Textual Pattern Recognition and Text Generation with Deep Language Models

Oral Qualifying Exam Prospectus

Yichao Zhou
University of California, Los Angeles
yz@cs.ucla.edu

Abstract

Deep language models have been popular in natural language processing and text mining. The contexualized training enables the language model to capture richer and deeper semantic and syntactic proximity among words and sentences. This prevailing embedding method brought with a revolution in various pattern recognition tasks. For example, BERT and its new version, RoBERTa are refreshing the state-of-the-art results of the named entity recognition benchmarks. However, recognition problems in specific domains with small sets of training data such as figurative language processing and biomedical text mining still require a careful utilization of the deep language models. Besides, in aspects of language generation, few research works have been done and how to leverage deep language models remains an open question. In this prospectus, I will introduce three recent published and ongoing researches on the textual pattern detection and generation with deep language models. Extensive experiments show that our proposed models gain extraordinary improvements compared to the baseline models.

1 Introduction

Textual pattern recognition is one important subtask of text mining and natural language processing. Textual pattern recognition is the automated recognition of patterns and regularities in text data. For instance, named entity recognition (NER) identifies the types of specific words in sentences; relation extraction collects relational triples with two named entities and their relation. Traditional methods encode words and sentences using static word embeddings such as Word2Vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014b). In recent years, contextualized embeddings (Peters et al., 2018a; Devlin et al., 2018a; Yang et al., 2019; Liu et al., 2019) trained with deep language models were applied to these tasks and gained huge improvements (Imani et al., 2019; Gupta et al., 2019). Contextualized embeddings can capture precise characteristics of words by deriving proper word representations with consideration of contexts. However, in term of text in specific domains with small language corpus, directly applying pretrained contextualized embeddings will not achieve large improvement as expected. We suppose the conditional probability learnt by pretrained language models on the corpus of the general domain might be different from those specific domains. Additional sets of features could be necessary for learning a model in this case.

Text generation includes downstream applications such as question answering and text summarization. Different from learning the joint distribution of words in corpus as GAN (Goodfellow et al., 2014) does, we talk about learning the conditional probability here, i.e. predicting the next word given the current word or context. Few works have been done in the field of text generation using deep language models. However, contextualized encoding of features can enable a better prediction of next word based on the context.

In this prospectus, I would introduce three published or ongoing related research works. The first one is pronunciation-attentive contextualized pun recognition. Besides the deep contextualized embeddings, we also leverage phonetic features to improve the pun detection and location tasks. The second work is about learning to discriminate perturbations for blocking adversarial attacks in text classification. Deep language models are utilized to recognize the adversarial attacks and to generate the original word or word with similar semantic meaning to recover the text classification performance before attacks. The last work is to apply the contextualized language models to named entity recognition in clinical case reports. Pretraining a biomedical BERT with PubMed abstracts and PMC corpus is time-consuming and computationally expensive. We only extract a domain-specific corpus, which is much smaller, to pretrain the model, surprisingly obtaining a better recognition performance.

These three works focus on textual pattern recognition and text generation tasks leveraging deep language models. Deep language models are powerful tools, which can easily boost the performance of tasks in general domains. However, we need to be careful when using them in specific domains that lack large labeled dataset, such as figurative language processing and biomedical text mining. Significant improvements from extensive experiments are demonstrated.

2 "The Boating Store Had Its Best Sail Ever": Pronunciation-attentive Contextualized Pun Recognition

Humor plays an important role in human languages and interaction. For example, puns perform wordplay for humorous effects by exploiting words with multiple entendres and high phonological similarity. However, humor in natural languages are usually extremely implicit so that machines can have a hard time recognizing the whimsy for natural language understanding. In this paper, we propose Pronunciation-attentive Contextualized Pun Recognition (PCPR) to perceive human humor by detecting and locating the puns in natural language text. The technique of contextualized language modeling is first deployed to derive contextualized word representations. In addition, each word is decomposed into phonetic symbols and encoded by self-attention mechanism, thereby capturing phonological delights. Finally, the joint representations can benefit both of pun detection and location. Extensive experiments have been conducted on two publicly available datasets. The experimental results demonstrate that our approach significantly outperforms the state-of-the-art baseline methods.

2.1 Background and Motivation

Humor is one of the most intricate behaviors in natural language semantics so that even humans sometimes cannot comprehend the complicated whimsy in amusing sentences. To represent humorous ideas, puns are a ubiquitous approach that exploits the different meanings of homographs or homophones of words to explain texts or utterances. More specifically, homographic puns rely on multiple word interpretations (e.g., the word *bass* as a *guitar* or a *fish*) while heterographic puns take advantage of phonologically similar homophone words (e.g., the words *dear* and *deer*). In other words, understanding puns can be a big fish to fry for deeper comprehension of complex semantics. However, although recent researchers have demonstrated their interests in related topics, such as pun detection (Cai et al., 2018) and generation (He et al., 2019), the drawn attention is relatively insufficient so that there is still huge room for improvement of existing approaches.

The ultimate goal of pun recognition is to identify equitable intention of words in utterances and texts so that word sense disambiguation (WSD) (Yarowsky, 1995; Navigli, 2009) intuitively becomes one of the feasible approaches to recognize puns. For example, Pedersen (2017) deploys different WSD methods to identify the senses of words and sentences for pun recognition. External knowledge bases such as WordNet (Miller, 1998) have been applied in determining word senses of pun words (Oele and Evang, 2017). However, these methods cannot tackle heterographic puns with distinct word spellings while knowledge bases only incorporate very limited words. To resolve the issues of sparseness and homophones, the word embedding techniques (Mikolov et al., 2013a; Pennington et al., 2014a) provide more flexible representations to model puns (Hurtado et al., 2017; Indurthi and Oota, 2017; Cai et al., 2018; He et al., 2019). However, a word should have very different representations regarding its contexts and the way to deliver puns, a word embedding can be insufficient to represent all possible semantics.

Puns are an art that interacts with the entire text, so the contexts of words also affect pun representations, including appropriate word embeddings for a specific text. To capture the precise characteristics of words, contextualized word embeddings (Peters et al., 2018b; Devlin et al., 2018a) derive proper word representations with consideration of contexts and have been applied in many domains, such as information retrieval (Imani et al., 2019) and text mining (Gupta et al., 2019). Intuitively, contextualized word embeddings can be also beneficial to pun recognition. For each word in texts, the potential for being puns and the actual meanings in texts can both be reflected in the embeddings that consider the contexts. However, none of the previous studies applies the technique of contextualized word embeddings into pun recognition.

Although contexts are capable of carrying much information, only considering context information is still insufficient because heterographic puns count on phonological structures to create whimsy. However, most of the previous studies in pun recognition are unaware of exploiting either phonological structures or pronunciations. More specifically, the only two related studies (Doogan et al., 2017; Jaech et al., 2016) apply pronunciations to recover the intended words of puns, but all of the puns need to be given in texts so that those approaches cannot tackle the pun recognition task.

In this paper, we propose Pronunciation-attentive Contextualized Pun Recognition (PCPR) to recognize puns in texts for two tasks: pun detection and pun location. The words in the input text are first encoded in embedding vectors based on bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018a) as contextualized word embeddings. To capture phonological structures of words, an intuitive approach is to break each word into a sequence of phonemes as its pronunciation so that homophones can have similar sets of phonemes. For instance, the phonemes of the word *pun* are {P, AH. N}. For each word, the corresponding phonemes are derived from the phonological structures and projected as phoneme embeddings. The attention mechanism (Bahdanau et al., 2014) is then applied to identify important phonemes and obtain pronunciation embeddings for each word. Based on the contextualized word embeddings and pronunciation embeddings, the self-attention mechanism (Vaswani et al., 2017) is applied to encode the whole input text into a self-attentive embedding. Finally, the self-attentive embedding can be deployed to tackle both pun detection and location.

2.2 Pronunciation-attentive Contextualized Pun Recognition

In this section, we first formally define the problem, and then introduce the proposed method, PCPR.

2.2.1 Problem Statement

Suppose the input text consists of a sequence of N words $\{w_1, w_2, \cdots, w_N\}$. For each word w_i with M_i phonemes in its pronunciation, the phonemes are denoted as $R(w_i) = \{r_{i,1}, r_{i,2}, \cdots, r_{i,M_i}\}$, where $r_{i,j}$ is the j-th phoneme in the pronunciation of w_i. In this paper, we aim to recognize potential puns in the text with two tasks, including pun detection and pun location, as described in the following.

Task 1: Pun Detection. In pun detection task, the input text does not necessarily contain a pun. The goal of this task is to predict the binary label y^D that indicates if the text contains at least a pun. More specifically, pun detection can be treated as a task of binary classification.

Task 2: Pun Location. Different from pun detection, this task assumes each input text contains at least one pun word. Given the words in the input text, pun location aims to unearth the pun word. More precisely, for each word w_i, we would like to predict the binary label y_i^L that indicates if w_i is a pun word.

In addition to independently solving the above two tasks, the ultimate goal of pun recognition is to build a pipeline from scratch to detect and then locate the puns in texts. Hence, we also evaluate the pipeline performance by aggregating the solutions for two tasks. Note that any positive prediction for texts without a pun is regularly considered to be incorrect, so the performance of pun detection will also affect the task of pun location.

2.2.2 Framework Overview

Figure 1 shows the overall framework of our proposed Pronunciation-attentive Contextualized Pun Recognition (PCPR). For each word in the input text, we first derive two continuous vectors, including contextualized word embedding and pronunciation embedding, as representations in different aspects. Contextualized word embeddings derive appropriate word representations with consideration of context words and capture the accurate semantics in the text. To learn the phonological characteristics, each word is divided into phonemes while each phoneme is projected to a phoneme embedding space, thereby obtaining pronunciation embeddings with the attention mechanism (Bahdanau et al., 2014). Finally, a self-attentive encoder blends contextualized word embeddings and pronunciation embeddings to capture the overall semantics for both pun detection

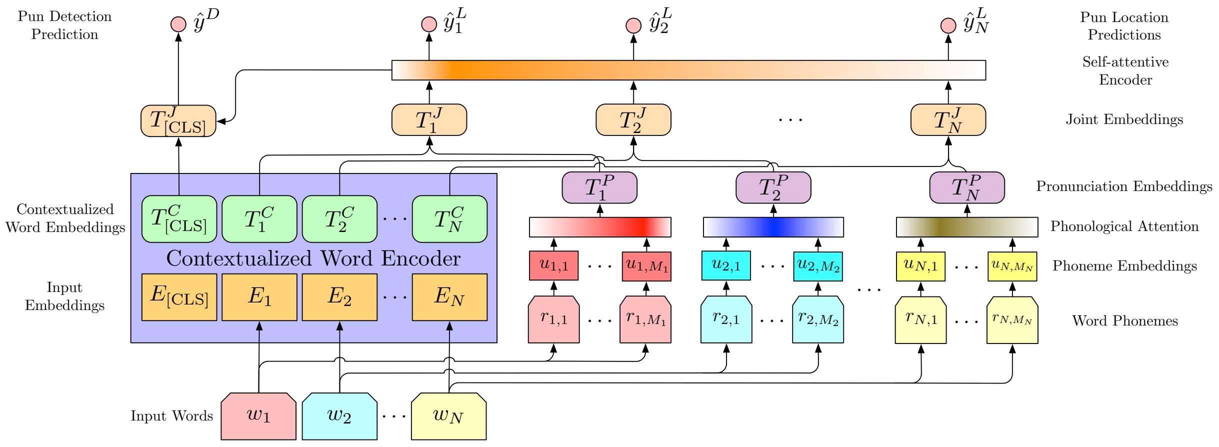


Figure 1: The overall framework of PCPR.

and location.

2.2.3 Contextualized Word Embeddings

To understand the specific semantics of a word in the text, it is essential to consider the contexts. Hence, we propose to apply contextualized word embeddings to derive word representations. In the framework of PCPR, any contextualized word embedding method, such as BERT (Devlin et al., 2018a) and ELMo (Peters et al., 2018a), can be utilized. In this paper, we choose BERT as the state-of-the-art approach to derive contextualized word embeddings.

BERT deploys a multi-layer bidirectional encoder based on transformers with multi-head self-attention (Vaswani et al., 2017) to model words in the text after integrating both word and position embeddings (Sukhbaatar et al., 2015). As a result, all of the contexts can be conditioned in all layers of BERT, thereby deriving representative contextualized word embeddings. Here we denote T_i^C as the d_C-dimensional contextualized word embedding for the word w_i. In addition, BERT contains a special position [CLS] with an embedding vector in BERT to represent the semantics of the whole input text.

2.2.4 Pronunciation Embeddings

To learn the phonological characteristics of words, PCPR models the word phonemes. For each phoneme $r_{i,j}$ of the word w_i, we project $r_{i,j}$ to a d_P-dimensional embedding space as a trainable vector $u_{i,j}$ to represent its phonological properties.

Based on the phoneme embeddings of a word, we apply the attention mechanism (Bahdanau et al., 2014) to simultaneously identify important phonemes and derive the pronunciation embedding T_i^P. More specifically, the phoneme embeddings are transformed by a fully-connected hidden layer to measure the importance scores α_i^P as follows:

$$v_{i,j} = \tanh(\mathcal{F}_P(u_{i,j})),$$

$$\alpha_{i,j}^P = \frac{v_{i,j}^T v_s}{\sum_k v_{i,k}^T v_s},$$

where $\mathcal{F}_P(\cdot)$ is a fully-connected layer with d_A outputs and d_A is the attention size; v_s is a d_A-dimensional context vector to estimate importance scores. Finally, the pronunciation embeddings T_i^P can be represented as the weighted combination of phoneme embeddings as follows:

$$T_i^P = \sum_j \alpha_{i,j} u_{i,j}.$$

Moreover, we can further derive the joint embedding T_i^J to indicate both word semantics and phonological knowledge for the word w_i by concatenating two different embeddings as follows:

$$T_i^J = \left[T_i^C; T_i^P\right].$$

Note that the joint embeddings are d_J-dimensional vectors, where $d_J = d_C + d_P$.

2.2.5 Pronunciation-attentive Contextualized Embedding with Self-attention

For the task of pun detection, the understanding of the whole input text is essential. To encode the input text, we deploy the self-attention mechanism (Vaswani et al., 2017) to capture the overall semantics represented in the joint embeddings. For each word w_i, the self-attention mechanism estimates an importance vector α_i^S:

$$\mathcal{F}_S(Q) = QW_SQ^T,$$

$$\alpha_i^S = \frac{\exp(\mathcal{F}_S(T_i^J))}{\sum_j \exp(\mathcal{F}_S(T_j^J))},$$

where $\mathcal{F}_S(\cdot)$ is the function to estimate the attention for queries, and W_S is a trainable weight matrix. Hence, the self-attentive embedding vector can be computed by aggregating joint embeddings:

$$T^J_{\texttt{[ATT]}} = \sum_i \alpha_i^S \cdot T_i^J.$$

Note that the knowledge of pronunciations is considered by the self-attentive encoder but not the contextualized word encoder. Finally, the pronunciation-attentive contextualized representation for the whole input text can be derived by concatenating the overall contextualized embedding and the self-attentive embedding:

$$T^J_{\texttt{[CLS]}} = \left[T^C_{\texttt{[CLS]}}; T^J_{\texttt{[ATT]}}\right].$$

Moreover, each word w_i can also be benefited by the self-attentive encoder and obtains a self-attentive joint embedding:

$$T^J_{\texttt{i, [ATT]}} = \alpha_i^S \cdot T_i^J.$$

2.2.6 Inference and Optimization

Based on the joint embedding for each word and the pronunciation-attentive contextualized embedding for the whole input text, both tasks can be easily tackled with simple fully-connected layers.

Pun Detection. Pun detection can be modeled as a task of sequence binary classification. Given the overall embedding for the input text $T^J_{\texttt{[CLS]}}$, the prediction $\hat{y}^D$ can be generated by a fully-connected layer and the softmax function:

$$\hat{y}^D = \operatorname*{argmax}_{k \in \{0,1\}} \mathcal{F}_D(T^J_{\texttt{[CLS]}})_k,$$

where $\mathcal{F}_D(\cdot)$ derives the logits of two classes in binary classification.

Pun Location. For each word w_i, the corresponding self-attentive joint embedding $T^J_{\texttt{i, [ATT]}}$ is applied as features for pun location. More specifically, the prediction $\hat{y}_i^L$ can be also generated in a similar way to pun detection as:

$$\hat{y}_i^L = \operatorname*{argmax}_{k \in \{0,1\}} \mathcal{F}_L(T^J_{\texttt{i, [ATT]}})_k,$$

where $\mathcal{F}_L(\cdot)$ derives two logits for classifying if a word is a pun word.

For optimization, since both of the tasks focus on binary classification, Cross-Entropy (De Boer et al., 2005) is applied as our loss function.

Dataset	SemEval		PTD
	Homo	Hetero	
Examples w/ Puns	1,607	1,271	2,423
Examples w/o Puns	643	509	2,403
Total Examples	2,250	1,780	4,826

Table 1: Data statistics of two datasets. Note that "Homo" and "Hetero" denote homographic and heterographic puns. Pun detection employs all of the examples in the two datasets while pun location only exploits the examples with puns in SemEval due to the limitation of annotations.

2.3 Experiments

2.3.1 Experiment settings

Experimental Datasets. The experiments have been conducted on the SemEval 2017 shared task 7 dataset[1] (SemEval) (Miller et al., 2017) and the Pun of The Day dataset (PTD) (Yang et al., 2015). For pun detection, the SemEval dataset consists of $4,030$ and $2,878$ examples for pun detection and location while each example with a pun can be a homographic or heterographic pun. In contrast, the PTD dataset provides $4,826$ examples without labels of pun types. Table 1 further shows the data statistics.

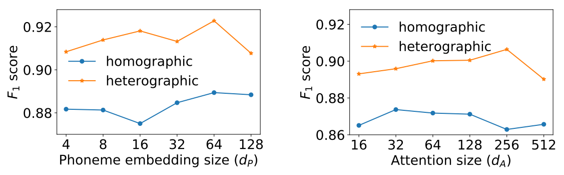
Evaluation Metrics. We adopt precision (P), recall (R), and F_1-score (Manning et al., 2010; Powers, 2011) to compare the performance of PCPR with previous studies in both pun detection and location. Note that the experimental settings are consistent with previous studies (Zou and Lu, 2019; Cai et al., 2018). More specifically, we apply 10-fold cross-validation to conduct evaluation.

Implementation Details. For data preprocessing, all of the numbers and punctuation marks are removed as noises. The phonemes of each word are derived by the CMU Pronouncing Dictionary[2]. To obtain better performance, the *fastText* word embedding model (Mikolov et al., 2018) is applied to pre-train the phoneme embeddings with a dump of Wikipedia articles[3] crawled in December, 2017. The PCPR is implemented in PyTorch while the fused Adam optimizer (Kingma and Ba, 2014) optimizes the parameters with an initial learning rate of 5×10^{-5}. The dropout

[1]`http://alt.qcri.org/semeval2017/task7/`

[2]`http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/`

[3]`https://dumps.wikimedia.org/enwiki/latest/`



(a) Phoneme emb. size d_P (b) Attention size d_A

Figure 2: Pun detection performance over different phoneme embedding sizes d_P and attention sizes d_A on the SemEval dataset.

and batch size are set as 10^{-1} and 32. We follow BERT (BASE) (Devlin et al., 2018a) to obtain 12 Transformer layers and self-attention heads. We also create a variant of PCPR called CPR by exploiting only the contextualized word encoder without considering phonemes to demonstrate the effectiveness of pronunciation embeddings.

To fine-tune the hyperparameters, we search the phoneme embedding size d_P and the attention size d_A from $\{4, 8, 16, 32, 64, 128, 256, 512\}$ as shown in Figure 2. For the SemEval dataset, the best setting is $(d_P = 64, d_A = 256)$ for the homographic puns while heterographic puns favor $(d_P = 64, d_A = 32)$. For the PTD dataset, $(d_P = 64, d_A = 32)$ can reach the best performance.

Baseline Methods. We compare PCPR with several baseline methods with reported performance for each of the datasets.

For the SemEval dataset, nine baseline methods are compared in the experiments. As a shared task, the baselines involve some participated methods, including Duluth (Pedersen, 2017), JU_CES_NLP (Pramanick and Das, 2017), PunFields (Mikhalkova and Karyakin, 2017), UWAV (Vadehra, 2017), Fermi (Indurthi and Oota, 2017), and UWaterloo (Vechtomova, 2017), while most of them extract complicated linguistic features to train rule based and machine learning based classifiers. In addition to task participants, Sense (Cai et al., 2018) incorporates word sense representations into RNNs to tackle the homographic pun location task. The traditional CRF (Zou and Lu, 2019) can absorb linguistic features to model puns. Moreover, the Joint method (Zou and Lu, 2019) can jointly model two tasks with RNNs and a CRF tagger.

For the PTD dataset, four baseline methods with reported performance are selected for comparisons. MCL (Mihalcea and Strapparava, 2005) exploits word representations with multiple stylistic features while HAE (Yang et al., 2015) applies a random forest model with Word2Vec and human centric features. PAL (Chen and Lee, 2017) trains a convolutional neural network (CNN) to learn essential feature automatically. Based on existing CNN models, HUR (Chen and Soo, 2018) improves the performance by adjusting the filter size and adding a highway layer.

2.4 Experimental Results

Pun Detection. Table 2 presents the pun detection performance of methods for both homographic and heterographic puns on the SemEval dataset while Table 3 shows the detection performance on the PTD dataset. For the SemEval dataset, comparing to the nine baseline models, PCPR achieves the highest performance with 2.7% and 4.9% improvements of F_1 against the best Joint method for the homographic and heterographic datasets, respectively. For the PTD dataset, PCPR also has an 8.7% improvement against the best HUR method. Moreover, the variant CPR beats all of the baseline methods and shows the effectiveness of contextualized word embeddings. In addition, PCPR further improves the performances by 2.3% and 0.4% with the attentive pronunciation feature for detecting homographic and heterographic puns, respectively.

Pun Location. Table 2 also shows that our PCPR model achieves highest F_1-scores on both homographic and heterographic pun location tasks with 8.7% and 12.6% incredible increment against the best baseline method. The improvement is much larger than that on pun detection task. It can be because predicting pun locations relies much more on the comparative relations among different tokens in one sentence. As a result, contextualized word embeddings can acquire an enormous advantage. By applying the pronunciation-attentive representations, different words with similar pronunciations are linked, leading to a much better pinpoint of pun word for the heterographic dataset.

Pipeline Recognition. As described in Section 2.2.1, the ultimate goal of pun recognition is to establish a pipeline to detect and then locate puns. Table 4 shows the pipeline performances of PCPR and Joint, which is the only baseline with reported pipeline performance, for recognizing the homographic and heterographic puns in the

Model	Homographic Puns						Heterographic Puns					
	Pun Detection			Pun Location			Pun Detection			Pun Location		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Duluth	78.32	87.24	82.54	44.00	44.00	44.00	73.99	86.62	68.71	-	-	-
JU_CSE_NLP	72.51	90.79	68.84	33.48	33.48	33.48	73.67	94.02	71.74	37.92	37.92	37.92
PunFields	79.93	73.37	67.82	32.79	32.79	32.79	75.80	59.40	57.47	35.01	35.01	35.01
UWAV	68.38	47.23	46.71	34.10	34.10	34.10	65.23	41.78	42.53	42.80	42.80	42.80
Fermi	90.24	89.70	85.33	52.15	52.15	52.15	-	-	-	-	-	-
UWaterloo	-	-	-	65.26	65.21	65.23	-	-	-	79.73	79.54	79.64
Sense	-	-	-	81.50	74.70	78.00	-	-	-	-	-	-
CRF	87.21	64.09	73.89	86.31	55.32	67.43	89.56	70.94	79.17	88.46	62.76	73.42
Joint	91.25	93.28	92.19	83.55	77.10	80.19	86.67	93.08	89.76	81.41	77.50	79.40
CPR	91.42	94.21	92.79	88.80	85.65	87.20	93.35	**95.04**	94.19	92.31	88.24	90.23
PCPR	**94.18**	**95.70**	**94.94**	**90.43**	**87.50**	**88.94**	**95.00**	94.25	**94.62**	**94.23**	**90.41**	**92.28**

Table 2: Performance of nine baseline methods and PCPR with a variant CPR on the SemEval dataset. Comparing to PCPR, CPR only applies the contextualized word encoder without considering word pronunciations.

Model	P	R	F_1
MCL	83.80	65.50	73.50
HAE	83.40	88.80	85.90
PAL	86.40	85.40	85.70
HUR	86.60	94.00	90.10
CPR	98.12	**99.34**	98.73
PCPR	**98.44**	99.13	**98.79**

Table 3: Pun detection performance of four baselines and PCPR with a variant CPR on the PTD dataset.

Model	Homographic Puns			Heterographic Puns		
	P	R	F_1	P	R	F_1
Joint	67.70	67.70	67.70	68.84	68.84	68.84
PCPR	**87.21**	**81.72**	**84.38**	**85.16**	**80.15**	**82.58**

Table 4: Performance of two methods for pipeline recognition of homographic and heterographic puns in the SemEval dastaset.

SemEval dataset. It is obvious that Joint obtains an extremely low performance while the authors suppose that the detection and location tasks badly interfere with each other. In contrast, PCPR improves the F_1-scores against Joint by 24.6% and 20.0% on two pun types because it performs well in both tasks of pun detection and location.

2.4.1 Ablation Study and Analysis

Ablation Study. To better understand the effectiveness of each component in PCPR, we conduct an ablation study on the homographic puns of the SemEval dataset. Table 5 shows the results of a ablation study on different features of PCPR, including pre-trained phoneme embeddings, the self-attentive encoder, and phonological attention.

Model	P	R	F_1
PCPR	**90.43**	**87.50**	**88.94**
w/o Pre-trained Phoneme Emb.	89.37	85.65	87.47
w/o Self-attention Encoder	89.17	86.42	87.70
w/o Phonological Attention	89.56	87.35	88.44

Table 5: Ablation study on different features of PCPR for homographic pun detection on the SemEval dataset.

As a result, we can see the drop after removing each of three features. It shows that all of these components are beneficial for PCPR to recognize puns.

Attentive Weights Interpretation. Figure 3 illustrates the self-attention weights α_i^S of three examples from heterographic puns in the SemEval dataset. The word highlighted in the upper sentence (marked in pink) is a pun while we also color each word of lower sentence in blue according to the magnitude of its attention weights. Note that deeper colors indicate higher attention weights. In the first example, “busy” has the largest weight because it has the most similar semantic meaning as “harried”. “The barber” also has relatively high weights. We suppose it is related to “hairy” which should be the other word of this double entendre. Similar, “the zoo” is corresponded to “lion” while “phone” and “busy” indicate “line” for the pun. Moreover, “boating” confirms “sail” while “store” supports “sale”. Interpreting the weights out of our self-attentive encoder explains the significance of each token when the model is detecting the pun in the context. The pronunciation is essential in these cases because it strengthens the relationship

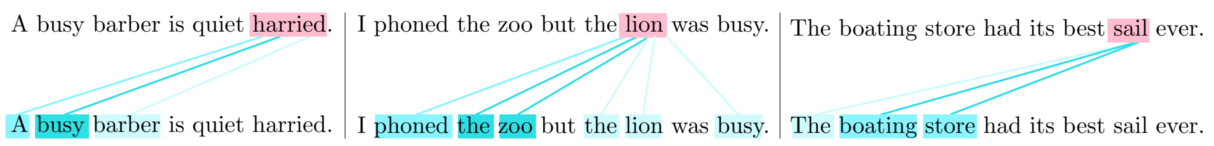
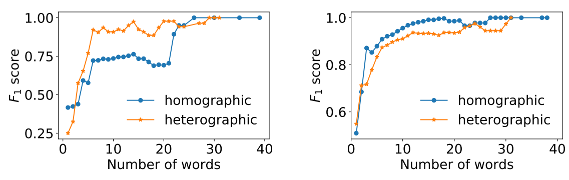

A busy barber is quiet harried.
A busy barber is quiet harried.
I phoned the zoo but the lion was busy.
I phoned the zoo but the lion was busy.
The boating store had its best sail ever.
The boating store had its best sail ever.

Figure 3: Visualization of attention weights of each word for the pun word (marked in pink) in the sentences. A deeper color indicates a higher attention weight.

Sentence	Pun	CPR	PCPR
Giving praise is when you let off esteem.	esteem	-	esteem
He stole an invention and then told patent lies.	patent	patent	lies
A thief who stole a calendar got twelve months.	got	-	-

Table 6: A case study of the model predictions for the pun location task of SemEval 2017.



(a) Pun Detection (b) Pun Location

Figure 4: Pun recognition performance over different text lengths for homographic and heterographic puns in the SemEval dataset.

among words with distant semantic meanings but similar phonological expressions.

Sensitivity to Text Lengths. Figure 4 shows the performance of pun detection and location over different text lengths for homographic and heterographic puns in the SemEval dataset. For both tasks, the performance gets higher when the text lengths are longer because the context information also becomes more sufficient. The results also verify the importance of contexts for pun recognition.

Case Study and Error Analysis. Table 11 shows the results of a case study with the outputs of CPR and PCPR. In the first case, "let off esteem" is one common idiom so that the CPR does not recognize "esteem" as a pun word. However, PCPR is able to connect "esteem" with "steam" because they share several phonemes. More specifically, pronunciation features boost the performance of pun recognition under such conditions. However, the pronunciation features in some cases can mislead the model to make wrong predictions. For example, "patent" in the second sentence is a homographic pun word and has several meanings, which can be found with the contextual features. In addition, the phonemes in "lies" can be ubiquitous in many other words like "laws", thereby confusing the model. In the last case, "got" is a widely used causative with dozens of meanings so that the word is extremely hard to be recognized as a pun word with its contextual and phonological features.

3 Learning to Discriminate Perturbations for Blocking Adversarial Attacks in Text Classification

Recently, adversarial attacks against machine learning models have been more and more prevailing and threatened various real-world applications such as email spam filters and sentiment analysis. In this paper, we propose a novel framework, learning to discriminate perturbations (DISP), to discriminate and adjust malicious perturbations, thereby blocking adversarial attacks for text classification models. To identify adversarial attacks, a perturbation discriminator validates the likelihood to be perturbed for each token in the text and provides a set of potential perturbations. For each potential perturbation, an embedding estimator exploits context tokens to approximate the actual embeddings. Hierarchical navigable small world graphs are then applied to efficiently convert the embeddings to appropriate replacement tokens for approximate kNN search. Finally, DISP can block adversarial attacks for any NLP model without modifying the model structure or training procedure. Extensive experiments conducted on two benchmark datasets demonstrate that DISP significantly outperforms baseline methods in blocking adversarial attacks for text classification. In addition, in-depth analysis also shows the robustness of DISP across different situations.

3.1 Background

Deep learning techniques (Goodfellow et al., 2016) have achieved enormous success in many fields, such as computer vision and NLP. However, complex deep learning models are often sensitive and vulnerable to a tiny modification. In other words, malicious attackers can destroy the models by adding a few inconspicuous perturbations into input data, such as masking images with unrecognizable filters and making low-key modifications for texts. Therefore, adversarial attacks that crash prediction performance have attracted more and more research interests.

Existing studies on adversarial attacks can be classified into two groups, generation of adversarial examples and defense against adversarial attacks (Yuan et al., 2019). In the field of NLP, most of the existing studies focus on the former. For example, some studies (Ebrahimi et al., 2017; Alzantot et al., 2018) replace a word with synonyms or similar words while other works (Ebrahimi et al., 2017; Gao et al., 2018; Liang et al., 2017) conduct character-level manipulations to fool the models. However, to the best of our knowledge, none of them focuses on blocking adversarial attacks. Moreover, it is not straightforward to adapt existing approaches for blocking adversarial attacks, such as data augmentation (Krizhevsky et al., 2012) and adversarial training (Goodfellow et al., 2015), to NLP applications. Hence, the defense against adversarial attacks in NLP remains a challenging and unsolved problem.

Recognizing and removing the inconspicuous perturbations are the grail of defense against adversarial attacks. For instance, in computer vision, denoising auto-encoders (Warde-Farley and Bengio, 2017; Gu and Rigazio, 2015) are applied to remove the noises introduced by perturbations; Prakash et al. (2018) manipulate the images to make the trained models more robust to the perturbations; Samangouei et al. (2018) apply generative adversarial networks to generate perturbation-free images. However, all of these approaches are not applicable to the NLP tasks for the following two reasons. First, images consists of continuous pixels while texts are discrete tokens. As a result, a token can be replaced with another semantically similar token that drops the performance, so perturbations with natural looks cannot be easily recognized compared to previous approaches that capture unusual differences between the intensities of neighboring pixels. Second, sentences consist of words with an enormous vocabulary size, so it is intractable to enumerate all of the possible sentences. Therefore, existing defense approaches in computer vision that rely on pixel intensities cannot be directly exploited for the NLP tasks.

After recognizing the perturbed tokens, the naïve way to eliminate the perturbations for blocking adversarial attacks is to remove these perturbed tokens. However, this is insufficient for the defense against adversarial attacks in NLP because the removed tokens can enclose essential semantics so that removing them can drop performance. Therefore, it is essential to recover the removed tokens. To recover the removed tokens, a possible approach is to exploit neighboring tokens as context, which is the objective of masked language models (Devlin et al., 2018a). Nevertheless, training a satisfactory language model requires myriad and diverse training data, which is often unavailable. An inaccurate language model that incoherently patches missing tokens can further worsen the prediction performance. Therefore, recovering the tokens from discriminated perturbations is a difficult task for NLP because of the sparsity of text data.

In this paper, we propose *Learning to Discriminate Perturbations* (DISP), as a framework for blocking adversarial attacks in NLP. More specifically, we aim to defend the model against adversarial attacks without modifying the model structure and the training procedure. DISP consists of three components, perturbation discriminator, embedding estimator, and hierarchical navigable small world graphs. Given a perturbed testing data, the perturbation discriminator first identifies a set of perturbed tokens. For each perturbed token, the embedding estimator optimized with a corpus of token embeddings infers an embedding vector to represent its semantics. Finally, we conduct an efficient kNN search over hierarchical navigable small work graphs to translate each of the embedding vectors into appropriate token to replace the associated perturbed word.

3.2 DISP for Blocking Adversarial Attacks

In this section, we first formally define the goal of adversarial defense and then introduce the proposed framework DISP, learning to discriminate perturbations, for blocking adversarial attacks.

Problem Statement. Given an NLP model $F(X)$, where $X = \{t_1, \ldots, t_N\}$ is the input text of N tokens while t_i indicates the i-th token. A malicious attacker can add a few inconspicuous perturbations into the input text as an adversarial example X_a so that $F(X) \neq F(X_a)$ with unsatisfactory prediction performance. For example, a perturbation can be insertion or deletion of a character in a token while a token can be replaced with a synonym. In this paper, we aim to block adversarial attacks for general text classification models. More specifically, we seek to preserve the model performances by recovering original input text and universally improve the robustness of any text classification model.

3.2.1 Framework Overview

Figure **??** illustrates the overall schema of the proposed framework. DISP consists of three components, including a perturbation discriminator, an embedding estimator, and a token embedding corpus with the corresponding small world graphs G. In the training phase, DISP applies a clean training dataset D to train the perturbation discriminator so that it is capable of recognizing the perturbed tokens. The corpus of token embeddings C is then applied to train the embedding estimator to recover the removed tokens after establishing the small world graphs G of the embedding corpus. In the prediction phase, for each token in testing data, the perturbation discriminator predicts if the token is perturbed, and then derives a set of potential perturbations based on neighborhood tokens as contexts. For each potential perturbation, the embedding estimator generates an approximate embedding vector and retrieve the token with the closest distance in the embedding space for token recovery. Finally, the recovered testing data can be applied for prediction. Note that the prediction model can be any NLP model, and it is unnecessary to adopt the training data for perturbation discriminator and embedding estimator. Moreover, DISP is a conceptual framework for blocking adversarial attacks, so the model selection for discriminator and estimator can also be flexible.

3.2.2 Perturbation Discrimination

Perturbation Discriminator. The perturbation discriminator plays an important role to classify whether a token t_i in the input X_a is perturbed based on its neighboring tokens as contexts. Based on this concept, any text classification model

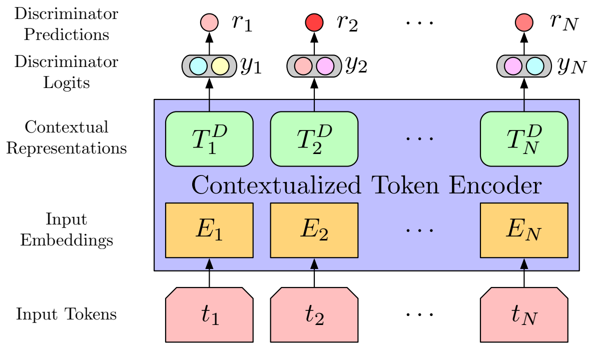


Figure 5: The illustration of the perturbation discriminator in DISP. Note that any contextualized token encoder is applicable to be the discriminator.

can be applied as a discriminator. In this paper, we adopt contextualized language modeling, such as BERT (Devlin et al., 2018a), to derive d-dimension contextualized token representation T_i^D for each token t_i so that the representations can also have the semantics of contexts. Finally, the discriminator predictions can be derived based on contextual representations.

Figure 5 illustrates the perturbation discriminator based on a contextualized word encoder. The discriminator classifies a token t_i into two classes $\{0, 1\}$ with logistic regression based on the contextual representation T_i^D to indicate if the token is perturbed. More formally, for each token t_i, the discriminator predictions r_i can then be derived as:

$$r_i = \operatorname{argmax} c \, y_i^c = \operatorname{argmax} c \left(\boldsymbol{w_c} \cdot T_i^D + b_c \right),$$

where y_i^c is the logit for the class c; $\boldsymbol{w_c}$ and b_c are the weights and the bias for the class c. Finally, the potential perturbations R is the set of tokens with positive discriminator predictions $R = \{t_i \mid r_i = 1\}$.

3.2.3 Efficient Token-level Recovery with Embedding Estimator

After predicting the perturbations R, we need to correct these disorders to preserve the prediction performance. One of the most intuitive approaches to recover tokens with context is to exploit language models. However, language models require sufficient training data while the precision to exact tokens can be dispensable for rescuing prediction performance. Moreover, over-fitting limited training data can be harmful to the prediction quality. To resolve this problem, we assume that most of the tokens with similar semantics are sufficient to recover the performance. Based on the assump-

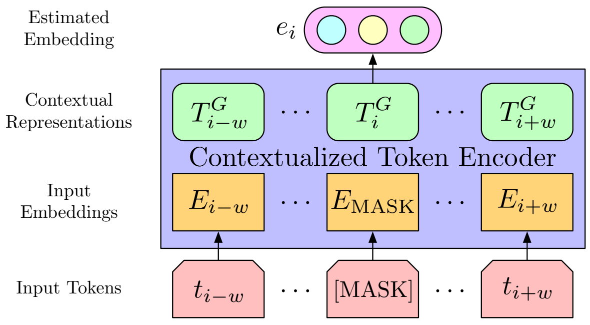


Figure 6: The illustration of the embedding estimator in DISP with a window size $2w + 1$ for the token at the position i. Note that any contextualized token encoder is applicable to be the estimator.

tion, we first generate approximate token embeddings for the potential perturbations with an embedding estimator based on context tokens. The tokens can then be appropriately recovered by an efficient k-nearest neighbors (kNN) search in the embedding space of a token embedding corpus C.

Embedding Estimator. Similar to the perturbation discriminator, any regression model can be employed as an embedding estimator based on the proposed concept. Here we adopt the contextualized language modeling again as an example of the embedding estimator. For each token t_i, the contextualized token embedding can be derived as a d-dimensional contextual representation vector T^G_i to be features for estimating appropriate embeddings.

Figure 6 shows the embedding estimator based on BERT. For each potential perturbation $t_i \in R$, $2w$ neighboring tokens are selected as the context for estimating the appropriate embedding. More precisely, a segment of tokens with a window size $2w + 1$ from t_{i-w} to t_{i+w} is the input tokens for BERT, where t_i is replaced with a `[MASK]` token as the perturbed position. Finally, for the target t_i, a weight matrix $W^G \in^{d\times k}$ projects the contextual representation T^G_i to a k-dimensional estimated embedding e_i as follows:

$$e_i = T^G_i W^G,$$

where the dimension size k is required to be consistent with the embedding dimension in the token embedding corpus C.

Efficient Token-level Recovery. To universally benefit all NLP models, embedding-level recovery is inadequate. Specifically, the input text X needs to be recovered from the perturbed text X_a by fixing token-level perturbations based on its approximate embeddings.

Algorithm 1: Efficient Token-level Recovery

```
Input: Perturbed text $X_a$, potential perturbations $R$,
       estimated embeddings $\{e_i\}$, small world
       graphs $G$, token embedding corpus $C$.
Output: Recovered text $X_r$.
1 $X_r = X_a$;
2 for $t_i \in R$ do
3     index = QuerySmallWorldGraph($G$, $e_i$);
4     $z = C$[index].token;
5     Replace $t_i$ in $X_r$ with $z$;
6 return $X_r$;
```

Given the token embedding corpus C, it is simple to transform an embedding to a token by finding the nearest neighbor token in the embedding space. However, a naïve kNN search query can take $O(kn)$ time complexity, where n is the number of embeddings in C; k is the embedding dimension. To accelerate the search process, we apply hierarchical navigable small world graphs (SWGs) (Malkov and Yashunin, 2018) for fast approximate kNN search. More precisely, embeddings are transformed into a hierarchical set of SWGs based on the proximity between different embeddings. To conduct kNN searches, the property of degree distributions in SWGs significantly reduces the search space of each kNN query from $O(n)$ to $O(\log n)$ by navigating on the graphs, so a kNN query can be efficiently completed in $O(k \log n)$ time complexity. Finally, the recovered text X_r can be obtained by replacing the perturbations R in X_a as shown in Algorithm 1.

3.2.4 Learning and Optimization

To learn a robust discriminator, adversarial learning (Goodfellow et al., 2015) is applied to generate different adversarial samples for training. More specifically, we randomly sample adversarial examples from both character-level and word-level attacks in each training epoch. The loss function optimizes the cross-entropy (Hinton and Salakhutdinov, 2006) between the labels and the probabilistic scores computed by the logits y_i and the softmax function (Goodfellow et al., 2016).

The embedding estimator is similar to masked language models in learning process. The major difference between two models is that language models optimize the likelihood to generate exactly the same original token while the embedding estimator minimizes the distance between the de-

Dataset	Train	Test	Length		
			Max.	Min.	Avg.
SST-2	67,349	1,821	56	1	19
IMDb	25,000	25,000	2,738	8	262

Table 7: The statistics of two experimental datasets.

rived embedding and the original token embedding. To learn the embedding estimator, a size-$(2w + 1)$ sliding window is applied to enumerate $(2w+1)$-gram training data for approximating embeddings with context tokens. For optimization, mean square error (MSE) is adopted to minimize the Euclidean distances from the inferred embeddings to the original token embeddings.

To take the advantage of hierarchical navigable SWGs for efficient recovery, although a preprocess to construct SWGs G is required, the preprocess can be fast. The established SWGs can also be serialized in advance. More precisely, the time complexity is $O(kn \log n)$ for one-time construction of reusable SWGs, where n is the number of embeddings in the embedding corpus C.

3.3 Experiments

In this section, we conduct extensive experiments to evalute the performance of DISP in improving model robustness with two publicly available benchmark datasets in text classification.

3.3.1 Experimental Settings

Experimental Datasets. Experiments are conducted on two publicly available benchmark datasets used by state-of-the-art text classification tasks (Howard and Ruder, 2018; Radford et al., 2017) with varying numbers of documents and sentence lengths, including Stanford Sentiment Treebank Binary (SST-2) (Socher et al., 2013) and Internet Movie Database (IMDb) (Maas et al., 2011). SST-2 and IMDb are both sentiment classification datasets which involve binary labels to describe a sentence from a movie review. Detailed statistics of two datasets are listed in Table 7.

Attack Generation. In the experiments, we focus on three types of character-level attacks and two types of word-level attacks. The character-level attacks consist of *insertion*, *deletion*, and *swap*. *Insertion* and *deletion* attacks inject and remove a character, respectively, while a *swap* attack flips two adjacent characters. The word-level attacks include *random* and *embed*. A *random* attack randomly samples a word to replace the target word while a *embed* attack replaces the word with a similar word among the top-10 nearest words in the embedding space. To mimic strong adversarial attacks, for each testing sample, we randomly generate 50 adversarial examples and sample one example that reduces performance as testing data. If none of them can lower the performance, the sample with the least confidence is selected.

Base Model and Baselines. We choose BERT (Devlin et al., 2018a) as the base model to be attacked because it is one of the state-of-the-art approaches. To evaluate the performance of DISP, we consider the following the baseline methods: (1) Adversarial Data Augmentation (ADA) samples adversarial examples to increase the diversity of training data while (2) Adversarial Training (AT) samples different adversarial examples in each training epoch. (3) Spelling Correction (SC) can be also used as a special baseline for discriminating perturbations and blocking character-level attacks. Note that only ADA and AT will re-train BERT with the updated training data. DISP and SC modify the input text and then exploit the original model for prediction. SC is also the only baseline for evaluating the discriminator performance. In addition, we also try to make an ensemble of DISP and SC (DISP+SC) by conducting DISP on the spelling corrected input.

Evaluation Metrics. For the perturbation discriminator, we adopt precision, recall, and F1-score as evaluation metrics (Manning et al., 2010). For the performance of blocking adversarial attacks, we evaluate methods with classification accuracy.

Implementation Details. The model is implemented by PyTorch (Paszke et al., 2017). We adopt Adam optimizer (Kingma and Ba, 2014) to optimize the parameters and perform back-propagation algorithm based on gradients. The initial learning and dropout parameter are set as 2×10^{-5} and 0.1. We use crawl-300d-2M word embeddings from *fastText* (Mikolov et al., 2018) to search similar words. The dimensions of word embedding k and contextual representation d are set as 300 and 768. w is set as 2 to implement the sliding window. We follow BERT$_{\text{BASE}}$ (Devlin et al., 2018a) to set the numbers of layers (i.e., Transformer blocks) and self-attention heads as 12.

3.3.2 Experimental Results

Discrimination Performance. Table 8 shows the performance of DISP and SC in discriminating

Dataset	Method	Metric	Character-level Attacks			Word-level Attacks		Overall
			Insertion	Deletion	Swap	Random	Embed	Attacks
SST-2	SC	Precision	0.5087	0.4703	0.5044	0.1612	0.1484	0.3586
		Recall	0.9369	0.8085	0.9151	0.1732	0.1617	0.5991
		F1	0.6594	0.5947	0.6504	0.1669	0.1548	0.4452
	DISP	Precision	0.9725	0.9065	0.9552	0.8407	0.4828	0.8315
		Recall	0.8865	0.8760	0.8680	0.6504	0.5515	0.7665
		F1	**0.9275**	**0.8910**	**0.9095**	**0.7334**	**0.5149**	**0.7952**
IMDb	SC	Precision	0.0429	0.0369	0.0406	0.0084	0.0064	0.0270
		Recall	0.9367	0.8052	0.8895	0.1790	0.1352	0.5891
		F1	0.0820	0.0706	0.0777	0.0161	0.0122	0.0517
	DISP	Precision	0.9150	0.8181	0.8860	0.5233	0.2024	0.6690
		Recall	0.5068	0.4886	0.5000	0.3876	0.2063	0.4179
		F1	**0.6523**	**0.6118**	**0.6392**	**0.4454**	**0.2044**	**0.5106**

Table 8: The performance of two methods in discriminating perturbations.

Dataset	Method	Attack	Character-level Attacks			Word-level Attacks		Overall
		-free	Insertion	Deletion	Swap	Random	Embed	Attacks
SST-2	BERT	**0.9232**	0.6498	0.6544	0.6774	0.5385	0.6556	0.6351
	SC	0.9174	**0.9082**	0.8186	**0.8840**	0.5993	0.7003	0.7821
	ADA	0.9174	0.8071	0.8071	0.8209	0.7394	0.7681	0.7885
	AT	0.9186	0.8186	0.8175	0.8025	0.6935	0.7646	0.7793
	DISP	**0.9232**	0.8278	**0.8278**	0.8301	**0.7773**	**0.7784**	**0.8083**
	DISP+SC	0.9197	**0.9128**	**0.8681**	**0.9060**	**0.7784**	**0.7853**	**0.8501**
IMDb	BERT	**0.9431**	0.8586	0.8599	0.8568	0.8468	0.8615	0.8567
	SC	0.9193	0.8834	0.8794	0.8825	0.8695	0.8753	0.8780
	ADA	0.9393	0.8766	0.8765	0.8754	0.8722	0.8755	0.8752
	AT	0.8998	0.8958	0.8822	0.8787	0.8886	0.8822	0.8855
	DISP	0.9378	**0.9310**	**0.9297**	**0.9301**	**0.9281**	**0.9347**	**0.9307**
	DISP+SC	0.9395	**0.9316**	0.8772	**0.9313**	0.8755	0.9292	0.9090

Table 9: The accuracy scores of methods with different adversarial attacks on two datasets.

perturbations. Compared to SC, DISP has an absolute improvement by 35% and 46% on SST-2 and IMDb in terms of F1-score, respectively. It also proves that the context information is essential when discriminating the perturbations. An interesting observation is that SC has high recall but low precision scores for character-level attacks because it is eager to correct misspellings while most of its corrections are not perturbations. Conversely, DISP has more balanced recall and precision scores since it is optimized to discriminate the perturbed tokens. For the word-level attacks, SC shows similar low performance on both *random* and *embed* attacks while DISP behaves much better. Moreover, DISP works better on the *random* attack because the embeddings of the original tokens tend to have noticeably greater Euclidean distances to randomly-picked tokens than the distances to other tokens.

Defense Performance. Table 9 reports the accuracy scores of all methods with different types of adversarial attacks on two datasets. Compared to the base model BERT, all of the methods alleviate the performance drops to different extents. The performance of all methods for blocking character-level attacks are generally better than the performance with word-level attacks because word-level attacks eliminate more information. For the baselines, consistent with Table 8, SC performs the best for character-level attacks and the worst for word-level attacks. In contrast, ADA and AT are comparably more stable across different types of attacks. The differences between performance for character- and word-level attacks

Method	Insertion	Delete	Swap
BERT	0.6498	0.6544	0.6774
DISP$_{\text{SST-2}}$	0.8278	0.8278	0.8301
DISP$_{\text{IMDb}}$	0.8243	0.8197	0.8278
Method	**Random**	**Embed**	**Overall**
BERT	0.5385	0.6556	0.6351
DISP$_{\text{SST-2}}$	0.7773	0.7784	0.8083
DISP$_{\text{IMDb}}$	0.7623	0.7681	0.8005

Table 10: The accuracy of DISP over different types of attacks on the SST-2 dataset with the tokens recovered by the perturbation discriminator and the embedding estimator trained on the IMDb dataset for robust transfer defense. Note that DISP$_x$ indicates the framework is established on the dataset x.

are less obvious in IMDb because documents in IMDb tend to be longer with more contexts to support the models. DISP works well to block all types of attacks. Compared with the best baseline models, DISP significantly improves the classification accuracy by 2.51% and 5.10% for SST-2 and IMDb, respectively. By making an ensemble of SC and DISP, DISP+SC achieves a better defense performance for blocking all types of attacks. However, the improvements are not consistent in IMDb because SC performs worse with lower discrimination accuracy and over-correcting the documents. In addition, DISP has a stable defense performance across different types of attacks on IMDb because richer context information in the documents benefits token recovery.

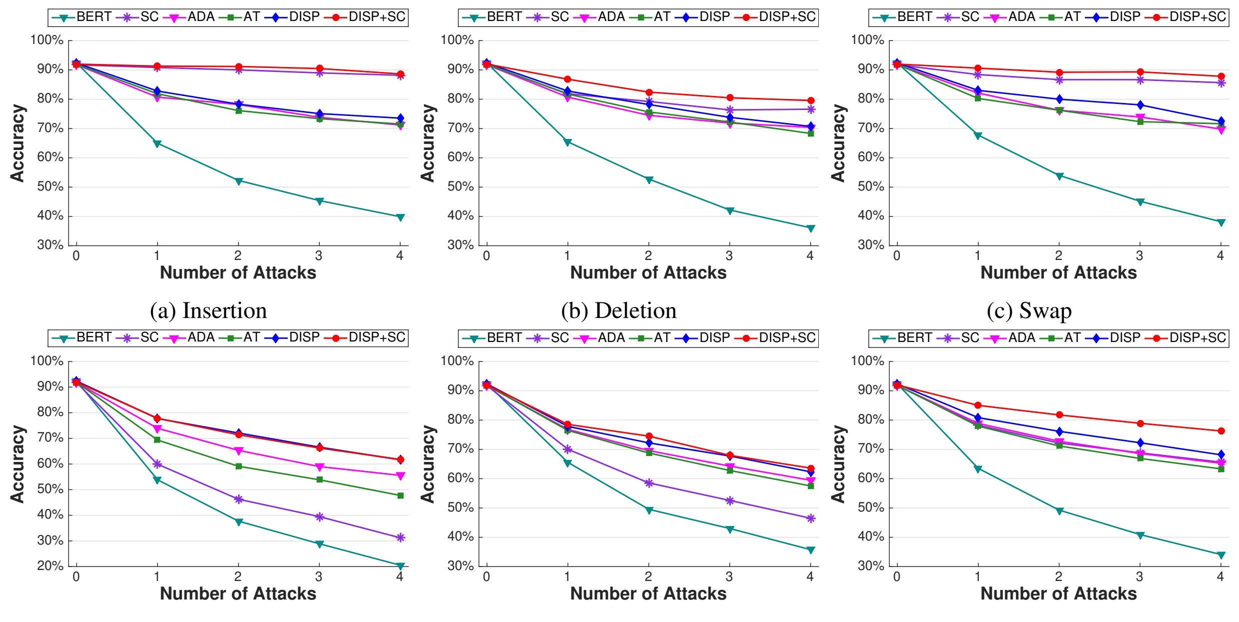
Number of Attacks. Figure 7 shows the classification accuracy of all methods over different numbers of attacks, i.e., perturbations, for different types of adversarial attacks. Without using a defense method, the performance of BERT dramatically decreases with more attacks when other defense approaches have gentler trends of performance drops. It further emphasizes the importance of defense methods for blocking adversarial attacks. Moreover, the relations between performance of methods are consistent across different perturbation numbers. DISP+SC consistently performs the best for all of the cases when DISP outperforms all of the single methods for most of the situations. These results demonstrate the robustness of our proposed approach.

Robust Transfer Defense. Since the perturbation discriminator and embedding estimator are not necessary to be trained on the same data to the prediction model, here we conduct the experiments of robust transfer defense to verify the robustness of DISP. We first train the discriminator and the estimator on IMDb denoted as DISP$_{\text{IMDb}}$ and then apply the framework to defend the prediction model on SST-2. Table 10 shows the experimental results of robust transfer defense. DISP$_{\text{IMDb}}$ achieve similar performance as the performance of DISP$_{\text{SST-2}}$ trained on the same training set. Hence, it shows that DISP can manage robust transfer defense and do not rely on any specific training corpus.

Case Study of Recovered Text. Table 11 lists four documents from SST-2 for a case study. We successfully recovered the attacked words from "orignal" and "bet" in the cases 1 and 2 to "imaginative" and "best". It demonstrates that embeddings generated by the embedding estimator are robust to recover the appropriate tokens and block adversarial attacks. However, DISP performs worse when the remaining sentence is lack of informative contexts as case 3. When multiple attacks exist, the incorrect context may also lead to unsatisfactory recoveries, e.g., DISP converts "funny" to "silly" in case 4, thus flipping the prediction. This experiment depicts a disadvantage of DISP and demonstrates that DISP+SC can gain further improvements.

4 Clinical named entity recognition using contextualized token representations

The clinical named entity recognition (CNER) task seeks to locate and classify clinical terminologies into predefined categories, such as diagnostic procedure, disease/disorder, severity, medication, medication/dosage, and sign/symptom. Existing approaches in extracting the entities leverage static word embeddings to represent each word. Such embeddings are insufficient to integrate the diverse interpretation of a word since the meaning of a word depends on the context of a sentence. To overcome this challenge, contextualized word embedding has been introduced to better capture the semantic meaning of each word based on its context. Two of these language models, ELMo and Flair, have been widely used in the field of Natural Language Processing to generate the contextualized word embeddings on domain-generic documents. However, these embeddings are usually too general to capture the proximity among vocabularies of specific domains. To facilitate various downstream applications using clin-



(a) Insertion (b) Deletion (c) Swap

(d) Random (e) Embed (f) Overall

Figure 7: The accuracy of methods over different numbers and types of attacks.

#	Attacked Sentence	Recovered Token	Label	Pred
1	Mr. Tsai is a very **orig~~i~~nal** artist in his medium, and what time is it there?	imaginative	1	1
2	Old-form moviemaking is at its **be~~s~~t**.	best	1	1
3	My reaction in a word: **disapponitment**.	that	0	1
4	a **painfulily funtny** ode to **gbad** behavior.	painfully; silly; one	1	0

Table 11: A case study of recovered tokens in SST-2. Note that Label and Pred represent the ground-truth label and the predicted label.

ical case reports (CCRs), we pre-train two deep contextualized language models, Clinical Embeddings from Language Model (C-ELMo) and Clinical Contextual String Embeddings (C-Flair) using the clinical-related corpus from the PubMed Central. Explicit experiments show that our models gain dramatic improvements compared to the state-of-the-art baseline models.

4.1 Background

Clinical case reports (CCRs) are written descriptions of the unique aspects of a particular clinical case (Cabán-Martinez and García-Beltrán, 2012). They are intended to serve as educational aids to science and medicine, as they play an essential role in sharing clinical experiences about atypical disease phenotypes and new therapies (Caufield et al., 2018). Unlike other types of clinical documents (e.g., electronic medical records, or EMRs), CCRs generally describe single clinical narratives at a time: these are stories of diseases as they were observed and treated, written in language requiring domain familiarity to fully interpret. Conveniently, accessing and reading any of the more than 2 million CCRs in publication does not involve the privacy responsibilities required by EMRs and other protected health information. CCRs therefore serve as rich, plentiful examples of clinical language.

Clinical named entity recognition (CNER) is an important text mining task in the domain of biomedical natural language processing. It aims to identify clinical entities and events from the text within documents such as CCRs (Zhang and Elhadad, 2013). For example, in the sentence "CT of the maxillofacial area showed no facial bone fracture." "CT of the maxillofacial area" is a "Test" and "facial bone fracture" belongs to the "Problem".As with documents describing experimental procedures and results—often the focus of general biomedical annotated corpora such as PubTator (Wei et al., 2013)—CCRs include a large variety of entity types and potential orders of events (Caufield et al., 2018). Methods to better enable biomedical and clinical NLP at scale, across numerous entity types, and with generalizable approaches across topics are necessary, as single-task or single-entity type methods provide insufficient detail for comprehensive CNER. Fine-grained CNER supports development of precision medicine's hope to leverage advanced computer technologies to deeply digitize, curate and understand medical records and case reports (Rajkomar et al., 2018; Bates et al., 2014).

Biomedical NER (BioNER), of which CNER is a subtask, has been a focus of intense, groundbreaking research for decades but has recently undergone a methodological shift. Its foundational methods are largely rule-based (e.g., Text Detective (Tamames, 2005)), dictionary-based (e.g., BioThesaurus (Liu et al., 2006) or MetaMap (Aronson, 2001)), and basic statistical approaches (e.g., the C-value / NC-value method (Frantzi et al., 2000)). Source entities for NER are sourced from extensive knowledgebases such as UMLS (Bodenreider, 2004) and UniProtKB (The UniProt Consortium, 2017). Readily applicable model-based BioNER methods, including those built upon non-contextualized word embeddings such as Word2Vec and GloVe (Mikolov et al., 2013b; Pennington et al., 2014b) now promise to more fully address the challenges particular to the biomedical domain: concepts may have numerous names, abbreviated forms, modifiers, and variants. Furthermore, biomedical and clinical text assumes readers have extensive domain knowledge. Its documents follow no single structure across sources or topics, rendering their content difficult to predict. For example, the context can thoroughly change an individual word's meaning, e.g., an "infarction" in the heart is a heart attack but the same event in the brain constitutes a stroke. Context is crucial for understanding abbreviations as well: "MR" may represent the medical imaging technique *magnetic resonance*, the heart condition *mitral regurgitation*, the concept of a *medical record*, or simply the honorific *Mister*. Non-contextualized word embeddings exacerbate the challenge of understanding distinct biomedical meanings as they contain only one representation per word. The most frequent semantic meaning within the training corpus becomes the standard representation.

Inspired by the recent development of contextualized token representations (Peters et al., 2018c; Devlin et al., 2018b; Akbik et al., 2018) supporting identification of how the meaning of words changes based on surrounding context, we refresh the technology of CNER to better extract clinical entities from unstructured clinical text. The deep contextualized token representations are pretrained with a large corpus using a language model (LM) objective. ELMo (Peters et al., 2018c) takes word tokens as input and pre-trains them with a bidirectional language model (biLM). Flair (Akbik et al., 2018) proposes a pre-trained character-level language model by passing sentences as sequences of characters into a bidirectional LSTM to generate word-level embeddings. BERT (Devlin et al., 2018b) is built with bidirectional multi-layered Transformer encoders on top of the WordPiece embeddings, position embeddings, and segment embeddings. In this paper, we address the CNER task with contextualized embeddings (i.e., starting with ELMo and Flair), then and compare structural differences in the resulting models. Following recent work demonstrating impressive performance and accuracy of pre-training word representations with domain-specific documents (Sheikhshabbafghi et al., 2018), we collected domain-specific documents all related to CCRs, roughly a thousandth of all PubMed Central (PMC) documents, and used them to pre-train two deep language models, C-ELMo and C-Flair. In this paper, we focus on the CNER task and evaluate the two language models across three datasets. Our two pre-trained language models can support applications beyond CNER, such as clinical relation extraction or question answering.

4.2 Method

In this subsection, we first introduce the architectures of both word-level and character-level language models in Section 4.2.1. We then explain our CNER model in Section 4.2.2.

4.2.1 Contextualized Language Models

ELMo is a language model that produces contextualized embeddings for words. It is pre-trained with a two-layered bidirectional language model (biLM) with character convolutions on a large corpus. The left lower part in Figure 8 is the high level architecture of ELMo, where $\mathrm{R}(\cdot)$ means the representation of a word. ELMo feeds a sequence of tokens $(t_1, t_2, ..., t_N)$ into the biLM which is a bidirectional Recurrent Neural Network (RNN). The backward-LM has the same structure as the forward-LM, except the input is the reverse sequence. Then, we jointly maximize the log-likelihood of both directions:

$$\sum_{k=1}^{N}\big(\log p(t_k|t_1,t_2,...,t_{k-1};\theta_x,\theta_f,\theta_s) \\ + \log p(t_k|t_{k+1},t_{k+2},...,t_N;\theta_x,\theta_b,\theta_s)\ \big) \tag{1}$$

where θ_x is the token representation and θ_s is the Softmax layer for both the forward and backward LM's, and θ_f and θ_b denotes the parameters of RNNs in two directions.

Flair is a character-level word representation model that also uses RNN as the language modeling structure. Different from ELMo, Flair treats the text as a sequence of characters.

The goal of most language models is to estimate a good distribution $p(t_0, t_2, ..., t_T)$ where $t_0, t_1, ..., t_n$ is a sequence of words. Instead of computing the distribution of words, Flair aims to estimate the probability $p(x_0, x_1, ...x_T)$, where $x_0, x_1, ..., x_T$ is a sequence of characters. The joint distribution over the entire sentence can then be represented as follows:

$$p(x_0,x_1,...,x_T) = \prod_{t=0}^{T} p(x_t|x_1,x_2,...,x_{t-1}) \tag{2}$$

where $p(x_t|x_0, ..., x_{t-1})$ is approximated by the network output h_t from one RNN layer.

$$p(x_t|x_0,...,x_{t-1}) = \prod_{t=0}^{T} p(x_t|h_t;\theta) \tag{3}$$

h_t is the hidden state that records the entire history of the sequence, which is computed recursively with a memory cell. θ denotes all the parameters of the RNN model.

4.2.2 CNER Model

We used a well-established BiLSTM-CRF sequence tagging model (Huang et al., 2015; Wang et al., 2018; Habibi et al., 2017) to address the downstream sequence labeling tasks.

First, it passes sentences to a user-defined token embedding model, which converts a sequence of tokens into word embeddings: $r_0, r_1, r_2, ..., r_n$. We may concatenate embedding vectors from different sources to form a new word vector. For example, the concatenated embeddings of GloVe and Flair is represented as:

$$r_i = r_i^{GloVe} \oplus r_i^{Flair} \tag{4}$$

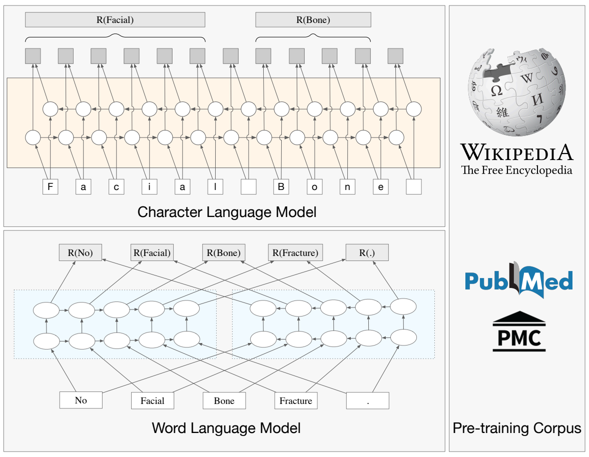


Figure 8: Character and Word Language models

Then, the concatenated embeddings are passed to the BiLSTM-CRF sequence labeling model to extract the entity types.

4.3 Experiments

4.3.1 Datasets and Corpus

MACCROBAT2018 is a subset of the MACCR data set developed by Caufield et al. (2018): a set of text corresponding to medical concepts among 3,100 curated CCRs, 15 disease groups, and more than 750 reports of rare diseases. The MACCROBAT2018 subset contains 200 reports deeply annotated by clinical domain experts following our ACROBAT typing scheme's 24 different entity/event types. The set contains 3,652 sentences in total. We randomly selected 10% case reports as the development set and 10% as the test set. The remaining documents are used to train the CNER model. A detailed description is shown in Table 12.

i2b2-2010 provides linguistic annotations over a set of clinical notes. In this study, we focus on the first task: given the plain text, we extract the clinical entities. The dataset contains three entity types which are "test", "problem", "treatment". We followed Uzuner et al. (2011) to split the dataset into train/development/test sets.

NCBI-disease is fully annotated at the mention and concept level to serve as a research resource for the biomedical natural language processing community (Doğan et al., 2014). The dataset contains 793 PubMed abstracts with 6,892 disease mentions which leads to 790 unique disease concepts. Therefore, the dataset only has one types which is "disease".

Pre-training Corpus To pre-train the two lan-

Table 12: Number of sentences and tokens in each CNER dataset

Dataset	# of entity types	# of sentences		
		Train	Dev	Test
MACCROBAT2018	24	2,894	380	351
i2b2-2010	3	14,683	1,632	27,626
NCBI-disease	1	5,423	922	939

guage models, we obtained articles through the PubMed Central (PMC) FTP server[4], and in total picked 47,990 documents that are related to clinical case reports. We indexed these documents with some keyword including "case report" and "clinical report". This corpus contains 0.1 billion words which is around 1/10 of the corpus used for the domain-generic ELMo (Peters et al., 2018c) and Flair (Akbik et al., 2018).

4.3.2 Pre-trained Language Model

To fairly compare the two models, we do not initialize C-ELMo and C-Flair with any pre-trained language models, and pre-train them on the same clinical corpus. Moreover, we tried to set both models' parameter sizes to a similar scale. C-Flair's parameter size is 20M. We chose the medium-size C-ELMo model correspondingly, which has 25M parameters (Peters et al., 2018c). All models were pre-trained on one NVIDIA Tesla V100 (16GB), with each requiring roughly one week to complete. For C-Flair, we followed the default settings of Flair, a hidden size of 2048, a sequence length of 250, and a mini-batch size of 100. The initial learning rate is 20, and the annealing factor is 4. For C-ELMo, we chose the medium-size model among all configurations, which has a hidden size of 2048 and projection dimension of 256.

4.3.3 Results

The results of our experiments are shown in Table 13. Note that "Embeddings" denotes individual embedding or the concatenation of different embedding vectors. We used the pre-trained GloVe embeddings of 100 dimensions [5]. The Flair embeddings are pre-trained with a 1-billion word corpus (Chelba et al., 2013). ELMo denotes the pre-trained medium-size ELMo on the same 1-billion word corpus and ELMoPubMed denotes the pre-trained ELMo model with the full PubMed and PMC corpus [6]. We used the micro F1-score as the evaluation metric.

Domain-specific v.s. Domain-generic corpus. From Table 13, we can observe that the models pre-trained on the selected case report corpus outperformed all other language models pre-trained on the domain-generic corpus. The concatenated embedding of GloVe and C-ELMo performs the best on MACCROBAT2018 and NCBI-disease datasets, while GloVe plus C-Flair achieved the best performance on i2b2-2010. We can conclude that pre-training the language models with a small domain-specific corpus can be more efficient and effective for some downstream tasks. Domain-specific knowledge can alter the distribution and proximity among words, thus contributing a better understanding of the relationship between word and entity types in this task.

Contextualized v.s. Non-contextualized embeddings. We also used the GloVe static word embeddings to represent tokens in the sequence labeling task. The results in Table 13 show that the concatenated contextualized embeddings dramatically boosted F1-scores on the three different datasets by 10.31%, 7.50%, and 6.94%. We suppose that richer semantic and syntactic knowledge could be inferred from the available context. We noticed that the F1-score of Flair on the MACCROBAT2018 dataset was surprisingly low, indicating that a purely character-level language model may not be as robust as the word-level models.

Compared with other baseline models. Ma and Hovy (2016) proposed a bi-directional LSTM-CNNs-CRF model.Wang et al. (2018) leveraged multi-task learning and attention mechanisms to improve the performance.Compared with these two state-of-the-art models which both use static word embeddings, our methods perform consistently better as shown in Table 14. We suppose that with the pre-trained contextualized embeddings, even a light-loaded downstream model

[4] ftp://ftp.ncbi.nlm.nih.gov/pub/pmc
[5] https://nlp.stanford.edu/projects/glove/
[6] https://allennlp.org/elmo/

Table 13: The comparison of F1-scores (%) on three datasets among different types of embeddings

Embeddings	MACCROBAT2018	i2b2-2010	NCBI-disease
GloVe	59.63	81.35	82.18
GloVe+ELMo	63.09	84.82	85.37
GloVe+Flair	62.63	81.21	85.58
GloVe+ELMoPubMed	64.56	86.50	87.04
GloVe+C-ELMo	**65.75**	87.29	**87.88**
GloVe+C-Flair	64.18	**87.45**	86.60

Table 14: The performance of three baseline methods and our best model on three datasets. Our models only leverage a simple LSTM-CRF sequence labeling module with the pre-trained contextualized embeddings.

Models	MACCROBAT2018	i2b2-2010	NCBI-disease
Ma and Hovy (2016)	60.13	81.41	82.62
Wang et al. (2018)	63.10	84.97	86.14
Lee et al. (2019)	64.38	86.46	**89.36**
Our best model	**65.75**	**87.45**	87.88

can achieve extraordinary performances. The BioBERT (Lee et al., 2019) was pre-trained using a contextualized language model with around 110M parameters and consuming a large number of computational resources (8 NVIDIA V100 32GB GPUs). However, it only gets better performance in the simplest dataset (NCBI-disease) with one entity type. On MACCROBAT2018 and i2b2-2010, we improved the performance by 2.13% and 1.15%, compared to (Lee et al., 2019). This shows that better experimental results can be achieved with limited resources.

Table 15: The comparison of F1-scores (%) between C-ELMo and C-Flair on different entity types of MACCROBAT2018

Entity	GloVe+C-ELMo	GloVe+C-Flair
Biological Structure	63.94	**64.88**
Detailed Description	**45.81**	40.00
Diagnostic Procedure	**74.93**	74.71
Disease Disorder	**50.84**	50.83
Dosage	77.42	**80.00**
Medication	**76.34**	72.13
Severity	**72.41**	61.81
Sign Symptom	**62.27**	60.64

4.3.4 Case Study and Analysis

We analyze C-Flair and C-ELMo on specific categories for the MACCROBAT2018 dataset and obtain F1-scores for 8 different entity types. All these types appear more than 50 times in the dataset. Table 15 shows that the character-level language model C-Flair takes an advantage in the type "Dosage". This entity type has a number of entities that do not appear in the word-level vocabulary, such as "60 mg/m2", "0.5 mg", and "3g/d". On the other hand, C-ELMo has a better performance in the type "Severity", which contains words like "extensive", "complete", "significant", and "evident". C-ELMo also extensively outperforms C-Flair in "Detailed Description". The representations of tokens rely more on the word-level context in these types, indicating that C-ELMo can better encode the relationship between the word-level contextual features with the entity types.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

A. R. Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. *Proc AMIA Symp*, pages 17–21.

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

David W Bates, Suchi Saria, Lucila Ohno-Machado, Anand Shah, and Gabriel Escobar. 2014. Big data in health care: using analytics to identify and manage high-risk and high-cost patients. *Health Affairs*, 33(7):1123–1131.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Research*, 32(90001):D267–70.

Alberto J Cabán-Martinez and Wilfredo F García-Beltrán. 2012. Advancing medicine one research note at a time: the educational value in clinical case reports. *BMC Research Notes*, 5(1):293.

Yitao Cai, Yin Li, and Xiaojun Wan. 2018. Sense-aware neural models for pun location in texts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 546–551.

J Harry Caufield, Yijiang Zhou, Anders O Garlid, Shaun P Setty, David A Liem, Quan Cao, Jessica M Lee, Sanjana Murali, Sarah Spendlove, Wei Wang, et al. 2018. A reference set of curated biomedical data and metadata from clinical case reports. *Scientific data*, 5:180258.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Lei Chen and Chong MIn Lee. 2017. Predicting audience's laughter using convolutional neural network. *arXiv preprint arXiv:1702.02584*.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 103–108.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus. *J. of Biomedical Informatics*, 47(C).

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

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

Shixiang Gu and Luca Rigazio. 2015. Towards deep neural network architectures robust to adversarial examples. In *ICLR*.

Ashim Gupta, Pawan Goyal, Sudeshna Sarkar, and Mahanandeeshwar Gattu. 2019. Fully contextualized biomedical ner. In *European Conference on Information Retrieval*, pages 117–124. Springer.

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *North American Association for Computational Linguistics (NAACL)*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Lluís-F Hurtado, Encarna Segarra, Ferran Pla, Pascual Carrasco, and José-Angel González. 2017. Elirf-upv at semeval-2017 task 7: Pun detection and interpretation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 440–443.

Ayyoob Imani, Amir Vakili, Ali Montazer, and Azadeh Shakery. 2019. Deep neural networks for query expansion using word embeddings. In *European Conference on Information Retrieval*, pages 203–210. Springer.

Vijayasaradhi Indurthi and Subba Reddy Oota. 2017. Fermi at semeval-2017 task 7: Detection and interpretation of homographic puns in english language. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 457–460.

Aaron Jaech, Rik Koncel-Kedziorski, and Mari Ostendorf. 2016. Phonological pun-derstanding. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 654–663.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

H. Liu, Z.-Z. Hu, M. Torii, C. Wu, and C. Friedman. 2006. Quantitative assessment of dictionary-based protein named entity tagging. *Journal of the American Medical Informatics Association*, 13(5):497–507.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

Elena Mikhalkova and Yuri Karyakin. 2017. Punfields at semeval-2017 task 7: Employing roget's thesaurus in automatic pun recognition and interpretation. *arXiv preprint arXiv:1707.05479*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):10.

Dieke Oele and Kilian Evang. 2017. Buzzsaw at semeval-2017 task 7: Global vs. local context for interpreting and locating homographic english puns with sense embeddings. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 444–448.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Ted Pedersen. 2017. Duluth at semeval-2017 task 7: Puns upon a midnight dreary, lexical semantics for the weak and weary. *arXiv preprint arXiv:1704.08388*.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014b. Glove: Global vectors for word representation. In *In EMNLP*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *NAACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018c. Deep contextualized word representations. *CoRR*, abs/1802.05365.

David Martin Powers. 2011. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. 2018. Deflecting adversarial attacks with pixel deflection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8571–8580.

Aniket Pramanick and Dipankar Das. 2017. Ju cse nlp @ semeval 2017 task 7: Employing rules to detect and interpret english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 432–435.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. 2018. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *ICLR*.

Golnar Sheikhshabbafghi, Inanc Birol, and Anoop Sarkar. 2018. In-domain context-aware token embeddings improve biomedical named entity recognition. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*.

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

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Javier Tamames. 2005. Text detective: a rule-based system for gene annotation in biomedical texts. *BMC Bioinformatics*, 6(Suppl 1):S10.

The UniProt Consortium. 2017. Uniprot: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169.

Ö. Uzuner, B.R. South, S. Shen, and S.L. DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Ankit Vadehra. 2017. Uwav at semeval-2017 task 7: Automated feature-based system for locating puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 449–452.

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

Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 7: Locating the pun using syntactic characteristics and corpus-based metrics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 421–425.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *CoRR*, abs/1801.09851.

David Warde-Farley and Yoshua Bengio. 2017. Improving generative adversarial networks with denoising feature matching. In *ICLR*.

Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. 2013. Pubtator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research*, 41(W1):W518–W522.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.

Yanyan Zou and Wei Lu. 2019. Joint detection and location of english puns.