Robust Backed-off Estimation of Out-of-Vocabulary Embeddings

Nobukazu Fukuda
The University of Tokyo*
fukuda@tkl.iis.u-tokyo.ac.jp

Naoki Yoshinaga
Institute of Industrial Science,
the University of Tokyo
ynaga@iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science, the University of Tokyo
National Institute of Informatics
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

Out-of-vocabulary (OOV) words cause serious troubles in solving natural language tasks with a neural network. Existing approaches to this problem resort to using subwords, which are shorter and more ambiguous units than words, in order to represent OOV words with a bag of subwords. In this study, inspired by the processes for creating words from known words, we propose a robust method of estimating OOV word embeddings by referring to pre-trained word embeddings for known words with similar surfaces to target OOV words. We collect known words by segmenting OOV words and by approximate string matching, and we then aggregate their pre-trained embeddings. Experimental results show that the obtained OOV word embeddings improve not only word similarity tasks but also downstream tasks in Twitter and biomedical domains where OOV words often appear, even when the computed OOV embeddings are integrated into a BERT-based strong baseline.

1 Introduction

The dynamic nature of language and the limited size of training data requires neural network models to handle out-of-vocabulary (OOV) words that are absent from the training data. We thus use an UNK embedding shared among diverse OOV words or break those OOV words into semantically-ambiguous subwords (even characters), leading to poor task performance (Peng et al., 2019; Sato et al., 2020).

To solve this problem, several approaches (Pinter et al., 2017; Zhao et al., 2018; Sasaki et al., 2019) learn subword embeddings from pre-trained embeddings and then use these subword embeddings for computing OOV word embeddings (§ 2). However, the embeddings computed by these approaches are subject to the noisiness and ambiguity of intermediate subwords. For example, the **Glove.840B**[1] embedding of "higher" is closer to "high" compared with the embedding of "higher" reconstructed from its subwords using the method of BoS (Zhao et al., 2018), due to the ambiguous subword `er>` as shown in Table 1.

(sub)word	BoS					GloVe
	`highe`	`<high`	`high`	`er>`	higher	high
cosine	48.4	34.2	20.1	−7.8	36.5	69.8

Table 1: Cosine similarity between **Glove.840B** embedding of "higher" and related embeddings: subword and reconstructed embeddings of "higher" by BoS (Zhao et al., 2018) and **Glove.840B** embedding of "high."

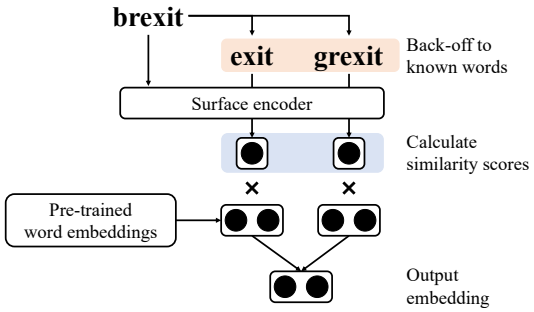

Figure 1: Backed-off estimation of OOV embedding.

Contextual word embeddings such as BERT (Devlin et al., 2019) can mitigate subword ambiguity by considering context. However, it has been reported that adversarial typos can degrade a BERT model that uses subword tokenization (Pruthi et al., 2019; Sun et al., 2020). Subword meanings change across domains, making domain adaptation difficult (Sato et al., 2020). These problems are more critical in the processing of noisy text (Wang et al., 2020; Niu et al., 2020).

To solve the above problems, we propose di-

*Currently, he works for NTT Laboratories.

[1] http://nlp.stanford.edu/data/glove.840B.300d.zip

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4827–4838
November 16 - 20, 2020.

rectly leveraging word-based pre-trained embeddings to compute OOV embeddings (Figure 1) (§ 3). Inspired by the two major processes for creating words, compounding and derivation, our method dynamically extracts words with pre-trained embeddings whose surfaces are similar to the target OOV word. Our method then aggregates pre-trained word embeddings on the basis of similarity score over the known words calculated by a character-level CNN encoder. We further integrate this method into a BERT-based fine-tuning model. In essence, the proposed method directly computes OOV embeddings from pre-trained word embeddings, whereas the existing methods compute indirectly via their subwords.

To investigate the performance of the proposed method against baseline approaches, we conduct both intrinsic and extrinsic evaluations of OOV word embeddings (§ 4). In the experiments for the intrinsic evaluation, we examine the performance of the proposed method in inducing embeddings for rare words by using the CARD benchmark (Pilehvar et al., 2018) and for misspelled words by using the TOEFL dataset (Flor et al., 2019). Then, in those for the extrinsic evaluation, we demonstrate the effectiveness of the calculated OOV word embeddings in two downstream tasks, named entity recognition (NER) and part-of-speech (POS) tagging for Twitter and biomedical domains, where OOV words frequently appear. We finally evaluate the BERT-based fine-tuning model with our method on these tasks and adversarial perturbations (Sun et al., 2020).

The contributions of this work are as follows.

- We propose a **robust backed-off approach for estimating OOV word embeddings**, inspired by two processes for creating words: compounding and derivation.
- We confirm by intrinsic and extrinsic evaluations that the **proposed method outperforms subword-based methods** in computing OOV word embeddings.
- We demonstrate that the **proposed extension to BERT boosts the performance of BERT except for one POS dataset** and robustness to adversarial perturbations on a sentiment dataset.

2 Related work

Existing approaches for leveraging surface information in computing OOV word embeddings basically learn the embeddings of characters or subwords to reconstruct pre-trained word embeddings from them and then use the obtained embeddings to compute embeddings for OOV words (Pinter et al., 2017; Zhao et al., 2018; Sasaki et al., 2019). Zhao et al. (2018) proposed Bag-of-Subwords (BoS) to reconstruct pre-trained word embeddings from bag-of-character n-grams in the same way as fastText (Bojanowski et al., 2017). Sasaki et al. (2019) extended BoS to reduce the number of embedding vectors and introduce a self-attention mechanism into the aggregation of subword embeddings. However, these methods compute embeddings via ambiguous character or subword embeddings. This will degrade the quality of embeddings for target OOV words as we will confirm later in § 4.

Other approaches utilize the embeddings of the known words around a target OOV word as its contextual information (Lazaridou et al., 2017; Khodak et al., 2018; Schick and Schütze, 2019; Hu et al., 2019). Schick and Schütze (2020) reported that they can improve BERT (Devlin et al., 2019) for understanding rare words. Notably, in these approaches for utilizing both surface and context information, the surface-based embeddings are the same as (Zhao et al., 2018). These approaches can have difficulties in representing misspelled words or spelling variations when a small number of contexts are available in a text corpus.

Several approaches utilize external data such as a knowledge base (Bahdanau et al., 2018; Yang et al., 2019; Yao et al., 2019). Existing approaches successfully impute OOV word embeddings by convolutional graph neural network (Yang et al., 2019) or by spectral embeddings derived from an affinity matrix of entities (Yao et al., 2019). These approaches can have difficulties in representing OOV words that do not exist in the external data and have little versatile applicability to misspelled words.

Recently, contextualized word embeddings such as BERT (Devlin et al., 2019) mitigate the problem of subword ambiguities by dynamically inferring meanings of OOV words from their contexts. However, several researchers reported that BERT remains brittle to misspellings (Pruthi et al., 2019; Sun et al., 2020), rare words (Schick and Schütze, 2020), and out-of-domain samples (Park et al., 2019). Pre-trained word embeddings are reported to be more effective for these cases and morphological tasks such as entity typing and NER (Zhu et al., 2019).

We therefore improve not only models based on pre-trained word embeddings but also the brittle subword-based BERT. Our approach will broaden the application range of neural-network models.

3 Robust backed-off estimation of OOV word embeddings

In this section, we describe our method of computing embeddings for target OOV words by using a weighted sum of pre-trained word embeddings. Specifically, we calculate the weights over known words from similarity scores derived on the basis of their surface information (Figure 1).

In what follows, we first describe how to retrieve known words that have surfaces similar to a target OOV word (§ 3.1), and we next describe how to aggregate their pre-trained embeddings on the basis of the similarity scores calculated through a neural network surface encoder (§ 3.2). We then discuss how to integrate our method into the BERT-based fine tuning model (Devlin et al., 2019) (§ 3.3).

3.1 Efficient back-off to known words

We first describe methods of efficiently extracting known words with a similar surface to a target OOV word: (i) segmentation of the target OOV word referring to known words and (ii) approximate string matching used for extracting known words with a similar surface from the OOV word. These components are inspired by the two major processes for creating words, namely, compounding and derivation, from existing words; we back-off unknown words to known words to rewind and replay the processes for creating words.

In this paper, we assume that word embeddings are already trained on a large corpus in an unsupervised method such as GloVe (Pennington et al., 2014). Backing off to these known words can alleviate the ambiguity of subwords because word-level pre-trained embeddings can be expected to be less polysemous than subword embeddings. Moreover, we do not update word-level pre-trained embeddings in training the reconstruction task described below. Then, we dynamically calculate the embeddings for OOV words in the same continuous space with known words.

Segmentation by known words Inspired by the compounding of words such as German nouns (*e.g.*, "Kinder|garten") and chemical compounds (*e.g.*, "dichloro|difluoro|methane"), the first approach extracts known words contained in the target OOV word. Using known words and characters as vocabulary, we first enumerate all possible segmentations of the OOV word as a lattice using dynamic programming and then choose the segmentation. We then extract n^{seg} known words in order of length from the segmentation with the smallest number of words/characters,[2] assuming longer words to be less ambiguous.

Approximate string matching Inspired by the derivation of words (*e.g.*, "ignore/ignorance"), the second approach extracts known words with similar surface features from the target OOV word by approximate string matching. As a similarity measure for the surface distance between the target OOV word and known words, we use string similarity coefficients that view a word as a bag of n-grams. The string similarity coefficient has been used for fast approximate string matching. We search for n^{approx} known words in order of string similarity to the OOV word. The approximate string matching is robust to subtle spelling variations and can extract a word with the correct spelling in most cases.

3.2 Aggregation of pre-trained word embeddings for known words

Next, we describe the calculation of OOV word embeddings from known words w_i^{seg} and w_i^{approx} that are extracted for the target OOV word q using the segmentation by known words and approximate string matching, respectively. Words extracted by the methods described above can contain undesired words whose meanings are far from the target OOV word. Thus, we calculate more accurate similarity scores for the extracted words through a neural network surface encoder.

By computing surface representations $\boldsymbol{v}_q$ and $\boldsymbol{v}_{w_i^k}$ of the target OOV word q and the extracted known words w_i^k ($k \in \{\text{seg}, \text{approx}\}$) through a character-level CNN (Zhang et al., 2015), we calculate the similarity score s_{q,w_i^k} between the OOV word q and known word w:

$$s_{q,w_i^k} = f(\boldsymbol{v}_q, \boldsymbol{v}_{w_i^k}; \boldsymbol{W}_k) \quad (1)$$

$$= \text{softmax}\left(\boldsymbol{v}_q^{\mathsf{T}} \cdot \boldsymbol{W}_k \cdot \boldsymbol{v}_{w_i^k}\right), \quad (2)$$

where $\boldsymbol{W}_k$ is a learnable parameter. We then aggregate the pre-trained word embeddings $\boldsymbol{e}_w$ on the basis of similarity scores $\boldsymbol{s}_{q,\boldsymbol{w}^k}$ to calculate the

[2] If there is more than one such segmentation, we extract unique known words from all segmentations in order of length.

OOV embedding $\hat{e}_q$ as follows:

$$\hat{\boldsymbol{e}}_q = \alpha_{\text{seg}} \boldsymbol{e}_q^{\text{seg}} + \alpha_{\text{approx}} \boldsymbol{e}_q^{\text{approx}}, \quad (3)$$

$$\text{where} \quad \boldsymbol{e}_q^k = \sum_i^{n^k} s_{q,w_i^k} \boldsymbol{e}_{w_i^k}, \quad (4)$$

$$\alpha_k = h(\boldsymbol{s}_{q,\boldsymbol{w}^k}; \boldsymbol{\theta}_k) \quad (5)$$

$$= \text{softmax}\left(\boldsymbol{\theta}_k \cdot \boldsymbol{s}_{q,\boldsymbol{w}^k}\right). \quad (6)$$

Here, $\boldsymbol{\theta}_k$ are learnable parameters, and $\boldsymbol{s}_{q,\boldsymbol{w}^k}$ represents the vector of similarities s_{q,w_i^k} between q and each known word w_i^k ($i = 1 \ldots n^{\text{seg}}$ or n^{approx}).

We consider utilizing cosine similarity as the reconstruction objective over an individual oracle embedding $\boldsymbol{e}_q$. Here, we used pre-trained embeddings, $\boldsymbol{e}_q$ (in experiments, **GloVe.840B**), as the oracle embeddings for each word:

$$\mathcal{L} = \text{cosine}(\boldsymbol{e}_q, \hat{\boldsymbol{e}}_q). \quad (7)$$

$\boldsymbol{W}_k$, $\boldsymbol{\theta}_k$, and the parameters of a character-CNN are updated in this manner. Although there are other ways of designing $\boldsymbol{W}_k$, such as using the same matrix for words of the same length, we leave this as future work.

3.3 Extension to BERT-based fine-tuning

We finally integrate the proposed method with contextualized word embeddings, BERT (Devlin et al., 2019). Although BERT works effectively in practice, the subword-based modeling is known to be brittle when the input has misspellings (§ 2). For example, typos in informative words can significantly change the set of subwords, which causes severe damage to subword-based modeling (*e.g.*, "robustness" → <robust|ness>, "robusntess" → <rob|us|ntes|s>). We thus utilize OOV embeddings computed by our method to enhance BERT.

We extend the pre-trained BERT to refer to oov embeddings in fine-tuning. We first tokenize each word into subwords with a BERT tokenizer (Wolf et al., 2019). For each subword, the embedding of the words containing the subwords is added to the BERT's pre-trained subword embedding as follows.

$$\boldsymbol{e} = (1-\alpha)\boldsymbol{e}^{\text{subword}} + \alpha \boldsymbol{W}_1 \cdot \boldsymbol{e}^{\text{word}} \quad (8)$$

$$\alpha = \text{sigmoid}(\boldsymbol{W}_2 \cdot \boldsymbol{e}^{\text{subword}}) \quad (9)$$

Here, $\boldsymbol{e}^{\text{subword}} \in \mathbb{R}^m$ is the BERT's subword embedding, and $\boldsymbol{e}^{\text{word}} \in \mathbb{R}^n$ is the pre-trained or OOV word embedding computed with our method. $\boldsymbol{W}_1 \in \mathbb{R}^{m \times n}$ and $\boldsymbol{W}_2 \in \mathbb{R}^n$ are learnable parameters in the fine-tuning. For example, when "robusntess" is tokenized into <rob|us|ntes|s>, <rob's embedding [$\boldsymbol{e}$ in (8)] is calculated from BERT's subword embedding of <rob ($\boldsymbol{e}^{\text{subword}}$) and our embedding of "robusntess" ($\boldsymbol{e}^{\text{word}}$).

Finally, we input the subword embeddings computed in this way to the BERT model and calculate the output label. In the fine-tuning process, we update the parameters of the BERT model including its embedding layers while fixing the word embeddings computed with the proposed method. This method is applicable to any neural network other than BERT-based fine tuning model.

4 Experiments

We evaluated the performance of OOV word embeddings calculated by the proposed method. We performed intrinsic evaluations on benchmark datasets (§ 4.1) and extrinsic evaluations on two downstream tasks: named entity recognition (NER) and part-of-speech (POS) tagging (§ 4.2). We then conducted experiments on the extension to a BERT-based fine-tuning model (§ 4.3 and § 4.4).

We used **Glove.840B** embeddings[1] (2.2M vocabulary size) as the pre-trained embeddings (known words) following Sasaki et al. (2019). In all the experiments, we used PyTorch (v1.0.1)[3] as the core architecture and regarded words that were absent from **GloVe.840B** as OOV words. For the extrinsic evaluations (§ 4.2, § 4.3 and § 4.4), the reported numbers are the medians of five trials.

4.1 Intrinsic evaluations: CARD and TOEFL

We evaluated the performance of OOV embeddings through similarity estimations of rare words or misspelled words. For the baseline methods, we also evaluated the **BoS** (Zhao et al., 2018) and **KVQ-FH** ($F = 1\text{M}, H = 0.5\text{M}$) (Sasaki et al., 2019) referred to in § 2. Although these studies focus on replacing infrequent words as well as OOV words, we here focus on replacing only OOV words. In addition to these subword-based baselines, we used a simple baseline (**Simple back-off**) that backs off an OOV word to the most orthographically-similar known word. This is a special case of the proposed method with $n^{\text{seg}} = 0, n^{\text{approx}} = 1$.

Datasets and experimental settings

The CARD-660 (Pilehvar et al., 2018) (hereafter, CARD) is a rare word benchmark that consists of pairs of words annotated with their similarity score.

[3] https://pytorch.org/

CARD contains 660 word pairs of 1306 words with duplicates collected from various domains, such as computer science, social media, and medicine. Among these word pairs, 289 pairs contain OOV words that are absent from **GloVe.840B**. We calculated the cosine similarities between the embeddings for the two words in a pair and evaluated the Spearman's correlation coefficient ρ between the cosine similarity and the annotated similarity. We also calculated the correlation for pairs containing OOV words.

The TOEFL-Spell dataset (Flor et al., 2019) (hereafter, TOEFL) contains examples of misspellings appearing in a corpus of essays written by English language learners. We extracted 1514 pairs of a correct, known word and a misspelled, OOV word for evaluation. We computed the embedding for the misspelled word and evaluated the cosine similarity to the gold embedding of the correct word. We could evaluate the robustness against practical misspelled words using TOEFL rather than synthetic adversarial perturbations.

To train the subword-based baselines and the proposed method, we randomly sampled 10^5 words with frequency f[4] ($10^3 < f < 10^5$) from the **Glove.840B** embeddings. We then sampled the 10^3 embeddings from the 10^5 embeddings as the development data and the remaining embeddings to optimize the parameters of the proposed method. We adopted Adam (Kingma and Ba, 2015) with a learning rate of 10^{-3} as the optimizer. We set the gradient clipping as 1, the dropout rate as 0.3, the number of epochs as 50, and the batch size as 1000, and we chose the model at the epoch that achieved the maximum total cosine similarities between the **GloVe.840B** embeddings and the induced embeddings for the target words in the development data.

To find orthographically similar words with the proposed method and **Simple back-off**, we ran **Simple back-off** with various similarity measures[5] (Dice, Cosine, Jaccard, and Overlap) and n-grams ($1 \leq n \leq 7$), and we obtained the Jaccard measure based on 3-grams as the best-performing configuration for the development set. In computing the surface representations $\boldsymbol{v}_q$ and $\boldsymbol{v}_{w_i^k}$ in Eq. 2 through a character-CNN with the proposed method, we set the dimension to 100, and the convolutions had window sizes of 1, 3, 5 and 7 characters. We then searched for the best-performing hyperparameters n^{seg} and n^{approx} ($1 \leq n^* \leq 10$) of the proposed method for the development set and obtained $n^{\text{seg}} = 7$, $n^{\text{approx}} = 10$.

[4]We used the word frequencies at https://github.com/losyer/compact_reconstruction.

[5]https://github.com/chokkan/simstring

	CARD (ρ)		TOEFL (cos)
	ALL	OOV	OOV
GloVe (Pennington et al., 2014)	27.3	-	-
BoS (Zhao et al., 2018)	40.7	17.2	33.4
KVQ-FH (Sasaki et al., 2019)	44.0	25.0	28.8
Simple back-off	45.8	32.5	**44.5**
This work	**47.6**	**35.3**	37.5

Table 2: Results of intrinsic evaluation of OOV embeddings. **CARD** was evaluated with Spearman's correlation coefficient ρ and **TOEFL** with cosine similarity.

Results

Table 2 shows the results of the intrinsic evaluations. ALL indicates the performance on all word pairs, while OOV indicates the performance only on pairs that contained an OOV word. We regarded the cosine similarity of a word pair as zero when a method could not compute embeddings for OOV words in pairs, following Yang et al. (2019). The correlation coefficients for the known word pairs were 55.5 for all methods for the CARD dataset. We observed that the proposed method outperformed all baselines except for **Simple back-off** on the TOEFL dataset. We considered the **Simple back-off** baseline to be tailored for misspelings. Notably, compared with the subword-based methods, the proposed method was robust against the misspelled words for the TOEFL dataset, which demonstrates the risk of relying on subword embeddings.

We conducted a qualitative analysis of our two modules: segmentation by known words (hereafter SEG) and approximate string matching (hereafter APPROX) described in § 3.2. SEG successfully handled compound words such as "horse|cloth" and "boat|master," while APPROX successfully handled "aeolipile." The meanings of these words can be inferred from their surfaces; we believe that our method could successfully compute their embeddings by relating them to the known words. Both methods failed to handle "covfefe" (a misspelling of "coverage") and proper nouns such as "Kobani" (a place) and "AccuRay" (a company). From these observations, the proposed method is considered to be less effective with these words as well as acronyms whose meanings are difficult to predict from their surfaces.

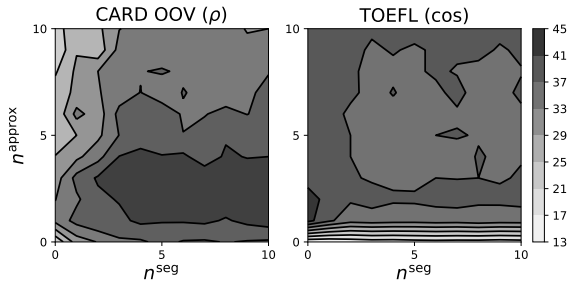

Figure 2: Results of intrinsic evaluations on pairs with OOV words when varying n^{seg} and n^{approx}.

Sensitivity to hyper-parameters

To investigate the impact of using the two modules, segmentation by known words and approximate string matching (§ 3.2), we evaluated the performance of the intrinsic evaluations with various n^{seg} and n^{approx}. Figure 2 depicts the performance of the proposed method with different $n^{\text{seg}}, n^{\text{approx}}$ ($0 \leq n^{*} \leq 10$). The larger the n^{seg} and n^{approx} used, the more shorter and less superficially-similar known words were taken into account. This result shows that segmentation by known words tended to benefit compound words in CARD, and approximate string matching tended to benefit misspellings in TOEFL. This is reasonable since we can guess the meaning of a compound word in a constructive way and the meaning of a misspelled word by considering words with close spellings.

4.2 Extrinsic evaluations: NER and POS tagging

We then conducted extrinsic evaluations on NER and POS tagging using the proposed and baseline methods to compute OOV embeddings. In addition to the baselines in the intrinsic evaluation, we used two baselines for comparison: a single unknown embedding (**Single-UNK**) and a context-based model (**HiCE**) (Hu et al., 2019). **Single-UNK** trains a single embedding for OOV words in training on downstream datasets. **HiCE** uses only context information to encode target OOV words. We also evaluated combinations of individual surface-based methods with **HiCE**. We integrated of a surface embedding, $\boldsymbol{e}^{\text{surface}}$, and a context embedding, $\boldsymbol{e}^{\text{context}}$, following Schick and Schütze (2019):

$$\boldsymbol{e} = (1-\alpha)\cdot \boldsymbol{e}^{\text{surface}} + \alpha \cdot \boldsymbol{e}^{\text{context}} \quad (10)$$

$$\alpha = \text{sigmoid}(\boldsymbol{W}[\boldsymbol{e}^{\text{surface}}; \boldsymbol{e}^{\text{context}}]). \quad (11)$$

We simultaneously trained $\boldsymbol{W}$ as well as $\boldsymbol{e}^{\text{surface}}$ and $\boldsymbol{e}^{\text{context}}$ with the pre-trained embeddings.

Datasets	#sents.	OOV% token	OOV% type
Twitter NER			
RARE-NER (Derczynski et al., 2017)	5690	7%	27%
MULTI-NER (Zhang et al., 2018)	8257	16%	49%
Biomedical NER			
BC2GM (Smith et al., 2008)	20,131	2%	22%
BC4CHEMD (Krallinger et al., 2015)	87,685	2%	29%
BC5CDR (Wei et al., 2016)	13,938	1%	8%
NCBI-DISEASE (Dogan et al., 2014)	7287	2%	13%
Twitter POS			
ARK (Gimpel et al., 2011)	1827	11%	29%
T-POS (Ritter et al., 2011)	787	7%	20%
DCU (Foster et al., 2011)	519	4%	10%

Table 3: Datasets used in extrinsic evaluations.

Datasets and experimental settings

We evaluated whether the computed embeddings captured semantic and morphosyntactic information through NER and POS tagging on domains where many OOV words appear. Table 3 shows a summary of the datasets used in the extrinsic evaluations. Here, OOV% represents the OOV word rate in each dataset. For each dataset, we used the standard split for training, development, and test sets. We used the training data of T-POS for the DCU training following Derczynski et al. (2013).

We adopted the Bi-LSTM-CRF (Lample et al., 2016) and Bi-LSTM tagger (Pinter et al., 2017) for NER and POS tagging and measured the performance in terms of the classification accuracy and entity-level F_1 score, respectively. We used two bidirectional LSTM layers of hidden size 200 for the two taggers. In the training of both taggers, we adopted Adam with a learning rate of 10^{-3} as the optimizer. We set the gradient clipping as 1, the dropout rate as 0.5, and the number of epochs as 50, and the batch size was 500 for the biomedical NER datasets and 50 for the Twitter POS and NER datasets, and we then chose the model at the epoch that achieved the best performance on the development data.

When training the taggers, we fixed their embedding layers to the pre-trained embeddings or OOV embeddings computed by each method except for the shared OOV embedding in **Single-UNK**. Since **HiCE** uses an external corpus [here, Wikitext-103 (Merity et al., 2017)] as contexts for training, we trained all the methods to compute OOV embeddings using a part of the pre-trained embeddings used for training in the intrinsic evaluations whose contexts are available in Wikitext-103. In training

	RARE-NER		MULTI-NER		BC2GM		BC4CHEMD		BC5CDR		NCBI-DISEASE	
	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK	37.4	4.0	69.0	29.6	77.5	76.1	85.0	71.1	84.8	64.0	**82.7**	67.3
BoS	38.6*	6.9*	68.6	28.3	77.4	76.4	84.8	70.4	84.7	59.2	81.8	66.1
KVQ-FH	38.4	6.5*	67.9	27.3	77.4	76.6	84.9	71.0	84.6	59.6	82.1	67.3
Simple back-off	38.5*	**19.1***	67.5	**40.3***	78.1*	78.1*	86.0*	76.4*	85.0	66.2*	**82.7**	76.2*
This work	**39.1**	12.1*	**69.5**	37.9*	**78.7***	**79.5***	**86.6***	**79.2***	**85.2***	**72.7***	82.5	**77.2***
HiCE	36.4	3.4	67.6	26.8	77.0	76.0	84.7	70.8	84.3	62.9	82.8	72.1
+BoS	38.1*	4.7*	67.9*	27.5	77.5*	76.3	84.7	70.6	84.8*	60.2	82.5	67.3
+KVQ-FH	36.8	4.8	67.9*	27.3	77.6*	76.2	84.7	71.5	84.9*	62.2	82.2	66.1
+Simple back-off	36.7	**17.6***	68.2*	**39.5***	**78.1***	78.3*	86.2*	77.0*	84.9*	66.0*	83.0	71.0
+This work	**38.4***	12.4*	**69.2***	37.9*	**78.1***	**79.4***	**86.6***	**79.1***	**85.3***	**74.1***	**83.4**	**77.6***

Table 4: Results of extrinsic evaluations of OOV embeddings for Twitter and biomedical NER; numbers indicate best F_1 among all methods with **each** and all resource settings, respectively. * indicates statistically significant improvements ($p < 0.05$) over **Single-UNK** and **HiCE** by paired permutation test.[6]

	ARK		T-POS		DCU	
	ALL	OOV	ALL	OOV	ALL	OOV
Single-UNK	82.6	53.9	81.0	56.7	82.1	62.1
BoS	82.7	53.1	80.3	53.9	81.9	62.9
KVQ-FH	82.5	53.9	80.5	54.5	82.0	62.9
Simple back-off	84.5*	71.8*	**81.5***	68.0*	82.4	71.8*
This work	**85.3***	**74.3***	**81.5**	**69.1***	**82.9***	**74.2***
HiCE	81.2	54.1	80.5	57.3	81.3	61.3
+BoS	82.8*	53.8	80.2	53.4	81.9*	59.7
+KVQ-FH	82.8*	54.0	80.3	53.9	81.9*	58.9
+Simple back-off	84.5*	71.7*	**81.7***	68.5*	82.1*	70.2*
+This work	**85.1***	**74.7***	81.5*	**69.1***	**83.1***	**72.6***

Table 5: Results of extrinsic evaluations of OOV embeddings on Twitter POS tagging; numbers indicate best accuracy among all methods with **each** and all resource settings, respectively. * indicates statistically significant improvements ($p < 0.05$) over **Single-UNK** and **HiCE** by Mann-Whitney U test.

(BC2GM) inhibitor of influenzavirus **neuraminidases**$_{\text{I-GENE}}$.		
BoS	peptidases peroxidases proteinases esterases oxidases	O
This work	neuraminidase Neuraminidase hemagglutinin sialidase haemagglutinin	**I-GENE**
(BC5CDR) during **amphotercin**$_{\text{B-Chemical}}$ B administration .		
BoS	amphora rhamphotheca gargantua verticillated canorous	O
This work	Amphotericin Amphoteric amphotericin AmB amphoteric	**B-Chemical**
(BC4CHEMD) A new **chromone**$_{\text{B-Chemical}}$ from the leaves of		
BoS	kairomones pheromone pheromones Pomone neuropeptide	**B-Chemical**
This work	chromos chromosones chromo givesome chromosone	O

Table 6: Example outputs for biomedical NER and nearest-neighbor known words for computed OOV embeddings of target OOV words.

HiCE, we set the hidden size as 300, intermediate hidden size as 600, number of self-attention layers as 2, number of self-attention heads as 10, dropout rate of the context encoder and multi-context aggregator as 0.3, the maximum number of contexts as 10, and the context window size as 10. With the trained **HiCE**, we computed the embeddings for OOV words from their contexts in the training/test data of the target task and Wikitext-103.

Results

Tables 4 and 5 list the results for the two tasks. Here, ALL shows the overall performance, and OOV indicates the performance only on words or entities that are absent from the training data[7] and have no GloVe embeddings. The proposed method outperformed the baselines for OOV words, except for RARE-NER and MULTI-NER. This result confirms the effectiveness of the proposed method in computing OOV embeddings.

The proposed method exhibited additive improvements over **HiCE**. The tendency for the surface-based methods to outperform the purely context-based baseline **HiCE** suggests that the LSTM models for the downstream tasks already captured the contextual information.

Examples Table 6 shows examples of the system outputs. The proposed method successfully predicted word embedding for two OOV words: "neuraminidases" (a plural form of "neuraminidase")

[6] http://cr.fvcrc.i.nagoya-u.ac.jp/~sasano/test/permutation.html

[7] This is because the models will be able to handle words in the training data regardless of the quality of their embeddings.

	RARE-NER		MULTI-NER		BC2GM		BC4CHEMD		BC5CDR		NCBI-DISEASE	
	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
BERT	43.7	26.0	72.5	57.6	79.9	79.8	85.6	**79.3**	84.7	71.7	84.5	**85.7**
+BoS	44.4	**29.7***	73.0*	57.2	80.2	79.1	**87.0***	78.6	85.2*	71.8	85.9*	84.9
+KVQ-FH	43.3	25.6	72.1	56.7	80.5*	80.0	86.8*	78.3	85.4*	69.3	**86.1***	82.9
+Simple back-off	42.8	28.3	73.1*	**58.1**	79.8	79.4	86.4*	78.3	85.4*	**73.1**	85.6*	81.8
+This work	**44.6**	23.2	**73.5***	57.7	**80.9***	**81.2***	86.8*	79.1	**85.7***	71.4	85.7*	82.0

Table 7: Results of BERT fine-tuning with and without proposed extension on Twitter and biomedical NER; * indicates statistically significant improvements ($p < 0.05$) over BERT by paired permutation test.

	ARK		T-POS		DCU	
	ALL	OOV	ALL	OOV	ALL	OOV
BERT	92.1	88.3	90.3	87.7	90.5	85.4
+BoS	92.2	87.3	90.4	**88.4**	**90.7**	**87.8**
+KVQ-FH	**92.4**	88.4	**90.8***	**88.4**	90.6	87.0
+Simple back-off	**92.4**	88.4	90.5	**88.4**	90.2	84.5
+This work	**92.4**	**88.7**	**90.8***	**88.4**	90.4	85.4

Table 8: Results of fine-tuned BERT model with and without proposed extension for Twitter POS tagging; * indicates statistically significant improvements ($p < 0.05$) over BERT by Mann-Whitney U test.

(RARE-NER) is that Mauro **renallo**$_{\text{I-person}}$?		
BERT	<renal|lo>	O
+This work	Ranallo Mathhew Prazak Bednarski Lesie	**I-person**

(BC5CDR) - bound dimethylarsenic (**DMAs**$_{\text{B-Chemical}}$) ,		
BERT	<dime|thy|lars|eni|c>	O
+This work	dimethylamine dimethylated dimethyl morpholine	**B-Chemical**

(BC5CDR) A phase I clinical study of the **antipurine**$_{\text{B-Chemical}}$		
BERT	<anti|puri|ne>	**B-Chemical**
+This work	anti-mine antisubmarine antimycotic glutethimide impetiginous	O

Table 9: Example outputs for Twitter/biomedical NER and BERT tokenization and nearest-neighbor known words for computed embeddings of OOV words.

and "amphotercin" (a misspelling of "amphotericin"), while **BoS** was strongly influenced by the subwords contained in the OOV words. As shown in the last example, however, the proposed method wrongly predicted "chromone" (a chemical compound) as "chromosome" away from the correct domain. This example suggests the limitations of using a surface-based approach that does not utilize context information.

4.3 Extrinsic evaluation: extension to BERT

To investigate the effectiveness of integrating our OOV embeddings into the BERT, we conducted experiments on the downstream tasks. We used BERT with a token classification head on top (Wolf et al., 2019) for POS tagging and that with a CRF layer for NER. In the integration of surface-based embeddings into BERT (§ 3.3), we fixed the surface-based embeddings and fine-tuned the WordPiece subword embeddings and model parameters of the BERT. We adopted Adam with a learning rate of 5×10^{-5} as the optimizer, and we set the number of epochs as 20, the batch size as 16, and other hyper-parameters the same as Wolf et al. (2019). To ensure that the number of tokens did not exceed the maximum length limit of BERT,[8] we only used sentences with 100 words or less for all datasets in the fine-tuning of the BERT model.

Tables 7 and 8 show a comparison of the BERT models with and without the proposed extension in POS tagging and NER. Our extension to BERT improved the overall performance on all datasets except for DCU; all the improvements were significant except for RARE-NER and ARK. Although our extension was sometimes harmful in recognizing entities that have OOV words (RARE-NER, BC4CHEMD, BC5CDR, and NCBI-DISEASE), it still improved the overall performance. We consider this to be because our extension helps BERT utilize contexts with OOV words to classify known words.

Examples Table 9 shows examples of the system outputs. The proposed method successfully predicted word embeddings for two OOV words: "renallo" (a person) and "dimethylarsenic" (a chemical compound), while the BERT tokenizer tokenized these words into short pieces, which might have lead to the incorrect labels. As shown in the last example of "antipurine," however, the proposed

[8] Due to this limitation, several test examples are removed from the original test data of NER except for MULTI-NER. Thus, it is difficult to compare numbers in Table 4 and Table 7.

method was influenced by words with common prefixes `<anti` and predicted a wrong embedding away from the context.

4.4 Extrinsic evaluation: adversarial typos

Finally, we evaluated the robustness to adversarial attacks on the extension to BERT-based fine-tuning. We considered adversarial typos on a keyboard (Pruthi et al., 2019; Sun et al., 2020) to consider natural perturbations in the real world.

We describe the generation of adversarial perturbation via keyboard typos. First, we selected words to be edited according to the max-grad policy (Sun et al., 2020). We computed the gradients of a pre-trained BERT classifier at a time and selected words in order of larger gradients. Second, we explored four types of subtle character-level edits for each word: (i) swapping two adjacent characters in a word, (ii) removing a character in a word, (iii) replacing a character with an adjacent character on the keyboard, and (iv) inserting an adjacent character on the keyboard before a character in a word. We did not edit stop words or words with less than three characters, following Pruthi et al. (2019). To reduce the computational cost, we limited the number of candidates of typos to N typos randomly. In this paper, we set the hyperparameters as $K \in \{0, 1, 3, 5\}, N = 10$.

We used movie reviews from the Stanford Sentiment Treebank (SST) (Socher et al., 2013), which consists of 8544 movie reviews. With only positive and negative reviews, we trained a BERT with a sequence classification head on top (Wolf et al., 2019) to generate the adversarial examples described above. We evaluated the accuracy for the BERT-based fine-tuning model and a word-based LSTM model with and without embeddings computed with the proposed method. For comparison, we also evaluated a variant of our extension to BERT that assigns a zero vector instead of embeddings computed by the proposed method to OOV words (**+GloVe**). As with the LSTM tagger in § 4.2, we used two bidirectional LSTM layers of hidden size 200. In the training of both models, we adopted Adam with a learning rate of 5×10^{-5} for the BERT model and of 10^{-3} for the LSTM model, set the gradient clipping as 1, the dropout rate as 0.5, the number of epochs as 10, and the batch size as 16.

Table 10 shows the results for the sentiment classification task. K in the table indicates the number of words edited in a text, and `#tokens per words` indicates the average number of subwords in words when the words were tokenized with the BERT tokenizer. Although this result shows that the BERT outperformed the other models without any perturbations ($K = 0$), its performance degraded as K and `#tokens` increased, which is consistent with (Sun et al., 2020). Moreover, the proposed method could mitigate this performance degradation of both BERT and LSTM models.

K	0	1	3	5
`#tokens per word`	1.20	1.26	1.34	1.40
BERT	**90.6**	78.8	57.0	39.8
+GloVe	89.2	80.3	64.4*	51.8*
+This work	89.2	79.9	66.8	53.8*
LSTM	77.5	73.8	67.0	62.7
+GloVe	86.1*	80.8*	73.4*	66.7
+This work	85.8*	**81.5***	**75.4***	**70.7***

Table 10: Results of BERT model with and without embeddings computed with the proposed method on SST with adversarial perturbations. `#tokens per word` indicates average number of tokenized subwords in word. * indicates statistically significant improvements ($p < 0.05$) over BERT and LSTM by Mann-Whitney U test.

5 Conclusion

In this paper, inspired by two major processes for creating words, we proposed a method for computing OOV word embeddings by learning the similarities between a target OOV word and known words and integrated the method into BERT. We conducted intrinsic and extrinsic evaluations, and we confirmed that the proposed method more successfully mimics the pre-trained word embeddings for OOV words than existing subword-based methods that suffer from the noisiness and ambiguity of intermediate subwords. The proposed method boosted the performance of BERT and equipped BERT with robustness to adversarial typos.

We will release all code to promote the reproducibility of our results.[9] In future work, we will investigate better integrating our method into BERT.

Acknowledgments

The research was supported by NII CRIS collaborative research program operated by NII CRIS and LINE Corporation.

[9] http://www.tkl.iis.u-tokyo.ac.jp/~fukuda/emnlp2020/

References

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2018. Learning to compute word embeddings on the fly. *CoRR*, abs/1706.00286.

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

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 198–206, Hissar, Bulgaria.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Michael Flor, Michael Fried, and Alla Rozovskaya. 2019. A benchmark corpus of English misspellings and a minimally-supervised model for spelling correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 76–86, Florence, Italy.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: POS tagging and parsing the twitterverse. In *Proceedings of the fifth AAAI Conference on Analyzing Microtext*, pages 20–25.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47, Portland, Oregon, USA.

Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-shot representation learning for out-of-vocabulary words. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4102–4112, Florence, Italy.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Melbourne, Australia.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzábal, and Alfonso Valencia. 2015. CHEMDNER: The drugs and chemical names extraction challenge. In *Journal of Cheminformatics*, volume 7 (Suppl 1).

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive science*, 41 (Suppl 4):677–705.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *The fifth International Conference on Learning Representations*.

Xing Niu, Prashant Mathur, Georgiana Dinu, and Yaser Al-Onaizan. 2020. Evaluating robustness to input perturbations for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8538–8544, Online.

Cheoneum Park, Juae Kim, Hyeon-gu Lee, Reinald Kim Amplayo, Harksoo Kim, Jungyun Seo, and Changki Lee. 2019. ThisIsCompetition at SemEval-2019 task 9: BERT is unstable for out-of-domain samples. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1254–1261, Minneapolis, Minnesota, USA.

Minlong Peng, Qi Zhang, Xiaoyu Xing, Tao Gui, Jinlan Fu, and Xuanjing Huang. 2019. Learning task-specific representation for novel words in sequence labeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5146–5152.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. 2018. Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1391–1401, Brussels, Belgium.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK.

Shota Sasaki, Jun Suzuki, and Kentaro Inui. 2019. Subword-based Compact Reconstruction of Word Embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3498–3508, Minneapolis, Minnesota.

Shoetsu Sato, Jin Sakuma, Naoki Yoshinaga, Masashi Toyoda, and Masaru Kitsuregawa. 2020. Vocabulary adaptation for domain adaptation in neural machine translation. In *Findings of EMNLP 2020*.

Timo Schick and Hinrich Schütze. 2019. Attentive mimicking: Better word embeddings by attending to informative contexts. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 489–494, Minneapolis, Minnesota.

Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 8766–8774.

Larry Smith, Lorraine K. Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M. Friedrich, Kuzman Ganchev, Manabu Torii, Hongfang Liu, Barry Haddow, Craig A. Struble, Richard J. Povinelli, Andreas Vlachos, William A. Baumgartner, Lawrence E. Hunter, Bob Carpenter, Richard Tzong-Han Tsai, Hong-Jie Dai, Feng Liu, Yifei Chen, Chengjie Sun, Sophia Katrenko, Pieter Adriaans, Christian Blaschke, Rafael Torres, Mariana Neves, Preslav Nakov, Anna Divoli, Manuel J. Maña-López, Jacinto Mata, and W. John Wilbur. 2008. Overview of biocreative ii gene mention recognition. *Genome Biology*, 9 (Suppl 2).

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, Seattle, Washington, USA.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jiugang Li, Philip S. Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *ArXiv*, abs/2003.04985.

Zhiwei Wang, Hui Liu, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zhiwei Liu. 2020. Learning multi-level dependencies for robust word recognition. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 9250–9257.

Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Jiao Li, Thomas C. Wiegers, and Zhiyong Lu. 2016. Assessing the state of the art in biomedical relation extraction: overview of the biocreative v chemical-disease relation (cdr) task. *Database : the journal of biological databases and curation*, 2016.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Ziyi Yang, Chenguang Zhu, Vin Sachidananda, and Eric Darve. 2019. Embedding imputation with grounded language information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3356–3361, Florence, Italy.

Shibo Yao, Dantong Yu, and Keli Xiao. 2019. Enhancing domain word embedding via latent semantic imputation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 557–565.

Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. 2018. Adaptive co-attention network for named entity recognition in tweets. In *Proceedings of the thirty-Second AAAI Conference on Artificial Intelligence*, pages 5674–5681.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing word embeddings using bag of subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 601–606, Brussels, Belgium.

Yi Zhu, Benjamin Heinzerling, Ivan Vulić, Michael Strube, Roi Reichart, and Anna Korhonen. 2019. On the importance of subword information for morphological tasks in truly low-resource languages. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 216–226, Hong Kong, China.