack [10], while others have designed targeted black-box attacks for text classifiers [8] [3], neural machine translators [37], neural dependency parsers [30], or morphological tagging [12].

This paper proposes a defense against black-box adversarial attacks on the text classification task. A spelling correction algorithm that utilizes frequency and contextual information is used to correct nonword misspellings. At testing time, the proposed defense detects the contaminated (adversarial) input texts and recovers them before they are classified by the model.

2. RELATED WORK

Google's Perspective API[1] uses deep learning models to classify texts and predict whether a message is toxic or not. Hosseini et al. [14] introduced errors into the Perspective API, by adding a dot or space between two letters of a word, by deliberately misspelling a word by repeating a letter or swapping two letters, and by adding "not" to negate phrases. They concluded that such adversarial examples greatly undermine the usefulness of DNN-based toxic comment detection systems.

Belinkov and Bisk [3] designed black-box attacks to investigate the sensitivity of neural machine translation to natural and synthetic noise containing common misspellings. Their natural attack collects naturally occurring errors from real-world corpora and inserts them into adversarial datasets. In addition, four types of synthetic attacks were used: swapping two adjacent letters (one swap per word), randomizing the order of all the letters in a word except for the first and the last, completely randomizing letters in words, and randomly replacing one letter in the word with an adjacent key. The authors experimented with the char2char [18], Nematus [29], and charCNN [36] models and used the TED talks (French, German, and Czech) to English parallel corpus prepared for IWSLT 2016 [22]. The French corpus BLEU score degraded from 42.5 to

[1] https://www.perspectiveapi.com/

7.5, while that for the German corpus went down from 33.0 to 4.0 and that of the Czech corpus went from 27.11 to 3.5.

Gao et al. [8] attacked sentiment classification models by designing a black-box adversary: DeepWordBug. They scored the importance of tokens in an input sequence using a classifier. A simple algorithm was used to transform high-scoring tokens with four transformation methods: swapping two contiguous letters in the word, substituting a letter for a random letter, deleting a letter at random, and inserting a random letter. The approach was tested on eight text datasets which included a variety of NLP tasks (text classification, sentiment analysis and spam detection) targeting a word-level LSTM model and a charCNN model [36]. Their approach reduced the model prediction accuracy by 64.38% on average for the Word-LSTM model and 30.35% on average for the charCNN model.

In 2014, DNNs were shown to be easily exploited by adversarial attacks [32]. Since then, this discovery has prompted increased research in adversarial defense. In text, an adversarial attack was designed to produce misspelled words [12] [8] [3]. It is natural to explore whether data processing is useful against such attacks. Recent studies have found that spelling correction is a relatively effective method to detect adversarial examples.

Gao et al. [8] used the Python autocorrector[2] to mitigate their attacks: Substitute two letters for random letters (Substitute-2) and delete two random letters (Delete-2). Under the Substitute-2 attack, the Python autocorrector increased the model's accuracy from 11.90% to 54.54%. Under Delete-2, the model's accuracy was increased from 14.25% to 33.67%. The prediction accuracy of their model under no attacks was 88.45%.

Belinkov and Bisk [3] used Google's spell-checker to correct the misspellings generated by natural noise, in which they had compiled naturally occurring errors from real-world corpora and inserted them into the corpus of the targeted model. The BLEU scores of their vanilla original corpora were 43.3, 38.7, and 26.5 for the French, German, and Czech corpora respectively.Under their natural attack, Google's spell-checker increased the French corpus' BLEU score from 16.7 (under attack) to 21.4 and the German corpus' BLEU score from 18.6 to 25.0. However, the Czech corpus' BLEU score decreased from 12.3 to 11.2.

Gao et al. and Belinkov and Bisk used spell-checkers that were freely available, but novel spell-checkers have been proposed by recent papers. Fivez et al. [7] proposed a context-sensitive spelling correction method for clinical text in English. They collected their replacement candidates from a reference lexicon based on both graphical and phonological distance. Within a specified window of text, their method vectorized each word on each side, applied reciprocal weighting, and summed the vectors. Then, they computed the cosine similarity with each candidate vector and divided the result by the edit distance between the misspelling and the candidate. The authors trained their vectors with the fastText model [4] and evaluated their method on the MIMIC-III [15] corpus, resulting in an average correction accuracy of 78.48%.

Lu et al. [21] developed CSpell to correct spelling errors in consumer health questions. They ranked their candidates by orthographic similarity scores first, then by context scores, followed by word frequency scores in a sequential order. They calculated the orthographic similarity score by using a weighted sum of edit distances, phonetic similarity scores and character overlap similarity scores. A word's frequency score is calculated as the word's frequency in the corpus divided by the number of occurrences of the most frequent words in this corpus. The context score is computed by multiplying the context vector by both input and output matrices of the Word2vec continuous bag-of-words (CBOW) model [24]. They tested their method on a test set collected from consumer health questions submitted to the National Library of Medicine. On nonword correction, their model scored 0.7644 F1-score.

[2]https://github.com/phatpiglet/autocorrect/

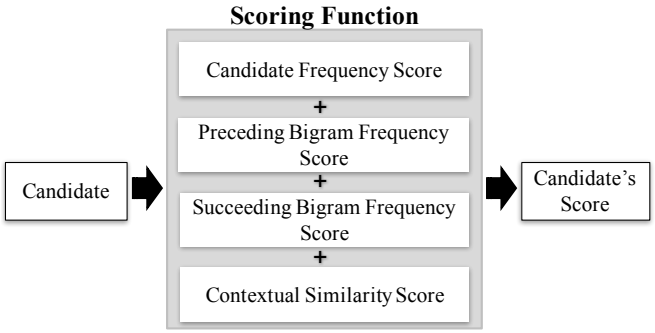

Fig. 1. Scoring the candidates based on four scores: 1- Frequency score; 2- Preceding bigram frequency score; 3- Succeeding bigram frequency score and 4- Contextual similarity score.

3. METHODOLOGY

In order to defend a text classifier against black-box adversarial attacks such as Gao et al.'s, it is recommended to remove noise from the testing set before feeding test data to the classifier. At testing time, words that hold the noise must be detected, and the noise must be removed. To achieve this goal, the misspelled words must be corrected.

The spell-checkers proposed by Fivez et al. [7] and Lu et al. [21] are in biomedical domains. The implementation code of Fivez et al.'s is not publicly available. The analysis conducted in this paper has shown that Lu et al.'s method (CSpell) performs poorly on misspellings created by the Delete-2 and Substitute-2 attacks (see Section 6). Gao et al. and Belinkov and Bisk used the Python autocorrector and Google's spell-checker to mitigate attacks. Their results showed that these popular auto-correctors were not sufficient to correct misspellings. This highlights the need for an alternative method that can mitigate more difficult attacks, like Delete-2 and Substitute-2, and can be applied on all domains. The proposed approach is motivated by this need.

This paper proposes a method that combines edit distance, frequency counts, and contextual similarity techniques for correcting misspellings. Before detecting and correcting misspellings, the text is filtered for numbers, special symbols, dates, email addresses and hyperlinks. It is worth mentioning that these are reinserted into the text after the spellings are corrected.

After filtering the dataset, the nonword misspellings are detected using our corpus as a dictionary. The detection policy is quite simple, a word in a text is potentially a nonword misspelling if it does not exist in the dictionary. The proposed method corrects nonword misspellings by the following procedure:

(1) **Generation of a candidate pool:** Candidate suggestions for each detected misspelling are generated by returning all words from our dictionary that have an edit distance [19] up to a given threshold.

(2) **Scoring Function:** Given a misspelled token in a text and a set of candidate corrections for that token, the scoring function ranks all suggested candidates based on the following four scores (see Figure 1):

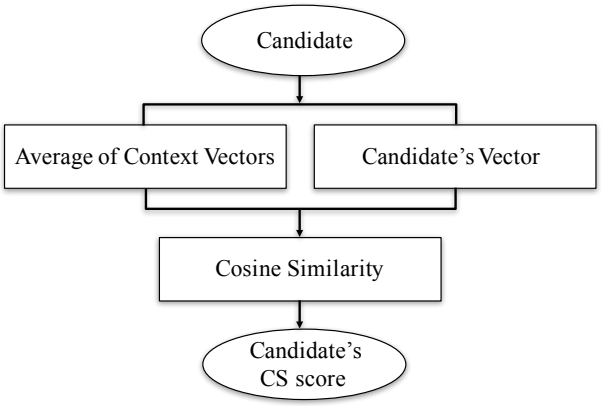

Fig. 2. Calculating the contextual similarity (CS) score of a candidate: Compute the cosine similarity between the candidate's vector and the average of the context vectors.

—Candidate Frequency Score (CF): a count of occurrence of the candidate in the corpus [16].
—Preceding Bigram Frequency Score (PB): a count of how many times the candidate and the token immediately before occur in the corpus.
—Succeeding Bigram Frequency Score (SB): a count of how many times the candidate and one token after occur in the corpus.
—Contextual Similarity Score (CS): Fivez et al. [7] calculated contextual similarity scores using neural word embeddings, taking the context around the misspelling into account. To calculate the contextual similarity score for a candidate, this paper uses a similar approach. The cosine similarity between the context vector and the candidate vector is computed. The context vector is the average of the vectors associated with the words around the misspelling. Figure 2 demonstrates the procedure for calculating the contextual similarity score of a candidate. If the cosine similarity is less than zero or if there is no vector corresponding to the candidate, the contextual similarity is set to zero. If a word does not have a corresponding vector, the processing moves to the next word in the same direction.

(3) **Winner candidate:** After computing and normalizing the four scores, the final score of the candidate is their sum:

$$final\_score = CF + PB + SB + CS. \quad (1)$$

where CF is the Candidate Frequency Score, PB is the Preceding Bigram Frequency Score, SB is the Succeeding Bigram Frequency Score, and CS is the Contextual Similarity Score. The candidate with the highest score is chosen as the correct spelling.

Fivez et al. used the cosine similarity with word vectors between context and candidates to calculate context scores. However, they ignored the importance of n-gram frequencies. Lu et al. computed the context score differently by multiplying the context vector by both the input and the output matrices of the Word2vec CBOW model. Although they utilized the unigram frequency in their proposed model, they ignored the bigram frequency.
The approach used in this paper considers frequency information (word unigrams), the order of the words (word bigrams), and the flexibility of the context (the contextual information) to outperform these two latest spell-checkers as well as others and also mitigate adversarial attacks.

Table 1. Datasets details.

Statistics	**Yelp Polarity**	**Yelp Full**
Total number of training samples	560,000	650,000
Total number of testing samples	38,000	50,000
Average sample length in words	135.9	137.5
Average token length in letters	4.08	4.09

4. EXPERIMENTS

The proposed method is implemented using Python [34] with Editdistance 0.5.2[3], Pandas [23], Numpy [35], and PyTorch 0.4.0 [27] libraries.

4.1 Corpora

Two large-scale text datasets from Zhang et al. [36] are used in the experiments : Details of the datasets are listed in Table 1.

(1) *Yelp Reviews Polarity dataset*: This dataset was built from the 2015 Yelp Dataset Challenge. It is constructed by considering reviews with 1 and 2 stars as negative, while reviews with 3 or more stars are positive. It has 280,000 training samples and 19,000 testing samples for each polarity.

(2) *Yelp Reviews Full dataset*: This dataset was also built from the 2015 Yelp Dataset Challenge. It is constructed by randomly taking 130,000 training samples and 10,000 testing samples for each review star from 1 to 5.

4.2 Text Classification Model

To evaluate the proposed approach, several experiments on the Word-level LSTM model of Gao et al. [8] were conducted . This model is a Bi-directional LSTM which contains an LSTM in both directions, reading from first word to last and from last word to first.

4.3 Word Embeddings

The Word2vec with Continuous Bag-of-Words (CBOW) model [24] trained on this paper's corpora with ten epochs and a window size of two was used to generate word vectors of 300 dimensions.

4.4 Dictionary

Since the used corpora were collected from Yelp reviews, which are drafted by the general public, the authors of this paper first filtered them for numbers, special symbols, dates, email addresses and hyperlinks. Next, they ran the TextBlob[4] autocorrector on them to correct potential misspellings before utilizing the data for the dictionary. In addition, they added English contractions to the dictionary so that tokens with English contractions such as "would've" will not be detected as a misspelling. They obtained a list of all English contractions from Wikipedia[5].

[3] `https://pypi.org/project/editdistance`
[4] `https://textblob.readthedocs.io/en/dev/index.html`
[5] `https://en.wikipedia.org/wiki/Wikipedia:List_of_English_contractions`

Table 2. A comparison between the accuracy of the original testing set with no spelling corrections and the accuracy after spelling corrections are applied by the defense.

Dataset	Original Data Acc	Corrected Original Data Acc
Yelp-Polarity	0.93750	0.93750
Yelp-Full	0.60938	0.60156

4.5 Context Vectors

As mentioned in Section 3, the contextual similarity scores are calculated by computing the cosine similarity between the context vector and the candidate vector. The context vector is the average of the vectors associated with the words around the misspelling. In the experiments, the context words are determined to be the four words before the misspelling and the four words after it.

4.6 Adversarial Attacks

The adversarial texts used in the experiments are generated using the DeepWordBug method proposed by Gao et al. [8]. Two attacks introduced by Gao et al. are used: (1) Insert Transformer: Insert one random character to the word; and (2) Flip Transformer: Substitute one character for a random character.
Besides, two stronger attacks of Gao et al's are utilized: (1) Flip-2 Transformer: Substitute two characters for two random characters; and (2) Remove-2 Transformer: Delete two characters from the word.
In addition to Gao et al.'s attacks, more adversarial samples are created using the attacks proposed by Belinkov and Bisk [3]. Two kinds of attacks are used: (1) Natural noise where naturally occurring misspellings were harvested from the EF-Cambridge Open English Learner Database corpus [9] and inserted into the testing corpus by replacing some words with a frequently encountered misspelled version; and (2) Swap two adjacent characters in the word (one swap per word). Since Belinkov and Bisk's attacks are designed for neural machine translation and not for classification models, Gao's combined scoring function is first utilized to determine the important tokens for the prediction and then Belinkov and Bisk's attacks are applied to replace the important tokens with misspelled versions.
To correct the misspellings generated by all transformers, the edit distance is sat to be up to two. All the adversarial texts are generated using Gao et al's BiLSTM model and their combined scoring function.

4.7 Performance Evaluation

Classification accuracy is used as the metric to evaluate the performance of the proposed defensive model. Higher accuracy denotes a more effective model.

5. RESULTS

Following the example of previous studies (e.g., Alzantot et al. [2]) that have worked on adversarial examples, the proposed approach is tested on a subset (the first 1280 samples) of the Yelp Reviews Polarity and the Yelp Reviews Full datasets. First, the defense is evaluated on clean data with no adversarial attacks. Table 2 shows a comparison between the accuracy of the model with the original testing set with no defense (no spelling corrections applied) and the accuracy of the model with the original testing set after spelling corrections were applied by the defense. On the Yelp Reviews Polarity dataset, the defense does not affect the accuracy of the clean data. On the other hand, the accuracy of the Yelp Reviews Full dataset has a 00.78% reduction which is negligible.
The attacks of Gao et al. and Belinkov and Bisk are then used to convert the same subsets, the first 1280 samples of the two datasets, to adversarial examples. As a result, the model's classification accuracy fell dramatically. The accuracy of the Yelp Reviews Polarity dataset went from an average of 93.75% to 62.10%, and the accuracy of the Yelp Reviews Full dataset went from an average of 60.93% to 40.23%.
The proposed defense has been experimented with four types of Gao et al's attacks and two types of Belinkov and Bisk's attacks, setting the maximum attack power (number of modified characters in the input sequence) to be 30 and having one isolated experiment for each attack. The effectiveness of the proposed defense is evaluated under these attacks and the results are summarized in Table 3.

5.1 The Effectiveness of the Defense

Table 3 shows that the proposed defense recovered most of the adversarial examples generated by the Insert attack. Although Insert transformer has the largest degradation in the adversarial data accuracy in both datasets, the defensive model was effectively able to counter it, recover most of its adversarial examples, and improve the accuracy by 35.15% on the Yelp Reviews Polarity dataset, and 22.65% on Yelp Reviews Full dataset.
On the Yelp Reviews Polarity dataset, the attack that flipped one character to a random character (Flip) and the attack that flipped two characters to two random characters (Flip-2) had almost identical effects on the classification accuracy of the model. In both attacks, the defense recovered the attacked data and improved the accuracy equally to 88.28%. Similarly, on the Yelp Reviews Full dataset and under Flip and Flip-2, the defense increased the accuracy of the model by an average of 17.96%.
Remove-2 attack (removes two characters from the word) had the least perturbation effect on the classification accuracy. The adversarial data scored the highest accuracy when the transformer was Remove-2. A straightforward explanation for this is that the average number of token characters in both datasets is 4, and hence to generate an adversarial sequence that is visually or morphologically similar to its original sequence, removing two characters from tokens was restricted. The number of perturbed tokens generated by Remove-2 attack is much smaller than the number of perturbed tokens generated by the other attacks. Among the four attacks, Remove-2 is the hardest attack to defend against. Under the Remove-2 attack, the proposed defense improved the classification accuracy by an average of 10.94%.
The natural attack that replaces some words with misspelled versions scored slightly higher in terms of adversarial accuracy. This attack pulls natural misspellings from a parallel corpus and then replaces the important tokens in the adversarial corpus with misspelled versions. Important tokens with no natural misspelled versions are kept as they are. As a result, the number of adversarial tokens inserted by the natural attack is lower than the number of perturbed tokens generated by the other attacks (except for Remove-2). This explains the increase of the undefended base model's accuracy

Table 3. Effectiveness of the defense technique on the Yelp Reviews Polarity Dataset (top) and the Yelp Reviews Full Dataset (bottom). Acc is the classification accuracy of the LSTM model. The defensive model corrected the adversarial data and the corrected version is given in the Corrected Data column. Percent Increase is the percent increase of the accuracy after recovering the data. The accuracy of the original data was 0.9375 for all cases on the top table, and 0.6093 for all cases in the bottom table. All results are under maximum attack power i.e. the number of modified characters in the input sequence: 30.

Attack	Adversarial Data Acc	Corrected Data Acc	Percent Increase
Yelp Reviews Polarity Dataset			
Insert	0.5625	0.9140	0.3515
Flip	0.5703	0.8828	0.3125
Flip-2	0.5781	0.8828	0.3047
Remove-2	0.7421	0.8593	0.1172
Natural	0.6562	0.8671	0.2109
Swap	0.6171	0.9140	0.2968
Average	0.6210	0.8866	0.2656
Yelp Reviews Full Dataset			
Insert	0.3750	0.6015	0.2265
Flip	0.3750	0.5468	0.1718
Flip-2	0.3671	0.5546	0.1875
Remove-2	0.4921	0.5937	0.1016
Natural	0.4062	0.5390	0.1328
Swap	0.3984	0.5546	0.1562
Average	0.4023	0.5650	0.1627

when tested under the Natural attack compared to the other attacks. Under this attack, the defense increased the model's accuracy by 21.09% and 13.28% for the Yelp Reviews Polarity dataset and the Yelp Reviews Full dataset, respectively.

According to Belinkov and Bisk, the Swap attack switches two adjacent characters in the word with one constraint: the length of the word has to be greater than or equal to 4. This constrains the number of adversarial tokens generated by the Swap attack, causing the accuracy of the base undefended model under the Swap attack to be higher than the accuracies found when the base model is subjected to the Insert, Flip, and Flip-2 attacks. Under the Swap attack, the defense increased the model's accuracy for the Yelp Reviews Polarity dataset and the Yelp Reviews Full dataset by 29.69% and 15.62%, respectively.

5.2 Convolutional Neural Networks

To show that the defense works on varied types of models apart from the word-level BiLSTM model, several experiments have been conducted with the character-level Convolutional Neural Network of Zhang et al. [36]. The character-level CNN of Zhang et al.'s used one-hot encoded characters as inputs to a nine-layer network with six convolutional layers and three fully-connected layers. The performance of the character-level CNN is evaluated under the Flip and Flip-2 attacks. The results are briefly summarized in Table 4. The accuracy of the CNN model after deploying the defense is also shown in Table 4. It is noted that the defense is effective in recovering the adversarial samples and in mitigating their effect on the CNN model's performance.

It is also noted that the character-level CNN is more resistant to adversarial samples (misspellings) than the word-level BiLSTM. Under the same attacks, Flip and Flip-2, the average performance of the word-level BiLSTM decreased by 30.07% while that of the character-level CNN reduced by only 10.94%. The models that are based on word-embeddings seem to be vulnerable to misspellings created by introducing unknown words.

This supports the conclusion from previous research that deep character-level CNNs trained on large scale datasets do not require knowledge about the syntactic or semantic structure of words. Instead, they can naturally learn and identify abnormal combinations of characters, including misspellings [36].

Table 4. Effectiveness of the defensive character-level CNN. Acc is the classification accuracy of the CNN model. The defensive model corrected the adversarial data and the corrected version is given in the Corrected Data column. The accuracy of the original data was 0.9453 for Yelp Reviews Polarity Dataset, and 0.5859 for Yelp Reviews Full Dataset. All results are under maximum attack power 30.

Attack	Adversarial Data Acc	Corrected Data Acc
Yelp Reviews Polarity Dataset		
Flip	0.8359	0.9453
Flip-2	0.8046	0.9375
Yelp Reviews Full Dataset		
Flip	0.5312	0.5625
Flip-2	0.4531	0.5156

6. AUTOCORRECTION

Two natural questions now arise. Are spell-checkers sufficient for detecting and recovering adversarial data, and is a new spell-checker necessary? To test, six tools are used to correct the same misspellings produced by the Flip-2 and Remove-2 attacks that used in Table 3. These tools are listed below:

(1) Google's spell-checker: The first suggestion for every detected misspelling is accepted .

(2) CSpell: Open-source distributable stand-alone spelling correction tool [21].

(3) TextBlob: An implementation of Peter Norvig's spelling corrector [6].

[6] http://norvig.com/spell-correct.html

Table 5. The average increase in the classification accuracy of the LSTM model after recovering the adversarial data of Flip-2 and Remove-2 using the defense and the six spelling correctors.

Method	Yelp Polarity	Yelp Full
Aspell	04.02%	01.94%
Python Autocorrector	05.59%	02.30%
Hunspell	07.54%	03.37%
CSpell	08.90%	03.51%
TextBlob	10.15%	04.39%
Google's spell-checker	10.97%	06.25%
Proposed Method	**21.09**%	**13.70**%

Table 6. The average correction accuracy for misspellings created by the hardest two attacks, Flip-2 and Remove-2, using the defense and the six spelling correctors.

Method	Yelp Polarity	Yelp Full
Aspell	10.15%	06.59%
Python Autocorrector	12.27%	09.53%
Hunspell	15.43%	14.92%
CSpell	17.94%	15.50%
TextBlob	21.24%	18.98%
Google's spell-checker	26.34%	25.00%
Proposed Method	**51.80**%	**50.66**%

(4) Hunspell[7]: A popular open source spell checker used by OpenOffice.org, Mozilla Firefox, Thunderbird, and Google Chrome.

(5) Aspell[8]: Another popular open source spell checker, which can be used as either an independent spell checker or as a library.

(6) Python Autocorrector.

The results of using these six spelling correctors to recover adversarial texts are reported in Table 5. By comparing the results of using the defense with the results of using the six spelling correctors (Table 5), the authors claim that their defense is more effective than Google's spell-checker, TextBlob, CSpell, Hunspell, Python Autocorrector and Aspell.
TextBlob is a probabilistic tool that finds the candidate suggestion with the highest probability. CSpell uses phonetic similarity, character overlap, contextual information and unigram frequency to rank its candidates. Hunspell uses n-gram similarity, morphological analysis, and stemming to improve its suggestions. These tools use frequency information to propose suggestions, making them more useful than the other tools.
Python Autocorrector and Aspell depend on dictionaries to find replacements. The lack of frequency information when making suggestions leads to lower performance than the correctors that use probabilities to determine suggestions.

6.1 Correction Accuracy

After correcting the misspellings generated by the Flip-2 and Remove-2 attacks that used in Table 3 and Table 5, the number of misspelled tokens that were amended correctly were calculated. Table 6 compares the performance of the defense to the performance of each of the six spelling correctors: Google's spell-checker, TextBlob, CSpell, Hunspell, Python Autocorrector and Aspell.
The results indicate that the proposed approach outperforms the six spelling correction tools and improves the correction accuracy by an average of at least 25.56% compared to Google's spell-checker, the best tool among the six spelling correction tools. Thus, the proposed spell-checker is better than all the others at its intended immediate task as well.

7. CONCLUSION

This paper proposes a defense against black-box adversarial attacks based on spelling corrections. Utilizing word frequency and contextual information made a powerful technique to correct nonword misspellings. After testing the defense on adversarial samples, it successfully improves the model's classification accuracy from an average of 62.10% to an average of 88.66% on the Yelp Reviews Polarity dataset, and from an average of 40.23% to an average of 56.50% on the Yelp Reviews Full dataset. The results also show that the proposed approach outperforms six of sate-of-the-art and popular spelling checkers, Google's spell-checker, TextBlob, CSpell, Hunspell, Python Autocorrector and Aspell, and improves the correction accuracy by at least 25.56%. Future work will involve ensembling the defense with adversarial training. In addition, it is worthwhile to investigate how the defense model handles real-word misspellings.

[7] `http://hunspell.github.io/`
[8] `http://aspell.net/`

8. REFERENCES

[1] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. Did you hear that? adversarial examples against automatic speech recognition. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

[2] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, 2018.

[3] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*, 2018.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[5] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Proceedings of the 31st Conference on Neural Information Processing Systems*.

[6] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for NLP. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 31–36, 2017.

[7] Pieter Fivez, Simon Šuster, and Walter Daelemans. Unsupervised context-sensitive spelling correction of English and Dutch clinical free-text with word and character n-gram embeddings. *Biomedical Natural Language Processing Workshop*, pages 143–148, 2017.

[8] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade

deep learning classifiers. In *IEEE Security and Privacy Workshops*, pages 50–56, 2018.

[9] Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. Automatic linguistic annotation of large scale l2 databases: The EF-Cambridge open language database (efcamdat). In *Proceedings of the Second Language Research Forum*, pages 240–254, 2013.

[10] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[12] Georg Heigold, Günter Neumann, and Josef van Genabith. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the Conference of the Association for Machine Translation in the Americas*, pages 68–80, 2018.

[13] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[14] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving Google's perspective API built for detecting toxic comments. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2017.

[15] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[16] Dan Jurafsky and James H Martin. *Speech and Language Processing*. Pearson London, 3rd edition, 2018.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[18] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.

[19] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.

[20] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017.

[21] Chris J Lu, Alan R Aronson, Sonya E Shooshan, and Dina Demner-Fushman. Spell checker for consumer language (CSpell). *Journal of the American Medical Informatics Association*, 26(3):211–218, 2019.

[22] Cettolo Mauro, Girardi Christian, and Federico Marcello. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation*, pages 261–268, 2012.

[23] Wes McKinney et al. Data structures for statistical computing in Python. In *Proceedings of the Python in Science Conference*, volume 445, pages 51–56, 2010.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[25] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium, Security and Privacy*.

[26] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *IEEE European Symposium on Security and Privacy*, 2018.

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *the Conference on Neural Information Processing Systems*, 2017.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[29] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, 2017.

[30] Anders Søgaard, Miryam de Lhoneux, and Isabelle Augenstein. Nightmare at test time: How punctuation prevents parsers from generalizing. In *Proceedings of the 2018 Empirical Methods in Natural Language Processing*.

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[33] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *the ISCA Speech Synthesis Workshop*, page 125, 2016.

[34] G van Rossum. Python tutorial, technical report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam. 1995.

[35] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[36] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[37] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *International Conference on Learning Representations*, 2018.