Context-Sensitive Malicious Spelling Error Correction

Hongyu Gong, Yuchen Li, Suma Bhat, Pramod Viswanath
University of Illinois at Urbana-Champaign
{hgong6,li215,spbhat2,pramodv}@illinois.edu

ABSTRACT

Misspelled words of the malicious kind work by changing specific keywords and are intended to thwart existing automated applications for cyber-environment control such as harassing content detection on the Internet and email spam detection. In this paper, we focus on malicious spelling correction, which requires an approach that relies on the context and the surface forms of targeted keywords. In the context of two applications–profanity detection and email spam detection–we show that malicious misspellings seriously degrade their performance. We then propose a context-sensitive approach for malicious spelling correction using word embeddings and demonstrate its superior performance compared to state-of-the-art spell checkers.

CCS CONCEPTS

• **Computing methodologies** → *Lexical semantics*; *Unsupervised learning*.

KEYWORDS

Malicious spelling correction; cyberbullying; machine learning

ACM Reference Format:
Hongyu Gong, Yuchen Li, Suma Bhat, Pramod Viswanath. 2019. Context-Sensitive Malicious Spelling Error Correction. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313431

1 INTRODUCTION

Automatic spelling correction has been an important natural language processing component of any input text interface of today [17]. While spelling errors in common writing could be regarded as innocuous omissions, affecting opinions about the writer's competence at worst, those of the malicious kind can potentially be highly offensive, since they are intended to deceive an automated mechanism–be it a spam filter for emails or a profanity detector for social media. For those applications that rely on automatic detection of specific keywords to enable real-time control of the cyber-environment, detecting and correcting spelling errors is of primary importance.

Perspective [11], is one such application that helps reduce abusive content online by detecting profane language. It has been pointed out that Perspective's otherwise good performance in detecting toxic comments is not robust to spelling errors [15]. Consider the sentence, "My thoughts are that people should stop being stupid." which was assigned a perceived toxicity score of 86% by Perspective. When the word "stupid" was misspelled as "stup*d" in the same sentence, its toxicity score reduces to 8%. This marked reduction induced by the deceptive spelling error on the keyword 'stupid' reflects the importance of accurate spelling correction for optimal toxicity detection. Had the spelling error been detected and corrected *before* scoring for toxicity, the score would not have lowered.

Likewise, deliberate typographic errors are common in phishing scams and spam emails. For instance, "Please pya any fees you may owe to our company" where *pya* is clearly a spelling error, included to deceive spam filters that are sensitive to keywords. In both these instances, spelling correction as a preprocessing step of the messages are critical for a robust performance of the target system. Spelling correction in the preprocessing stage in malicious settings constitutes the focus of this study.

In this paper, we empirically demonstrate the effect of spelling errors in a malicious setting by adding synthetic misspellings to sensitive words in the context of two applications–profanity detection and spam detection. We propose an *unsupervised* embedding-based algorithm to correct the targeted misspelled words. Earlier approaches to spelling correction primarily depend on edit distances to find words morphologically similar to corrections. More recently, improved spell checkers with the addition of contextual information (e.g., n-gram modeling in [37]), often with intensive computation and memory requirements (to obtain and store N-gram statistics). A recently studied neural network-based spell checker learns the misspelling pattern from annotated train data in a supervised way, and was found to be sensitive to data domains and dependent on human annotations [9]. Our approach to make the correction procedure context-aware involves harnessing the geometric properties of word embeddings. It is light-weight in comparison, unsupervised, and can be adapted to a variety of data domains including Twitter data and spam data, since domain information can be well captured by tuning embeddings on domain-specific data [33].

We first demonstrate the effect of spelling errors in a malicious setting in two applications–profanity detection and spam detection. Then we propose an unsupervised algorithm to correct the targeted spelling errors. Besides performing well on the correction of synthetic spelling errors, the algorithm also performs well on spell checking of real data, and shows that it can improve hate speech detection using Twitter data.

2 RELATED WORK

Spelling correction. Automatic spelling correction in non-malicious settings has had a long research track as an important component of text processing and normalization. Early works have used edit

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313431

distance to find morphologically similar corrections [29], noisy channel model for misspellings [17], and iterative search to improve corrections of distant spelling errors [12]. Word contexts have been shown to be improve the robustness of spell checkers with n-gram language model as one approach to incorporate contextual information [8, 14]. Other ways of incorporating contextual information include n-gram statistics capturing the cohesiveness of a candidate word with the given context [37].

Effect of malicious misspellings. Misspellings are commonly seen in online social platforms such as Twitter and Facebook, and recent studies have drawn attention to the fact that online users deliberately introduce spelling errors to thwart bullying detection [27] or to challenge to moderators of online communities [26]. This is because simple ways of filtering terms of profanity are rendered inadequate in the presence of spelling errors. Likewise, email spam filters are often obfuscated by misspelled sensitive words [38] because word-based spam filters tend to make false decisions in the presence of misspellings [32], and so are word-based cyberbullying detection systems [3]. Misspelling is also seen in web attacks where a phishing site has a domain name as a misspelling of a legitimate website [16].

Approaches to correct malicious misspellings. Given the malicious intent of misspellings in the context of the Internet, recent studies have proposed correction strategies specifically for malicious misspellings. Levenshtein edit distance is commonly used in spell checking of detection systems. Spam filters [38] and cyberbullying detection systems [36] rely on the idea of edit distance to recognize camouflaged words by capturing the pattern of misspellings. For example, [30] studied an edit distance function designed to reveal instances where spammers had interspersed black-listed words with non-alphabetic symbols. Lexical and context statistics, such as word frequency, have also been used to make corrections in social texts [5, 18]. Existing approaches have the problem of domain specificity, since their lexical statistics are obtained from a certain domain which might differ from the domain of target applications.

Our study considers the spelling correction in a malicious setting where errors are not random, but are carefully introduced. Our context-aware approach to spelling correction relies on the geometry of word embeddings, which has the advantage of efficient domain adaptation. As we will show in Section 4.2, the embedding can be easily tuned on a small corpus from the target domain to capture domain knowledge.

3 METHODS

Spelling errors include non-word errors and real-word errors [28]. Non-word misspelling is not a valid word in standard dictionaries, while real-word misspelling is. For example, *piecce* is a non-word error of *piece*, and *peace* is its real-word misspelling. In this work, we focus on non-word misspellings. We study malicious spelling correction for three target applications–toxicity detection using the Perspective API, email spam detection, and hate speech detection on Twitter data. We work with synthetic spelling errors in the first two applications, and study the real misspellings in the third.

For the toxicity detection task, we use the Perspective dataset [11], which provides a set of comments collected from the Internet with human annotated scores of toxicity. Using the Perspective API we obtained the toxicity score for 2767 comments in the dataset.

The spam dataset consists of 960 emails from the Ling-Spam dataset [1]. We randomly split the data into 700 train emails and 260 test emails. Both the train and test sets were balanced with respect to the positive and negative classes. The most frequent 2500 words in the training data were selected, and we counted the occurrence of each word in the emails. The number of occurrences of these frequent words were used as a 2500-dimension feature vector for each email. We first used these features to train a Naive Bayes classifier. It achieved an accuracy of 98% in spam detection.

As for the Twitter data, a total of 16K user posts were collected from Twitter, a social communication platform and used in used in [34]. Of these tweets, 1937 were labeled as being racist, 3117 were labeled as being sexist, and the rest were neither racist nor sexist. The task of hate speech detection is to classify these tweets into one of the three categories *racist, sexist, neither*. We randomly split the tweets into train and test data, and trained a neural-network based hate speech detection system on the training data (described later in Section 5.3).

3.1 Characterization

Here we summarize our assumptions on the characteristics of the malicious spelling errors.

(1) Malicious errors are usually made on the sensitive keywords in order to obfuscate the true intentions and deceive detection systems that rely on keywords [35].

(2) The error yields a word that is similar to the original word in surface form. The misspelled words would be words that humans can easily understand (thus the erroneous words still convey the intended meaning, while being in disguise). Their similarity is reflected in the small edit distance between the erroneous and the correct word [24]. The errors often involve character-level operations as shown in Table 1 [21].

(3) With edit distance as the similarity criterion, it is often the case that the word with an error is similar to multiple valid words, making correction a challenge in real applications. For example, both *stud* and *stupid* can be thought of as the correct form of the erroneous word *stupd*. In such cases, spelling correction relies on the context in which the word occurs.

3.2 Effect of malicious spelling errors

We quantify the effect of spelling errors by showing that a simple mechanism of injecting malicious spelling errors greatly degrades the performance of toxicity detection and spam detection. Toward this, we first describe the general mechanism to generate synthetic errors and then study their impact on toxicity and spam detection. We point out that owing to the absence of a dataset with intended malicious errors, we had to generate them synthetically.

Spelling error generation. We first choose the "sensitive" words that the detection algorithm relies (the mechanism will be discussed separately for each application we consider). We then replace these words with their erroneous forms (which may not be real words). Given the characteristics of malicious errors mentioned in Section 3.1, our assumption is that as a result of these errors the altered

[1] http://csmining.org/index.php/ling-spam-datasets.html

words will appear similar to the original ones. As illustrated in Table 1, we consider four basic character-level operations to change the sensitive words: insertion, permutation, replacement, and removal. In our experiments we perform these operations on randomly picked characters of the sensitive word. For permutation, we exchange this character with the next one. Each operation in Table 1 increases the edit distance by 1, and we generate malicious misspellings of the sensitive word that are at most 2 edit distance from the correct word.

Table 1: Common Spelling Errors

Type	insertion	permutation	replacement	removal
Error	idio.t	moeny	chanse	stupd
Correction	idiot	money	chance	stupid

Finally, we obtain revised sentences by replacing the sensitive words in the original sentences with their erroneous counterparts. Next, we introduce our mechanism for selecting the "sensitive" words for each application.

Perspective toxicity detection. We select 2767 clean comments from a total of 2969 comments in the Perspective data, with the selection criteria given below.

- The most toxic word in a comment should contain more than two characters. A word that is too short is not likely to be a meaningful toxic word, and can have too many candidate corrections in the dictionary.
- The most toxic word in a comment should appear at least 100 times in the dataset, since rare words tend to be misspellings.
- The most toxic word in a comment should be a content word, i.e., it should not belong in the list of function words such as "must", "ought", "shall" and"should". Function words are not toxic and so were excluded from our experiments.

For each comment, its predicted toxicity score lies in the range of $[0, 1]$. The higher the score, the more toxic the sentence is. For each sentence, we chose the most toxic word to be that word, without which the toxicity score reduced the most. We then added malicious errors, as described above, to this word. We note that for 2380 toxic comments with toxicity scores higher than 0.5, spelling errors brought down their predicted toxicity by 32%. Section 5 provides the details of the degradation in toxicity detection.

Spam detection. A Naive Bayes spam detection model provides the likelihood of each word given the spam and non-spam classes. For a given word, the difference between these probabilities reflects how important that word is in spam detection. We sorted all the words in decreasing order of this difference in probabilities, and picked the most spam-indicative words. We then added malicious errors to these words in test spam emails. For a spam filtering system which relies heavily on the counts of spam-indicative words, the errors can easily disguise spams as non-spams. This is seen in the test accuracy dropping from 98% on the original test data to 72% on the revised test data.

We note that for the Perspective data, we added errors to only one word per sentence, but did not have such limits for the spam data. Also, we do not limit the number of corrections during the spell check process.

4 SPELLING CORRECTION

We have shown that malicious errors degrade the performance of toxicity and spam detection. We next introduce our unsupervised algorithm on non-word error correction based on the relevant context. Our correction method proceeds in two stages. In the first stage of **candidate enumeration**, the algorithm detects the spelling error and proposes candidates that are valid words and similar in surface form to the misspelled word. In the next stage, **correction via context**, the best candidate is chosen based on its context.

4.1 Candidate Enumeration

As in the case of non-word spelling correction [17], we check whether an erroneous word is valid or not using a vocabulary of valid words. For toxicity detection and spam detection, the vocabulary consists of words that occur more than 100 times in English Wikipedia corpus [2] For hate speech detection, the vocabulary consists of standard words from Wikipedia and the list of internet slang words as detailed in Section 5.3, since slang words frequently occur in tweets. For an invalid word, we find candidate words with the smallest Damerau-Levenshtein edit distance [7, 20]. This distance between two strings is defined as the minimum number of unit operations required to transform a string into another, where the unit operations include character addition, deletion, transposition and replacement. We note that because there are potentially multiple candidates having the smallest distance to the misspelled word, the output of this stage is a set of candidate corrected words.

4.2 Correction via Context

After enumerating all possible candidates, we choose the one that fits the best in the context. We propose a word-embedding-based method to match the context with the candidate.

Pretrained embedding. We use the word2vec CBOW model to train word embeddings on the training data of the three applications [22]. The Perspective and Twitter data have an informal style, while email data consists of relatively formal expressions. Word embeddings are known to be domain-specific [25] and naturally domain-specific corpora are used to train word embeddings for use in a given domain. We note that in the absence of a large enough representative corpus to train domain-specific high-quality embeddings for this study, we reconcile with word embeddings trained on a large WikiCorpus [4] to capture the general lexical semantics and further tuned on a domain-specific corpus, such as that of the Perspective dataset, the spam emails data and the Twitter data. This step allows us to combine domain information in trained embeddings.

Error Correction. Word vectors permit us to decide the fit of a word in a given context by considering the geometry of the context words in relation to the given word. Firstly, the embedding of the compositional semantics of phrases or sentences can be approximated by a linear combination of the embeddings of their constituent words [31]. Let v_w be the embedding of the token w. Take the phrase "hate group" as an example, it holds that

$$v_{\text{hate\_group}} \approx \alpha v_{\text{hate}} + \beta v_{\text{group}},$$

where α and β are real-valued coefficients.

[2] available at: http://www.cs.upc.edu/~nlp/wikicorpus/

Table 2: Correction Accuracy on Perspective and Spam

Dataset	Our system	PyEnchant	Ekphrasis	Google
Perspective	**0.840**	0.476	0.713	0.323
Spam	**0.786**	0.618	0.639	0.526

This enables us to represent the context as a linear space of the component words. Another property is that semantically relevant words lie close in the embedding space. If a word fits the the context, it should be close to the context space, i.e., the normalized Euclidean distance from the word embedding to the context space should be small [10]. We quantify the inconsistency between the word and its context by the distance between the word embedding and the span of the context words.

For a misspelled word in a sentence, all words occurring within a window of size p are considered to be its context words. Let the context T_p be the set of words within distance p from w_0:

$$T_p = \{w_{-p}, w_{-p+1}, \ldots, w_0, \ldots, w_{p-1}, w_p\},$$

where w_0 is the misspelled word. Let C be the set of candidate replacements of w_0. Let v_i be the word embedding of context word w_i. We denote by $\text{dist}(c, T_p)$, the distance between a candidate c and the linear span of the words in its context T_p, termed as the candidate-context distance of a candidate, defined as the normalized distance between the word embedding of the candidate, v_c and its linear approximation obtained by the context vectors:

$$\text{dist}(c, T_p) = \min_{\{a_i\}} \frac{1}{\|v_c\|_2} \| \sum_{\substack{i=-p, \\ i\neq 0}}^{p} a_i v_i - v_c \|_2^2. \quad (1)$$

This is a quadratic minimization problem for which we can find a closed-form solution.

Instead of fixing one context window size, we consider multiple window sizes and weigh the distances obtained using different window sizes. The context words that are closer to the misspelled word are more informative in candidate selection. In the sentence "the stu*pid and stubborn administrators", the word *stubborn* suggests that the misspelled word should be a negative personality adjective, and the word *administrator* that it should be an adjective for people. Thus, the closest context word *stubborn* provides more relevant information than the distant word *administrator*.

We thus weigh the distance by the inverse of the context window size, i.e., the weight $\frac{1}{p}$ for the window size p. Suppose that T is the full context, and the candidate-context distance is defined below:

$$\text{dist}(c, T) = \sum_{p=1}^{P} \frac{1}{p} \cdot \text{dist}(c, T_p), \quad (2)$$

where $P = 4$ in our experiments.

Given the context T, the suggested correction c^* is the candidate with the smallest distance to the linear space of context words, i.e.,

$$c^* = \arg\min_{c \in C} \text{dist}(c, T). \quad (3)$$

5 EXPERIMENTS

We evaluated our spelling correction approach in three settings: toxicity and spam detection with synthetic misspellings, and hate speech detection with real errors. Even though some recent works using neural networks [9] are available, they require ground truth corrections for supervised training. For our unsupervised approach, we compare its performance with that of three strong unsupervised baselines below, which are generic off-the-shelf tools used in many NLP systems such as AbiWord [1] and Google search engine [2].
(1) Pyenchant: a generic spell checking library of multiple correction algorithms [19]. We use the MySpell library in this work.
(2) Ekphrasis: a spelling correction and text normalization tool for texts from social networks [5].
(3) Google spell checker: a Google search engine-based spell check [6]. We chose it for its ability to undertake context-sensitive correction on the basis of user search history.

Table 3: Correction on Perspective Data

Original	**stupid** and stubborn administrators	anti American **hate** groups
Revised	**stu*pid** and stubborn administrators	anti American **ahte** groups
Our result	**stupid** and stubborn administrators	anti American **hate** groups
PyEnchant	**sch*pid** and stubborn administrators	anti American **hate** groups
Ekphrasis	**stupid** and stubborn administrators	anti American **ate** groups
Google	**stupid** and stubborn administrators	anti American **ahte** groups

5.1 Toxicity Detection

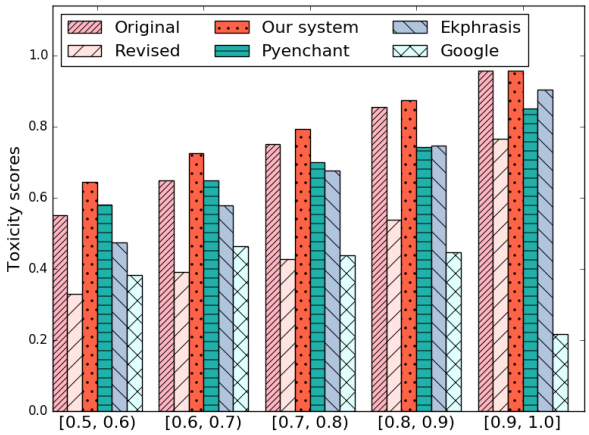

Figure 1: Toxicity scores by different approaches.

We took 2380 sentences from Perspective dataset whose toxicity scores were higher than 0.5 and added synthetic errors maliciously to these sentences as described in Section 3. Some example Perspective sentences, revised sentences and corrections given by different algorithms are shown in Table 3. The correction accuracy achieved by different approaches is reported in Table 2.

We divided toxic sentences into 5 bins according to their original toxicities. In Fig. 1, we report the average toxicity of the original, revised and corrected sentences in each bin. We see that the malicious errors drastically reduce the sentences' toxicities. The original average toxicity in the first bin is 0.55, whereas the revised toxicity is only 0.33 (a drop by 40%). This shows that toxicity detection is very sensitive to spelling errors.

Table 4: Correction Examples on Spam Data

Original sentence	it really be a **great** oppotunity to make relatively easy **money** , with little **cost** to you .	we have quit our **jobs** , and will soon buy a home on the beach and live off the interest on our **money** .
Revised sentence	it really be a **grfat** opportunity to make relatively easy **fmoney**, with little **cosgt** to you.	we have quit our **tobs**, and will soon buy a home on the beach and live off the interest on our **jmoney**.
Our correction	it really be a **great** opportunity to make relatively easy **money**, with little **cost** to you.	we have quit our **jobs**, and will soon buy a home on the beach and live off the interest on our **money**.
PyEnchant	it really be a **graft** opportunity to make relatively easy **fmoney**, with little **cost** to you.	we have quit our **bots**, and will soon buy a home on the beach and live off the interest on our **money**.
Ekphrasis	it really be a **great** opportunity to make relatively easy **money**, with little **cost** to you.	we have quit our **tobs**, and will soon buy a home on the beach and live off the interest on our **money**.
Google	it really be a **great** opportunity to make relatively easy **fmoney**, with little **cosgt** to you.	we have quit our **jobs**, and will soon buy a home on the beach and live off the interest on our **jmoney**.

From the same figure we see that our proposed error correction results in toxicity scores closer to the original ones when compared with the baselines, validating the effectiveness of our approach. We note that in some cases our approach might result in a slightly higher toxicity score than the original one, because of the pre-existing misspellings in the original sentences. For example, original sentences "you are a *fagget*", "*fukkin* goof's track record" and "I suspect that closedmouth is *g*y*" have crude misspellings, which are used as Internet slang. Our approach corrects these pre-existing misspellings in addition to the errors we add later, resulting in higher toxicity scores of corrected sentences. We perform corrections of all spelling errors, recovering more toxic words than we added maliciously.

5.2 Spam Detection

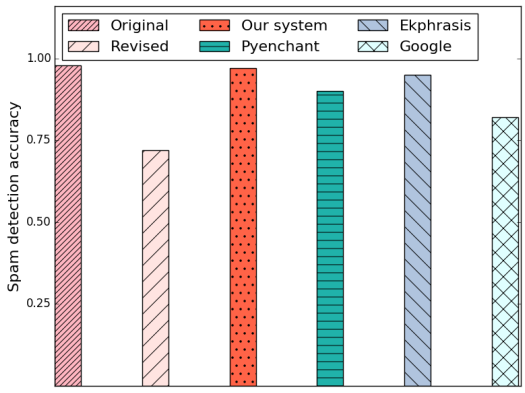

Figure 2: Spam detection accuracy on test emails.

We next evaluate our spelling correction on the spam data. Synthetic malicious errors were added to spam-indicative words in spam mails based on our misspelling generation mechanism.

Some example spams are shown in Table 4. Sensitive words such as "money" which are indicative of spams are highlighted, and the corrections given by different approaches are shown. Table 2 shows the spelling correction accuracy achieved by our algorithm and the baselines for the spam data.

Fig. 2 shows the spam detection accuracy on the original, revised and corrected test emails. Again we see that malicious errors resulted in a large accuracy drop, the accuracy was restored after spelling correction, and the increase in accuracy using our approach was the maximal among those compared.

5.3 Real Misspellings in Tweet Hate Speech Detection

We have shown that synthetic misspellings are able to deceive the toxicity detector and the spam detector, and reported the performance of spell checkers on the synthetic data. We also experimented with data containing real spelling errors collected from Twitter, where instances of hate speech contain user-generated spelling errors. The correction performance of spell checkers are now compared on the real misspellings.

Table 5: Example of Tweets and Hate Speech Categories

Category	Example sentence
Racism	#isis #islam pc puzzle: converting to a religion of peace leading to violence? http://t.co/tbjusaemuh
Sexism	Real question is do feminist liberal bigots understand that different rules for men/women is sexism
Neither	The mother and son team are sooooo nice !!!

Tweet normalization. Some example tweets of racist and sexist nature are shown in Table 5. Tweets are notoriously noisy and unstructured given the frequent occurrences of hashtags, URLs, reserved words and emojis. These non-standard tokens greatly increase the vocabulary size of tweets while also injecting noise to classification tasks. We use Tweet Preprocessor, a tweet preprocessing tool which can replace aforementioned tokens with a special set of tokens. For example, one tweet is "#isis #islam pc puzzle: converting to a religion of peace leading to violence ?,http://t.co/tbjusaemuh", which becomes "$HASHTAG$ $HASHTAG$ pc puzzle: converting to a religion of peace leading to violence? URL" after preprocessing. This preprocessing stage can clean texts without dealing with misspellings. Tweets are preprocessed right before they are fed into the neural network.

Hate speech detection. The state-of-the-art system for hate speech detection is a Bidirectional Long Short Term Memory (BLSTM) network with attention mechanism [3]. The model takes a sequence of words in a tweet, obtains word embeddings in the embedding layer, and passes them to bidirectional recurrent layers to generate

Table 6: Hate Speech Detection Results with Different Spelling Corrections

Category	Metric	Original	Our system	PyEnchant	Ekphrasis	Google
racist	Precision	0.630	**0.640**	0.630	0.630	0.569
	Recall	0.617	0.681	0.617	0.617	**0.702**
	F1 score	0.623	**0.660**	0.623	0.623	0.628
sexist	Precision	0.641	0.630	0.629	0.629	**0.649**
	Recall	0.775	0.767	**0.790**	0.790	0.759
	F1 score	0.701	0.692	**0.701**	**0.701**	0.700
Macro average over all categories	Precision	0.721	**0.721**	0.718	0.718	0.703
	Recall	0.741	0.757	0.743	0.743	**0.759**
	F1 score	0.727	**0.737**	0.727	0.727	0.727

a dense representation of the input tweet. The feedforward layer takes the tweet vector and predicts its probability distribution over all three classes. The class with the highest likelihood is chosen as the category of the input tweet. In our experiment, we use a BLSTM model for the hate speech detection task.

Vocabulary construction. Firstly we build a vocabulary list using both a standard dictionary and the frequent words (frequency higher than 5) in the training data. This list will serve as a reference for spelling correction; words outside the list will be taken as spelling errors and replaced with legal words from the vocabulary. The reason for collecting words from the training data is to include the Internet slangs in tweets which may not exist in the standard dictionary. Some examples are "tmr" (for *tomorrow*), "fwd" (for *forward*) and "lol" (for *laughing out loudly*). Some previous works proposed to replace these slang terms with their standard forms [13, 23], which require either expert knowledge or human annotations. We argue that because these Internet slangs change rapidly we should understand them in a data-driven manner instead of standardizing them based on human knowledge. Since the representation of these slangs and their use in hate speech will be learned by the neural network from the train data, we add these slangs to our vocabulary as legal words.

Spelling correction. Misspellings are another source of noise which cannot be handled in the tweet normalization stage. Some misspellings are user-created to deceive the online detection system. For example, the swear word "fucking" has a lot of variants in tweets such as "fckn", "f*ckin", "f**king", "fckin", and "fuckin". When a new variant of a swear word arises in the test data, the model takes it as an out-of-vocabulary word, and is unable to match it with any learned pattern. The purpose of spelling correction is to map these new variants to words that are known to the model. As discussed in Section 4, we enumerate the candidates of a misspelled word, and choose the candidate which best fits with the context as its correction.

Results. We do a random 80:20 train-test split of the Twitter dataset. A detection system was trained on the normalized train data (without spelling correction) using the state-of-the-art BLSTM model. There were 3218 test tweets, 571 of which had misspellings. We applied the different spelling correction approaches to these 571 tweets, and the corrected tweets were then cleaned as described in the tweet normalization stage. Processed tweets were input to the trained detection system. We compared the hate speech detection results on the 571 test tweets to evaluate the effect of spelling correction on this downstream application. As shown in Table 6, We report results on the original test data, and also on the test data which are corrected by our approach, PyEnchant, Ekphrasis, and google spell checker. We report precision, recall and F1-score of racist and sexist classification respectively and the macro-averages (to evaluate the overall performance).

The best performance of each metric is highlighted in the table. Compared with the classification performance on the test data with spelling errors for the racist category, our approach improves the precision by 1%, the recall by 6.4%, and the F1 score by 3.7% absolute points. Both PyEnchant and Ekphrasis spell checkers improve the recall of sexist category, but decrease the precision, so their corrected forms achieve F1 scores similar to the original test data. Google spell checker also gives similar F1 score on both racist and sexist classes. Our approach outperforms the other baselines in terms of F1 score for the racist class and the macro F1 score.

For sexism-related tweets, our approach does not improve the results compared to the original test data. Taking a closer look at these tweets we notice that they often contain some abbreviations which might be taken as misspellings, and that their contextual information is insufficient to decide the appropriate corrections. For example, in the sexist tweet "I'm sorry but if you watch women ufc fights kys". Our approach replaces *kys* with *keys*, and the trained neural network misclassified the corrected tweet. Another instance is "these nsw promo girls think way too highly of themselves", where *nsw* is incorrectly replaced with *new* by our approach.

6 CONCLUSION

In this study, we showed how malicious spelling errors can deceive profanity- and spam detectors. To deal with these malicious misspellings, we proposed a context-sensitive spelling corrector based on word embeddings. Our spell checker is light-weight, unsupervised and can be easily incorporated into downstream applications. It achieved a favorable spelling correction performance when compared with general purpose spell-checking tools such as PyEnchant, Ekphrasis and Google spell checkers on both synthetic and real misspellings from different datasets.

ACKNOWLEDGMENTS

This work is supported in part by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network.

REFERENCES

[1] 2018. AbiWord. Available at: https://www.abisource.com.

[2] 2018. Google Search Engine. Available at: https://www.google.com.

[3] Sweta Agrawal and Amit Awekar. 2018. Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms. In *European Conference on Information Retrieval*. Springer, 141–153.

[4] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed Word Representations for Multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, 183–192. http://www.aclweb.org/anthology/W13-3520

[5] Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, 747–754.

[6] Noah Coad. 2018. Google Spell Check. Available at: https://github.com/noahcoad/google-spell-check.

[7] Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7, 3 (1964), 171–176.

[8] Noura Farra, Nadi Tomeh, Alla Rozovskaya, and Nizar Habash. 2014. Generalized Character-Level Spelling Error Correction.. In *ACL (2)*. 161–167.

[9] Shaona Ghosh and Per Ola Kristensson. 2017. Neural Networks for Text Correction and Completion in Keyboard Decoding. *arXiv preprint arXiv:1709.06429* (2017).

[10] Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2017. Geometry of Compositionality.

[11] Jigsaw Google. 2017. Perspective. Available at: https://www.perspectiveapi.com/.

[12] Sergey Gubanov, Irina Galinskaya, and Alexey Baytin. 2014. Improved Iterative Correction for Distant Spelling Errors.. In *ACL (2)*. 168–173.

[13] Itisha Gupta and Nisheeth Joshi. 2017. Tweet normalization: A knowledge based approach. In *Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS), 2017 International Conference on*. IEEE, 157–162.

[14] Hany Hassan and Arul Menezes. 2013. Social Text Normalization using Contextual Graph Random Walks.. In *ACL (1)*. 1577–1586.

[15] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving Google's Perspective API Built for Detecting Toxic Comments. *arXiv preprint arXiv:1702.08138* (2017).

[16] Fraser Howard. 2008. Web Attacks: Modern web attacks. *Network Security* 2008, 4 (2008), 13–15.

[17] Dan Jurafsky and James H Martin. 2014. *Speech and language processing*. Vol. 3. Pearson London.

[18] Daniel Jurafsky and James H. Martin. 2017. Distributed Representations of Words and Phrases and their Compositionality. In *Speech and Language Processing*. Chapter 5, 1–12.

[19] Ryan Kelly. 2015. PyEnchant. Available at: https://github.com/rfk/pyenchant.

[20] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[21] Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 71–76.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119.

[23] Abiodun Modupe, Turgay Celik, Vukosi Marivate, and Melvin Diale. 2017. Semi-supervised probabilistics approach for normalising informal short text messages. In *Information Communication Technology and Society (ICTAS), Conference on*. IEEE, 1–8.

[24] Howard L. Morgan. 1970. Spelling correction in systems programs. In *Communications of the ACM*. 90–94.

[25] Thien Huu Nguyen and Ralph Grishman. 2014. Employing Word Representations and Regularization for Domain Adaptation of Relation Extraction.. In *ACL (2)*. 68–74.

[26] Etienne Papegnies, Vincent Labatut, Richard Dufour, and Georges Linares. 2017. Impact Of Content Features For Automatic Online Abuse Detection. In *International Conference on Computational Linguistics and Intelligent Text Processing*.

[27] Aurelia Power, Anthony Keane, Brian Nolan, and Brian O'Neill. 2018. Detecting Discourse-Independent Negated Forms of Public Textual Cyberbullying. *Journal of Computer-Assisted Linguistic Research* 2, 1 (2018), 1–20.

[28] Luz Rello, Miguel Ballesteros, and Jeffrey P Bigham. 2015. A spellchecker for dyslexia. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. ACM, 39–47.

[29] Eric Sven Ristad and Peter N Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 5 (1998), 522–532.

[30] Sergio A Rojas-Galeano. 2013. Revealing non-alphabetical guises of spam-trigger vocables. *Dyna* 80, 182 (2013), 15–24.

[31] Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A Word Embedding Approach to Predicting the Compositionality of Multiword Expressions.. In *HLT-NAACL*. 977–983.

[32] Manish Saxena and PM Khan. 2015. Spamizer: An approach to handle web form Spam. In *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*. IEEE, 1095–1100.

[33] Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 314–323.

[34] Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*. 88–93.

[35] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 1113–1120.

[36] Zar Zar Wint, Theo Ducros, and Masayoshi Aritsugi. 2017. Spell corrector to social media datasets in message filtering systems. In *Digital Information Management (ICDIM), 2017 Twelfth International Conference on*. IEEE, 209–215.

[37] Zar Zar Wint, Théo Ducros, and Masayoshi Aritsugi. 2018. Non-words Spell Corrector of Social Media Data in Message Filtering Systems. *Journal of Digital Information Management* 16, 2 (2018).

[38] Xinwang Zhong. 2014. Deobfuscation based on edit distance algorithm for spam filitering. In *Machine Learning and Cybernetics (ICMLC), 2014 International Conference on*, Vol. 1. IEEE, 109–114.