→ ~~method~~ embedding [illegible] OOV.

Imputing Out-of-Vocabulary Embeddings with LOVE Makes Language Models Robust with Little Cost

Lihu Chen[1], **Gaël Varoquaux**[2], **Fabian M. Suchanek**[1]
[1] LTCI & Télécom Paris & Institut Polytechnique de Paris, France
[2] Soda, Inria Saclay & CEA & Université Paris-Saclay, France
`{lihu.chen, fabian.suchanek}@telecom-paris.fr`
`{gael.varoquaux}@inria.fr`

Abstract

State-of-the-art NLP systems represent inputs with word embeddings, but these are brittle when faced with Out-of-Vocabulary (OOV) words. To address this issue, we follow the principle of mimick-like models to generate vectors for unseen words, by learning the behavior of pre-trained embeddings using only the surface form of words. We present a simple contrastive learning framework, LOVE, which extends the word representation of an existing pre-trained language model (such as BERT), and makes it robust to OOV with few additional parameters. Extensive evaluations demonstrate that our lightweight model achieves similar or even better performances than prior competitors, both on original datasets and on corrupted variants. Moreover, it can be used in a plug-and-play fashion with FastText and BERT, where it significantly improves their robustness.

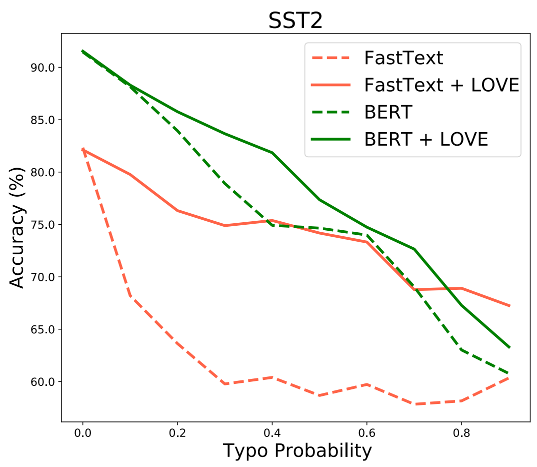


Figure 1: Performances of existing word embeddings as we gradually add typos to the datasets. Using our model, LOVE, to produce vectors for OOV words makes the models more robust.

1 Introduction

Word embeddings represent words as vectors (Mikolov et al., 2013a,b; Pennington et al., 2014). They have been instrumental in neural network approaches that brought impressive performance gains to many natural language processing (NLP) tasks. These approaches use a fixed-size vocabulary. Thus they can deal only with words that have been seen during training. While this works well on many benchmark datasets, real-word corpora are typically much noisier and contain Out-of-Vocabulary (OOV) words, i.e., rare words, domain-specific words, slang words, and words with typos, which have not been seen during training. Model performance deteriorates a lot with unseen words, and minor character perturbations can flip the prediction of a model (Liang et al., 2018; Belinkov and Bisk, 2018; Sun et al., 2020; Jin et al., 2020). Simple experiments (Figure 1) show that the addition of typos to datasets degrades the performance for text classification models considerably.

To alleviate this problem, pioneering work pre-trained word embeddings with morphological features (sub-word tokens) on large-scale datasets (Wieting et al., 2016; Bojanowski et al., 2017; Heinzerling and Strube, 2017; Zhang et al., 2019). One of the most prominent works in this direction is FastText (Bojanowski et al., 2017), which incorporates character n-grams into the skip-gram model. With FastText, vectors of unseen words can be imputed by summing up the n-gram vectors. However, these subword-level models come with great costs: the requirements of pre-training from scratch and high memory footprint. Hence, several simpler approaches have been developed, e.g., MIMICK (Pinter et al., 2017), BoS (Zhao et al., 2018) and KVQ-FH (Sasaki et al., 2019). These use only the surface form of words to generate vectors for unseen words, through learning from pre-trained embeddings.

Although MIMICK-like models can efficiently reduce the parameters of pre-trained representa-

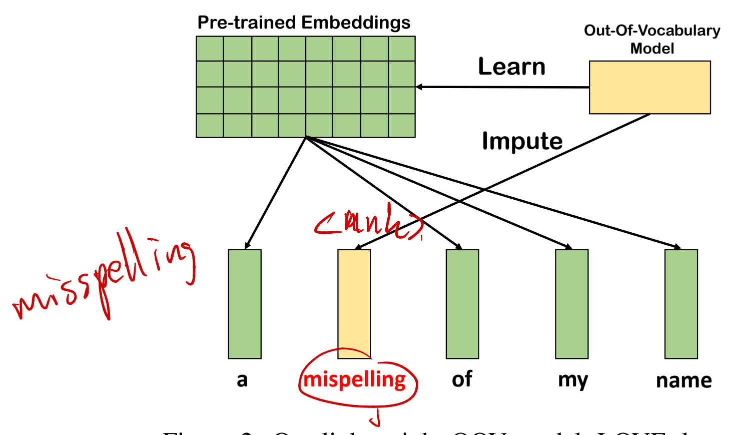


Figure 2: Our lightweight OOV model, LOVE, learns the behavior of pre-trained embeddings (e.g., FastText and BERT), and is then able to impute vectors for unseen words. LOVE can enhance the robustness of existing word representations in a plug-and-play fashion.

tions and alleviate the OOV problem, two main challenges remain. First, the models remain bound in the trade-off between complexity and performance: The original MIMICK is lightweight but does not produce high-quality word vectors consistently. BoS and KVQ-FH obtain better word representations but need more parameters. Second, these models cannot be used with existing pre-trained language models such as BERT. It is these models, however, to which we owe so much progress in the domain (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019; Liu et al., 2020). To date, these high-performant models are still fragile when dealing with rare words (Schick and Schütze, 2020), misspellings (Sun et al., 2020) and domain-specific words (El Boukkouri et al., 2020).

We address these two challenges head-on: we design a new contrastive learning framework to learn the behavior of pre-trained embeddings, dubbed LOVE, **L**earning **O**ut-of-**V**ocabulary **E**mbeddings. Our model builds upon a memory-saving mixed input of character and subwords instead of n-gram characters. It encodes this input by a lightweight Positional Attention Module. During training, LOVE uses novel types of data augmentation and hard negative generation. The model is then able to produce high-quality word representations that are robust to character perturbations, while consuming only a fraction of the cost of existing models. For instance, LOVE with 6.5M parameters can obtain similar representations as the original FastText model with more than 900M parameters. What is more, our model can be used in a plug-and-play fashion to robustify existing language models. We find that using LOVE to produce vectors for unseen words improves the performance of FastText and BERT by around 1.4-6.8 percentage points on noisy text – without hampering their original capabilities (As shown in Figure 2).

In the following, Section 2 discusses related work, Section 3 introduces preliminaries, Section 4 presents our approach, Section 5 shows our experiments, and Section 6 concludes. The appendix contains additional experiments and analyses. Our code and data is available at `https://github.com/tigerchen52/LOVE`

2 Related Work

missspell
misspell

2.1 Character-level Embeddings

To address OOV problems, some approaches inject character-level features into word embeddings during the pre-training (Wieting et al., 2016; Cao and Rei, 2016; Bojanowski et al., 2017; Heinzerling and Strube, 2017; Kim et al., 2018; Li et al., 2018; Üstün et al., 2018; Piktus et al., 2019; Zhu et al., 2019; Zhang et al., 2019; Hu et al., 2019). One drawback of these methods is that they need to pre-train on a large-scale corpus from scratch. Therefore, simpler models have been developed, which directly mimic the well-trained word embeddings to impute vectors for OOV words. Some of these methods use only the surface form of words to generate embeddings for unseen words (Pinter et al., 2017; Zhao et al., 2018; Sasaki et al., 2019; Fukuda et al., 2020; Jinman et al., 2020), while others use both surface and contextual information to create OOV vectors (Schick and Schütze, 2019a,b). In both cases, the models need an excessive number of parameters. FastText, e.g., uses ~2 million n-gram characters to impute vectors for unseen words.

2.2 Pre-trained Language Models

Currently, the state-of-the-art word representations are pre-trained language models, such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019) and XLnet (Yang et al., 2019), which adopt subwords to avoid OOV problems. However, BERT is brittle when faced with rare words (Schick and Schütze, 2020) and misspellings (Sun et al., 2020). To make BERT more robust, CharacterBERT (El Boukkouri et al., 2020) and CharBERT (Ma et al., 2020) infuse character-level features into BERT and pre-train the variant from scratch. This method can significantly improve the performance and robustness of BERT, but requires pre-training an adapted transformer on a large amount of data. Another work on combating spelling mistakes recommends placing a word corrector before downstream models (Pruthi et al., 2019), which is effective and reusable. The main weakness of this method is that an error generated by the word corrector propagates to downstream tasks. For example, converting *"aleph"* to *"alpha"* may break the meaning of a mathematical statement. And indeed, using the word corrector consistently leads to a drop (0.5-2.0 percentage points) in BERT's performance on the SST dataset (Socher et al., 2013).

2.3 Contrastive Learning

The origin of contrastive learning can be traced back to the work by Becker and Hinton (1992) and Bromley et al. (1993). This method has achieved outstanding success in self-supervised representation learning for images (Oord et al., 2018; Hjelm et al., 2018; He et al., 2020; Chen et al., 2020; Grill et al., 2020). The contrastive learning framework learns representations from unlabeled data by pulling positive pairs together and pushing negative pairs apart. For training, the positive pairs are often obtained by taking two randomly augmented versions of the same sample and treating the other augmented examples within a mini-batch as negative examples (Chen et al., 2017, 2020). The most widely used loss is the infoNCE loss (or contrastive loss) (Hjelm et al., 2018; Logeswaran and Lee, 2018; Chen et al., 2020; He et al., 2020). Although many approaches adopt contrastive learning to represent sentences (Giorgi et al., 2020; Wu et al., 2020; Gao et al., 2021), it has so far not been applied to word representations.

spell

	Input	Encoder	Loss
MIMICK (2017)	character sequence `{s,p,e,l,l}`	RNNs	$\mathcal{L}_{\text{dis}}$
BoS (2018)	n-gram subword `{spe,pel,ell}`	SUM	$\mathcal{L}_{\text{dis}}$
KVQ-FH (2019)	adapted n-gram subword `{spe,pel,ell}`	Attention	$\mathcal{L}_{\text{dis}}$

Table 1: Details of different mimick-like models, with the word `spell` as an example.

3 Preliminaries

3.1 Mimick-like Model

Given pre-trained word embeddings, and given an OOV word, the core idea of MIMICK (Pinter et al., 2017) is to impute an embedding for the OOV word using the surface form of the word, so as to mimic the behavior of the known embeddings. BoS (Zhao et al., 2018), KVQ-FH (Sasaki et al., 2019), Robust Backed-off Estimation (Fukuda et al., 2020), and PBoS (Jinman et al., 2020) work similarly, and we refer to them as mimick-like models.

Formally, we have a fixed-size vocabulary set $\mathcal{V}$, with an embedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times m}$, in which each row is a word vector $\mathbf{u}_w \in \mathbb{R}^m$ for the word w. A mimick-like model aims to impute a vector $\mathbf{v}_w$ for an arbitrary word $w \notin \mathcal{V}$. The training objective of mimick-like models is to minimize the expected distance between $\mathbf{u}_w$ and $\mathbf{v}_w$ pairs:

$$\mathcal{L}_{\text{dis}} = \frac{1}{|\mathcal{V}|} \sum_{w \in \mathcal{V}} \psi(\mathbf{u}_w, \mathbf{v}_w) \tag{1}$$

mse

Here, $\psi(\cdot)$ is a distance function, e.g., the Euclidean distance $\psi = \|\mathbf{u}_w - \mathbf{v}_w\|_2^2$ or the cosine distance $\psi = 1 - \cos(\mathbf{u}_w, \mathbf{v}_w)$. The vector $\mathbf{v}_w$ is generated by the following equation:

$$\mathbf{v}_w = \phi(\zeta(w)), \text{ for } w \in \mathcal{V} \text{ or } w \notin \mathcal{V} \tag{2}$$

Here, $\zeta(\cdot)$ is a function that maps w to a list of subunits based on the surface form of the word (e.g., a character or subword sequence). After that, the sequence is fed into the function $\phi(\cdot)$ to produce vectors, and the inside structure can be CNNs, RNNs, or a simple summation function. After training, the model can impute vectors for arbitrary words. Table 1 shows some features of three mimick-like models.

word emb(spell

3.2 Contrastive Learning

Contrastive learning methods have achieved significant success for image representation (Oord et al., 2018; Chen et al., 2020). The core idea of these methods is to encourage learned representations for positive pairs to be close, while pushing representations from sampled negative pairs apart. The widely used contrastive loss (Hjelm et al., 2018; Logeswaran and Lee, 2018; Chen et al., 2020; He et al., 2020) is defined as:

$$\ell_{cl} = -\log \frac{e^{\text{sim}(\mathbf{u}_i^\intercal \mathbf{u}^+)/\tau}}{e^{\text{sim}(\mathbf{u}_i^\intercal \mathbf{u}^+)/\tau} + \sum e^{\text{sim}(\mathbf{u}_i^\intercal \mathbf{u}^-)/\tau}} \tag{3}$$

Here, τ is a temperature parameter, $\text{sim}(\cdot)$ is a similarity function such as cosine similarity, and (u_i, u^+), (u_i, u^-) are positive pairs and negative pairs, respectively (assuming that all vectors are normalized). During training, positive pairs are usually obtained by augmentation for the same sample, and negative examples are the other samples in the mini-batch. This process learns representations that are invariant against noisy factors to some extent.

4 Our Approach: LOVE

LOVE (Learning Out-of-Vocabulary Embeddings) draws on the principles of contrastive learning to maximize the similarity between target and generated vectors, and to push apart negative pairs. An overview of our framework is shown in Figure 3. It is inspired by work in visual representation learning (Chen et al., 2020), but differs in that one of the positive pairs is obtained from pre-trained embeddings instead of using two augmented versions. We adopt five novel types of word-level augmentations and a lightweight Positional Attention Module in this framework. Moreover, we find that adding hard negatives during training can effectively yield better representations. We removed the nonlinear projection head after the encoder layer, because its improvements are specific to the representation quality in the visual field. Furthermore, our approach is not an unsupervised contrastive learning framework, but a supervised learning approach.

Our framework takes a word from the original vocabulary and uses data augmentation to produce a corruption of it. For example, "`misspelling`" becomes "`mispelling`" after dropping one letter "`s`". Next, we obtain a target vector from the pre-trained embeddings for the original word, and we generate a vector for the corrupted word. These two vectors are a pair of positive samples, and we maximize the similarity between them while making the distance of negative pairs (other samples in the same mini-batch) as large as possible. As mentioned before, we use the contrastive loss as an objective function (Eq 3). There are five key ingredients in the framework that we will detail in the following (similar to the ones in Table 1): the Input Method, the Encoder, the Loss Function, our Data Augmentation, and the choice of Hard Negatives.

4.1 Input Method

Our goal is to use the surface form to impute vectors for words. The question is thus how to design the function $\zeta(\cdot)$ mentioned in Section 3.1 to represent each input word. MIMICK (Pinter et al., 2017) straightforwardly uses the character sequence (see Table 1). This, however, loses the information of morphemes, i.e., sequences of characters that together contribute a meaning. Hence, FastText (Bojanowski et al., 2017) adopts character n-grams. Such n-grams, however, are highly redundant. For example, if we use substrings of length 3 to 5 to represent the word `misspelling`, we obtain a list with 24 n-gram characters – while only the substrings {`mis, spell, ing`} are the three crucial units to understand the word. Hence, like BERT, we use WordPiece (Wu et al., 2016) with a vocabulary size of around 30000 to obtain meaningful subwords of the input word. For the word `misspelling`, this yields {`miss`, `##pel`, `##ling` }. However, if we just swap two letters (as by a typo), then the sequence becomes completely different: {`mi`, `##sp`, `##sell`, `##ing` }. Therefore, we use both the character sequence and subwords (Figure A1).

We shrink our vocabulary by stemming all words and keeping only the base form of each word, and by removing words with numerals. This decreases the size of vocabulary from 30 000 to 21 257 without degrading performance too much (Section A.1).

4.2 Encoder

Let us now design the function $\phi(\cdot)$ mentioned in Section 3.1. We are looking for a function that can encode both local features and global features. Local features are character n-grams, which provide robustness against minor variations such as character swaps or omissions. Global features combine local features regardless of their distance. For the word `misspelling`, a pattern of prefix and suffix `mis+ing` can be obtained by combining the local information at the beginning and the end of the word. Conventional CNNs, RNNs, and self-attention cannot extract such local and global information at the same time. Therefore, we design a new **Positional Attention Module**. Suppose we have an aforementioned mixed input sequence and a corresponding embedding matrix $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}|\times d}$ where d is the dimension of vectors. Then the input can be represented by a list of vectors: $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^{n\times d}$ where n is the

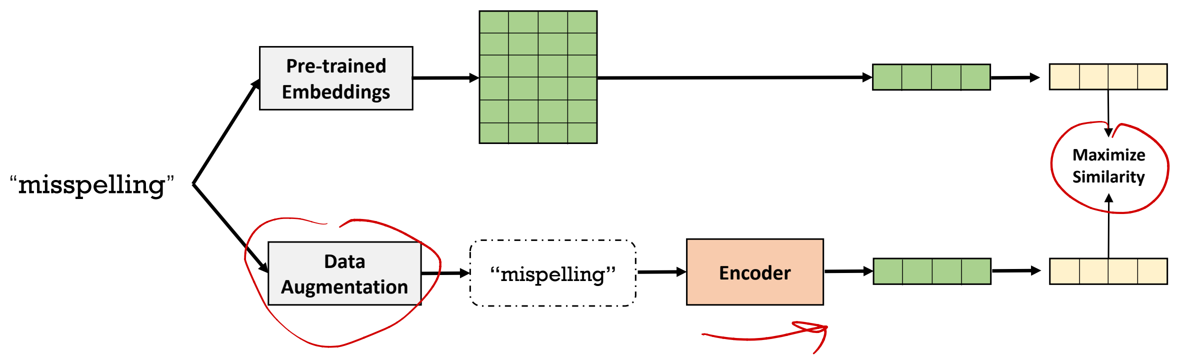


Figure 3: The framework of LOVE with an example of the word `misspelling`.

length of the input. To extract local information, we first adopt positional attention to obtain n-gram features, and then feed them into a conventional self-attention layer to combine them in a global way. This can be written as:

Encode
Rater Aug

$$\bar{\mathbf{X}} = \mathrm{SA}(\mathrm{PA}(\mathbf{X}))\,\mathbf{W}^O \tag{4}$$

Here, SA is a standard multi-head self-attention and PA is a positional attention. $\mathbf{W}^O \in \mathbb{R}^{d_V \times d_O}$ is a trainable parameter matrix, where d_V are the dimensions of values in SA and PA, and d_O is that of $\bar{\mathbf{X}}$. As for the Positional Attention, we adopt absolute sinusoidal embeddings (Vaswani et al., 2017) to compute positional correlations:

$$\mathrm{PA}(\mathbf{X}) = \mathrm{Softmax}\left(\frac{\mathbf{P}\mathbf{P}^\mathsf{T}}{\sqrt{d}}\right)(\mathbf{X}\,\mathbf{W}^V) \tag{5}$$

Here, $\mathbf{P} \in \mathbb{R}^{n \times d}$ are the position embeddings, and $\mathbf{W}^V \in \mathbb{R}^{d \times d_V}$ are the corresponding parameters. More details about the encoder are in Appendix C.4.

4.3 Loss Function

In this section, we focus on the loss function $\mathcal{L}(\cdot)$. Mimick-like models often adopt the mean squared error (MSE), which tries to give words with the same surface forms similar embeddings. However, the MSE only pulls positive word pairs closer, and does not push negative word pairs apart. Therefore, we use the contrastive loss instead (Equation 3). Wang and Isola (2020) found that the contrastive loss optimizes two key properties: *Alignment* and *Uniformity*. The Alignment describes the expected distance (closeness) between positive pairs:

$$\ell_{\text{align}} \triangleq \mathop{\mathbb{E}}_{(x,y)\sim p_{\text{pos}}} \psi(\mathbf{u}_x, \mathbf{u}_y) \tag{6}$$

Here, p_{pos} is the distribution of positive pairs. The Uniformity measures whether the learned representations are uniformly distributed in the hypersphere:

$$\ell_{\text{uniform}} \triangleq \log \mathop{\mathbb{E}}_{(x,y) \overset{i.i.d.}{\sim} p_{\text{data}}} e^{-t \cdot \psi(\mathbf{u}_x, \mathbf{u}_y)} \tag{7}$$

Here, p_{data} is the data distribution and $t > 0$ is a parameter. The two properties are consistent with our expected word representations: positive word pairs should be kept close and negative word pairs should be far from each other, finally scattered over the hypersphere.

4.4 Data Augmentation and Hard Negatives

Our positive word pairs are generated by data augmentation, which can increase the amount of training samples by using existing data. We use various strategies (Figure 4) to increase the diversity of our training samples: (1) Swap two adjacent characters, (2) Drop a character, (3) Insert a new character, (4) Replace a character according to keyboard distance, (5) Replace the original word by a synonymous word. The first four augmentations are originally designed to protect against adversarial attacks (Pruthi et al., 2019). We add the synonym replacement strategy to keep semantically similar words close in the embedding space – something that cannot be achieved by the surface form alone. Specifically, a set of synonyms is obtained by retrieving the nearest neighbors from pre-trained embeddings like FastText.

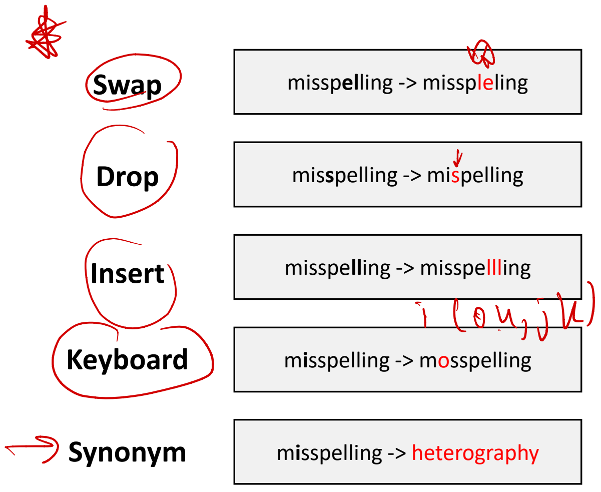

Swap
misspelling -> missleling
Drop
misspelling -> mispelling
Insert
misspelling -> misspellling
i (o,u,j,k)
Keyboard
misspelling -> mosspelling
Synonym
misspelling -> heterography

Figure 4: Illustrations of different augmentations for the word `misspelling`.

Negative word pairs are usually chosen randomly from the mini-batch. However, we train our model to be specifically resilient to *hard negatives (or difficult negatives)*, i.e., words with similar surface forms but different meanings (e.g., *misspelling* and *dispelling*). To this end, we add a certain number of hard negative samples (currently 3 of them) to the mini-batch, by selecting word pairs that are not synonyms and have a small edit distance.

4.5 Mimicking Dynamical Embeddings

Pre-trained Language Models (e.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019)) dynamically generate word representations based on specific contexts, which cannot be mimicked directly. To this end, we have two options: We can either learn the behavior of the *input embeddings* in BERT before the multi-layer attentions or mimic the static *distilled embeddings* (Bommasani et al., 2020; Gupta and Jaggi, 2021).

We use the BERT as an example to explain these two methods. Suppose we have a subword sequence after applying WordPiece to a sentence: $W = \{w_1, w_2, ..., w_n\}$. For the subword sequence W, BERT first represents it as a list of subword embeddings: $\mathbf{E}^{in} = \{\mathbf{e}_1^{sub}, \mathbf{e}_2^{sub}, ..., \mathbf{e}_n^{sub}\}$. We refer to this static representation as the Input Embedding of BERT, and we can use our model to mimic the behavior of this part. We call this method *mimicking input embeddings*. For ease of implementation, we learn only from the words that are not separated into pieces. After that step, BERT applies a multi-layer multi-head attention to the input embeddings E^{in}, which yields a contextual representation for each subword: $\mathbf{E}^{out} = \{\mathbf{e}_1^{out}, \mathbf{e}_2^{out}, ..., \mathbf{e}_n^{out}\}$. However, these contextual representations vary according to the input sentence and we cannot learn from them directly. Instead, we choose to mimic the distilled static embeddings from BERT, which are obtained by pooling (max or average) the contextual embeddings of the word in different sentences. We call this method *mimicking distilled embeddings*. The latter allows for better word representations, while the former does not require training on a large-scale corpus. Our empirical studies show that mimicking distilled embeddings performs only marginally better. Therefore, we decided to rather learn the input embeddings of BERT, which is simple yet effective

4.6 Plug and Play

One of the key advantages of our model is that it can be used as a plug-in for other models. For models with static word embeddings like FastText, one can simply use our model to generate vectors for unseen words. For models with dynamic word embeddings like BERT, if a single word is tokenized into several parts, e.g. `misspelling` = {`miss`, `##pel`, `##ling` }, we regard it as an OOV word. Then, we replace the embeddings of the subwords by a single embedding produced by our model before the attention layer. Our final enhanced BERT model has 768 dimensions and 16M parameters. Note that the BERT-base model has ~110M parameters and its distilled one has ~550M parameters.

5 Experiments

5.1 Evaluation Datasets

There are two main methods to evaluate word representations: Intrinsic and Extrinsic. Intrinsic evaluations measure syntactic or semantic relationships between words directly, e.g., word similarity in word clusters. Extrinsic evaluations measure the performance of word embeddings as input features to a downstream task, e.g., named entity recognition (NER) and text classification. Several studies have shown that there is no consistent correlation between intrinsic and extrinsic evaluation results (Chiu et al., 2016; Faruqui et al., 2016; Wang et al., 2019). Hence, we evaluate our representation by both intrinsic and extrinsic metrics. Specifically, we use 8 intrinsic datasets (6 word similarity and 2 word cluster tasks): RareWord (Luong et al., 2013), SimLex (Hill et al., 2015), MTurk (Halawi et al., 2012), MEN (Bruni et al., 2014), WordSim (Agirre

	parameters				Word Similarity				Word Cluster		Avg
	embedding	others	RareWord	SimLex	MTurk	MEN	WordSim	SimVerb	AP	BLESS	
FastText (2017)	969M	-	48.1	30.4	66.9	78.1	68.2	25.7	58.0	71.5	55.9
MIMICK (2017)	9M	517K	27.1	15.9	32.5	36.5	15.0	7.5	**59.3**	**72.0**	33.2
BoS (2018)	500M	-	**44.2**	27.4	55.8	65.5	53.8	22.1	41.8	39.0	43.7
KVQ-FH (2019)	12M	-	42.4	20.4	55.2	63.4	53.1	16.4	39.1	42.5	41.6
LOVE	6.3M	200K	42.2	**35.0**	**62.0**	**68.8**	**55.1**	**29.4**	53.2	51.5	**49.7**

Table 2: Performance on the intrinsic tasks, measured as Spearman's ρ and purity for word similarity and clustering. Best performance among the mimick-like models in bold, second-best underlined.

	parameters		**SST2**		**MR**		**CoNLL-03**		**BC2GM**		Avg
	embedding	others	original	+typo	original	+typo	original	+typo	original	+typo	
FastText (2017)	969M	-	82.3	60.5	73.3	62.2	86.4	66.3	71.8	53.4	69.5
Edit Distance	969M	-	-	67.4	-	68.3	-	76.2	-	66.6	-
MIMICK (2018)	9M	517K	69.7	62.3	73.6	61.4	68.0	65.2	56.6	56.7	64.2
BoS (2018)	500M	-	79.7	72.6	73.6	**69.5**	**79.5**	68.6	**66.4**	61.5	71.5
KVQ-FH (2019)	12M	-	77.8	71.4	72.9	66.5	73.1	**70.4**	46.2	53.5	66.5
LOVE	6.3M	200K	**81.4**	**73.2**	**74.4**	66.7	78.6	69.7	64.7	**63.8**	**71.6**

Table 3: Performance on the extrinsic tasks, measured as accuracy and F1 (five runs of different learning rates) for text classification and NER, respectively. Typos are generated by simulated errors of an OCR engine (Ma, 2019). The speed of producing word vectors with Edit Distance and LOVE is *380s/10K* words and *0.9s/10K* words, respectively.

et al., 2009), Simverb (Agirre et al., 2009), AP (Almuhareb, 2006) and BLESS (Baroni and Lenci, 2011). We use four extrinsic datasets (2 text classification and 2 NER tasks): SST2 (Socher et al., 2013), MR (Pang and Lee, 2005), CoNLL-03 (Sang and De Meulder, 2003) and BC2GM (Smith et al., 2008). It is worth noting that the RareWord dataset contains many long-tail words and the BC2GM is a domain-specific NER dataset. All data augmentations and typo simulations are implemented by NLPAUG[1]. Appendix B provides more details on our datasets and experimental settings.

5.2 Results on Intrinsic Tasks

Table 2 shows the experimental results on 8 intrinsic tasks. Compared to other mimick-like models, our model achieves the best average score across 8 datasets while using the least number of parameters. Specifically, our model performs best on 5 word similarity tasks, and second-best on the word cluster tasks. Although there is a gap between our model and the original FastText, we find our performance acceptable, given that our model is 100x times smaller.

5.3 Results on Extrinsic Tasks

Table 3 shows the results on four downstream datasets and their corrupted version. In this experiment, we introduce another non-trivial baseline: Edit Distance. For each corrupted word, we find the most similar word from a vocabulary using edit distance and then use the pre-trained vectors of the retrieved word. Considering the time cost, we only use the first 20K words appearing in FastText (2M words) as reference vocabulary.

The typo words are generated by simulating post-OCR errors. For the original datasets, our model obtains the best results across 2 datasets and the second-best on NER datasets compared to other mimick-like models. For the corrupted datasets, the performance of the FastText model decreases a lot and our model is the second best but has very close scores with BoS consistently. Compared to other mimick-like models, our model with 6.5M achieves the best average score. Although Edit Distance can effectively restore the original meaning of word, it is 400x times more time-consuming than our model.

5.4 Robustness Evaluation

In this experiment, we evaluate the robustness of our model by gradually adding simulated post-OCR typos (Ma, 2019). Table 4 shows the performances on SST2 and CoNLL-03 datasets. We observe that our model can improve the robustness of the original embeddings without degrading their performance. Moreover, we find our model can make FastText more robust compared to other commonly used methods against unseen words: a generic `UNK` token or character-level representation of neural networks. Figure 5 shows the robust-

[1] https://github.com/makcedward/nlpaug

Typo Probability	SST2 original	10%	30%	50%	70%	90%	CoNLL-03 original	10%	30%	50%	70%	90%	Avg
					Static Embeddings								
FastText	**82.3**	68.2	59.8	56.7	57.8	60.3	**86.4**	81.6	78.9	73.9	70.2	63.4	70.0
FastText + LOVE	82.1	**79.8**	**74.9**	**74.2**	**68.8**	**67.2**	86.3	**84.7**	**81.8**	**77.5**	**73.1**	**71.3**	**76.8**
					Dynamical Embeddings								
BERT	**91.5**	88.2	78.9	74.7	69.0	60.1	**91.2**	**89.8**	**86.2**	83.4	79.9	76.5	80.7
BERT + LOVE	**91.5**	**88.3**	**83.7**	**77.4**	**72.7**	**63.3**	89.9	88.3	86.1	**84.3**	**80.8**	**78.3**	**82.1**

Table 4: Robust evaluation (five runs of different learning rates) on text classification and NER under simulated post-OCR typos. We use uncased and cased BERT-base model for SST2 and CoNLL-03, respectively.

char, subword ตัวแทนของ word

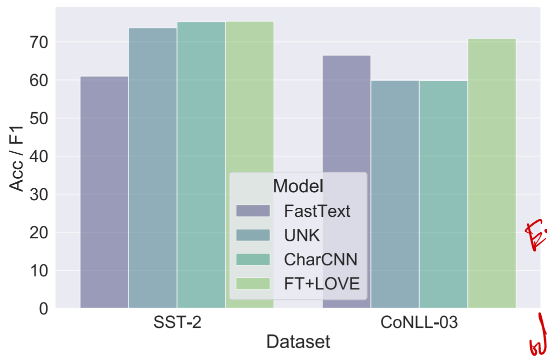


Figure 5: Evaluation of different methods based on FastText under typos.

	parameters embedding	parameters others	RareWord	SST2
The original LOVE	6.3M	200K	42.2	81.4
Varying the input method				
only use Char	299K	200K	17.7 ✓	71.5 ✓
only use Subword	6.0M	200K	25.3 ✓	76.0 ✓
Varying the encoder PA				
replace PAM with CNN	6.3M	270K	28.4	61.1
replace PAM with RNN	6.3M	517K	27.2	67.2
replace PAM with SA	6.3M	-	36.9	78.7 ✓
Varying the loss function				
use MSE ✓	6.3M	200K	34.5	76.0
use $\ell_{au}(\lambda = 1.0)$	6.3M	200K	40.8	80.8
Ablation of data augmentation and hard negatives				
w/out hard negatives	6.3M	200K	37.7	78.6
w/out hard negatives and augmentation	6.3M	200K	37.8	78.2

Encode

เปลี่ยน

algin+Uni

Table 5: **Ablation studies for the architecture of LOVE**, measured as Spearman's ρ and accuracy, respectively.

ness check of different strategies. FastText+LOVE has a consistent improvement on both SST2 and CoNLL-03 datasets. At the same time, LOVE degrades the performance on the original datasets only marginally if at all.

5.5 Ablation Study

We now vary the components in our architecture (input method, encoder and loss function) to demonstrate the effectiveness of our architecture.

Input Method. To validate the effect of our Mixed Input strategy, we compare it with two other methods: using only the character sequence or only the subword sequence. Table 5 shows that the Mixed method achieves better representations, and any removal of char or subword information can decrease the performance.

Encoder. To encode the input sequence, we developed the Positional Attention Module (PAM), which first extracts ngram-like local features and then uses self-attention combine them without distance restrictions. Table 5 shows that PAM performs the best, which validates our strategy of incorporating both local and global parts inside a word. At the same time, the number of parameters of PAM is acceptable in comparison. We visualize the attention weights of PAM in Appendix C.4, to show how the encoder extracts local and global morphological features of a word.

Loss Function. LOVE uses the contrastive loss, which increases alignment and uniformity. Wang and Isola (2020) proves that optimizing directly these two metrics leads to comparable or better performance than the original contrastive loss. Such a loss function can be written as:

$$\ell_{\text{au}} = \ell_{\text{align}} + \lambda \cdot \ell_{\text{uniform}} \quad (8)$$

Here, λ is a hyperparameter that controls the impact of ℓ_{uniform}. We set this value to 1.0 because it achieves the best average score on RareWord and SST2. An alternative is to use the Mean Squared Error (MSE), as in mimick-like models. Table 5 compares the performances of these different loss functions. The contrastive loss significantly outperforms the MSE, and there is no obvious improve-

LOVE + word Embe

typos per sentence	SST2 typo-0	typo-1	typo-2	typo-3
BERT	**91.5**	77.2	73.2	69.4
Mimicking Input Embeddings				
BERT + Add	91.3	77.2	73.5	70.7
BERT + Linear (2020)	91.4	79.6	77.2	72.8
BERT + Replacement	**91.5**	81.4	78.7	73.6
Mimicking Distilled Embeddings				
BERT + Add	91.3	78.8	75.6	72.3
BERT + Linear (2020)	91.3	81.4	78.7	73.6
BERT + Replacement	91.4	**81.5**	**78.9**	**73.8**

Table 6: Performances of different strategies that work with BERT together, measured as the accuracy among five different learning rates.

ment by directly using alignment and uniformity. We also tried various temperatures τ for the contrastive loss, and the results are shown in Table A3 in the appendix. In the end, a value of $\tau = 0.07$ provides a good performance.

Data Augmentation and Hard Negatives. In Table 5, we observe that the removal of our hard negatives decreases the performance, which demonstrates the importance of semantically different words with similar surface forms.

LOVE uses five types of word augmentation. We find that removing this augmentation does not deteriorate performance too much on the word similarity task, while it causes a 0.4 point drop in the text classification task (the last row in Table 5), where data augmentations prove helpful in dealing with misspellings. We further analyze the performance of single and composite augmentations on RareWord and SST2 in the appendix in Figure A3 and Figure A4. We find that a combination of all five types yields the best results.

5.6 The performance of mimicking BERT

As described in Section 4.5, we can mimic the input or distilled embeddings of BERT. After learning from BERT, we use the vectors generated by LOVE to replace the embeddings of OOV subwords. Finally, these new representations are fed into the multi-layer attentions. We call this method the *Replacement* strategy. To valid its effectiveness, we compare it with two other baselines: **(1) Linear Combination** (Fukuda et al., 2020). For each subword $\mathbf{e}^{sub}$, the generated vectors of word $\mathbf{e}^{word}$ containing the subwords are added to the subword vectors of BERT: linear

$$\mathbf{e}^{new} = (1 - \alpha)\, \mathbf{e}^{sub} + \alpha\, \mathbf{e}^{word} \tag{9}$$

$$\alpha = \text{sigmoid}\,(\mathbf{W} \cdot \mathbf{e}^{sub})$$

where $\mathbf{e}^{sub} \in \mathbb{R}^d$ is a subword vector of BERT, and $\mathbf{e}^{word} \in \mathbb{R}^d$ is a generated vector of our model. $\mathbf{W} \in \mathbb{R}^d$ are trainable parameters.
(2) Add. A generated word vector is directly added to a corresponding subword vector of BERT:

$$\mathbf{e}^{new} = \mathbf{e}^{sub} + \mathbf{e}^{word} \tag{10}$$

Table 6 shows the result of these strategies. All of them can bring a certain degree of robustness to BERT without decreasing the original capability, which demonstrates the effectiveness of our framework. Second, the replacement strategy consistently performs best. We conjecture that BERT cannot restore a reasonable meaning for those rare and misspelling words that are tokenized into subwords, and our generated vectors can be located nearby the original word in the space. Third, we find mimicking distilled embeddings performs the best while mimicking input embeddings comes close. Considering that the first method needs training on large-scale data, mimicking the input embeddings is our method of choice.

6 Conclusion

We have presented a lightweight contrastive-learning framework, LOVE, to learn word representations that are robust even in the face of out-of-vocabulary words. Through a series of empirical studies, we have shown that our model (with only 6.5M parameters) can achieve similar or even better word embeddings on both intrinsic and extrinsic evaluations compared to other mimick-like models. Moreover, our model can be added to models with static embeddings (such as FastText) or dynamical embeddings (such as BERT) in a plug-and-play fashion, and bring significant improvements there. For future work, we aim to extend our model to languages other than English.

7 Acknowledgements

We sincerely thank all the reviewers for their insightful comments and helpful suggestions. This work was partially funded by ANR-20-CHIA-0012-01 (“NoRDF”).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches.

Abdulrahman Almuhareb. 2006. *Attributes in lexical acquisition*. Ph.D. thesis, University of Essex.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.

Suzanna Becker and Geoffrey E Hinton. 1992. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163.

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

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.

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting pretrained contextualized representations via reductions to static embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of artificial intelligence research*, 49:1–47.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 18–26.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 767–776.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st workshop on evaluating vector-space representations for NLP*, pages 1–6.

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.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Nobukazu Fukuda, Naoki Yoshinaga, and Masaru Kitsuregawa. 2020. Robust backed-off estimation of out-of-vocabulary embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4827–4838.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

John M Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. 2020. Declutr: Deep contrastive learning for unsupervised textual representations. *arXiv preprint arXiv:2006.03659*.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

Prakhar Gupta and Martin Jaggi. 2021. Obtaining better static word embeddings using contextual embedding models. *arXiv preprint arXiv:2106.04302*.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Benjamin Heinzerling and Michael Strube. 2017. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. *arXiv preprint arXiv:1710.02187*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

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.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Zhao Jinman, Shawn Zhong, Xiaomin Zhang, and Yingyu Liang. 2020. Pbos: Probabilistic bag-of-subwords for generalizing word embedding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 596–611.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018. Learning to generate word representations using subword information. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2551–2561.

Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. 2018. Subword-level composition functions for learning word embeddings. In *Proceedings of the second workshop on subword/character level models*, pages 38–48.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4208–4215.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the seventeenth conference on computational natural language learning*, pages 104–113.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Charbert: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

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

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

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Jeffrey Pennington, Richard Socher, and Christopher D 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.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 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.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Édouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious 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 3226–3234.

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.

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.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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.

Timo Schick and Hinrich Schütze. 2019a. Attentive mimicking: Better word embeddings by attending to informative contexts. *arXiv preprint arXiv:1904.01617*.

Timo Schick and Hinrich Schütze. 2019b. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6965–6973.

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 AAAI Conference on Artificial Intelligence*, volume 34, 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, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):1–19.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

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

Ahmet Üstün, Murathan Kurfalı, and Burcu Can. 2018. Characters or morphemes: How to represent words? Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. 2019. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):1–9.

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.

Yi Zhu, Ivan Vulić, and Anna Korhonen. 2019. A systematic study of leveraging subword information for learning word representations. 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 912–932.

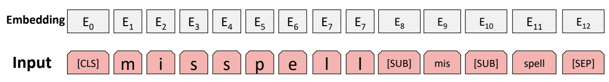


Figure A1: An illustration of our Mixed input for the word `misspell`.

Hyperparam	SST2	MR	CONLL-03	BC2GM
model	CNN	CNN	BiLSTM+CRF	BiLSTM+CRF
layer	1	1	1	1
kernel	[3,4,5]	[3,4,5]	-	-
filter	100	100	-	-
hidden size	300	300	300	300
optimizer	Adam	Adam	SGD	SGD
dropout	0.5	0.5	0.5	0.5
batch size	50	50	768	768
epoch	5	5	100	100

Table A2: Hyperparameters for extrinsic datasets.

A Details of Our Approach

A.1 Shrinking Our Model

We consider the following four methods to reduce the total parameters of our model:

(1) Matrix Decomposition. The original matrix can be decomposed into two smaller matrices $\mathbf{V} = \mathbf{U} \times \mathbf{M}, \mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times h}, \mathbf{M} \in \mathbb{R}^{h \times m}$ and $h < m$. Here, we set $m = 300$ and $h = 200$ respectively.

(2) Top Subword. We use only the top-k frequent subwords, using the word frequencies from a corpus. We set the parameter $k = 20000$.

(3) Hashing. We use a hashing strategy to share memory for subwords aiming to reduce the parameters. We use a bucket size of 20000.

(4) Preprocessing. The original vocabulary contains plurals and conjugations, therefore we stem all complete words and remove words with numerals, obtaining a new vocabulary of 21257 words.

Table A1 shows that the preprocessing method can reduce parameters very effectively while obtaining a very competitive performance.

	parameters		RareWord	SST2
	embedding	non-embedding		
Original	9M	200K	43.5	80.7
Decomposition	5.6M	200K	38.1	80.3
Top-K	6M	200K	39.2	80.1
Hashing	6M	200K	40.5	80.4
Preprocessing	6.3M	200K	**42.4**	**80.7**

Table A1: Performance of different shrinkage strategies, measured as Spearman's ρ and accuracy, respectively. The target vectors are from `fasttext-crawl-300d-2M`.

B Details of Our Experiments

B.1 Training of Mimick-like Models

Our target pre-trained embeddings are those from FastText `fasttext-crawl-300d-2M`, because they provide a strong baseline. They sum up subword-level information to produce vectors for arbitrary words. We also compare to MIMICK, BoS, and KVQ-FH, which do not train on contextual words. We do not compare to Robust Backed-off Estimation (Fukuda et al., 2020) and PBoS (Jinman et al., 2020), because they need larger and more complex models. Robust Backed-off Estimation uses string matching to find the top-k similar words from the entire vocabulary when imputing. Using the same target vectors, the number of parameter of BoS and PBoS are 163M and 316M, respectively. We re-train MIMICK, BoS, and KVQ-FH as baselines according to the published settings. In order to compare at the same parameter level, we use subwords for MIMICK instead of pure characters and adjust the hashing size $H = 40K$ for KVQ-FH.

B.2 Robustness Evaluations

As for our model, we first lower-case and tokenize each word by using WordPiece (Wu et al., 2016) with a vocabulary of $30K$ subwords and preprocess them by stemming and removing subwords with numerals. This yields a vocabulary of 21257 words. Each subword is represented by corresponding vectors from our model and we adopt a modified attention model to encode the subword sequence. Specifically, the layer number of this encoder is just 1 for efficiency and the hidden dimension is 300. In each block, the number of attention heads is 1 and we use fixed sinusoidal position embeddings (Vaswani et al., 2017) for positional information. To train the contrastive learning framework, we use the open-source tool (Ma, 2019) to augment a word, and use the probabilities $\{0.07, 0.07, 0.07, 0.07, 0.36, 0.36\}$ for six augmentations: swap, drop, insert, keyboard, synonym, no-operation. Hard negatives are generated by edit distance. For each target word, we store the top-100 similar words and insert 3 of them into a mini-batch as hard negatives. The loss function is a standard contrastive loss with temperature $\tau = 0.07$. The optimizer is Adam and the learning rate is 0.002. The dropout rate is 0.2 and we train the model for 20 epochs in total.

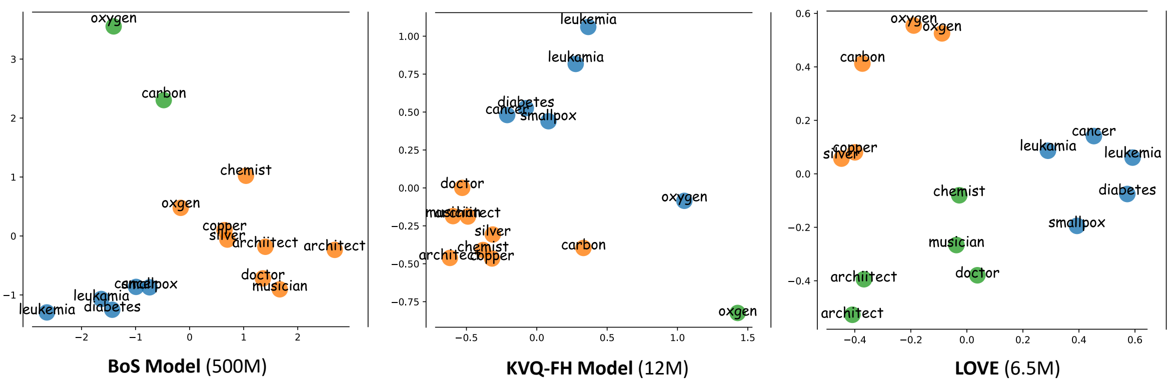


Figure A2: PCA visualizations of word vectors generated by LOVE, BoS, and KVQ-FH. Different colors mean different clusters, as predicted by K-means. There are three OOV words: `oxgen`, `archiitect` and `leukamia`.

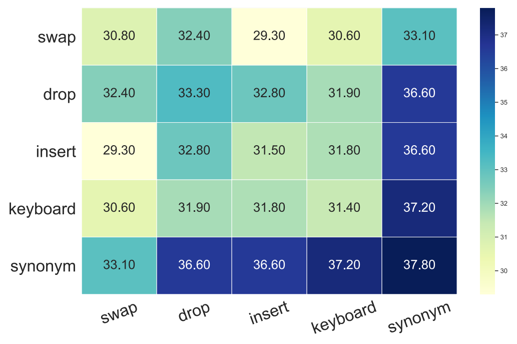


Figure A3: Performances of different augmentations on RareWord, measured as Spearman's ρ. Diagonal entries correspond to individual augmentation and off-diagonal entries correspond to composite augmentation.

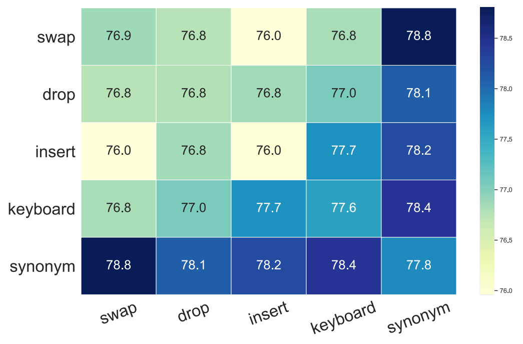


Figure A4: Performances of different augmentations on SST2, measured as accuracy. Diagonal entries correspond to individual augmentation and off-diagonal entries correspond to composite augmentation.

B.3 Intrinsic and Extrinsic Evaluations

We choose the setting discussed in Section 4 to train our model for 20 epochs, and evaluate each intrinsic task based on the vectors that the models produce. As for the extrinsic tasks, we feed word vectors into each neural network and fix them during training. We use CNNs for text classification (Zhang and Wallace, 2015) and BiLSTM+CRF for NER (Huang et al., 2015). We compare different embeddings on both intrinsic and extrinsic datasets by using generated vectors. For the word cluster tasks, the produced vectors are clustered by K-Means and then measured by Purity. The hyper-parameters of the extrinsic tasks are shown in Table A2. For each dataset, our model is trained with five learning rates $\{5e-3, 3e-3, 1e-3, 8e-4, 5e-4\}$. We select the best one on the development set to report its score on the test set.

To generate a corrupted dataset, we simulate post-OCR errors. We adopt the augmentation tool developed by Ma (2019) to corrupt 70% of the original words. To check the robustness of BERT, we directly finetune a BERT-base model using Huggingface (Wolf et al., 2020). During finetuning, the batch size is 16 and we train 5 epochs. We select the best model among five learning rates $\{9e-5, 7e-5, 5e-5, 3e-5, 1e-5\}$ on the development set and report the score of the model on the test set.

B.4 Datasets

Intrinsic Datasets. We use six word similarity datasets: (1) RareWord (Luong et al., 2013) (2) SimLex (Hill et al., 2015) (3) MTurk (Halawi et al., 2012) (4) MEN (Bruni et al., 2014) (5) Word-

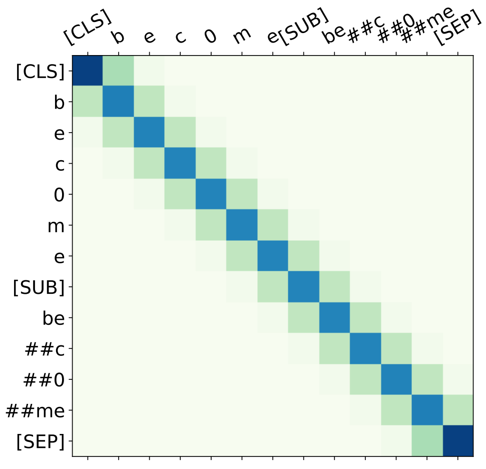


Figure A5: Visualization of positional weights for the post-OCR word `bec0me` (the correct one is `become`).

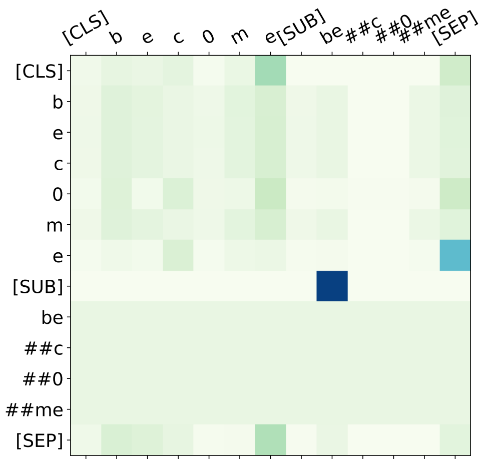


Figure A6: Visualization of self-attention weights for the post-OCR word `bec0me`.

Sim (Agirre et al., 2009), and (6) Simverb (Agirre et al., 2009). The task is scored by Spearman's ρ, which computes the correlation between gold similarity and the similarity obtained from generated vectors. For the word cluster task, we use (1) AP (Almuhareb, 2006) and (2) BLESS (Baroni and Lenci, 2011). The generated word vectors are first clustered by K-means (MacQueen et al., 1967) and then scored by the cluster purity.

Extrinsic Datasets. We use both sentence-level and token-level downstream datasets to evaluate the quality of word representations. For the sentence level, we use SST2 (Socher et al., 2013) and MR (Pang and Lee, 2005), and the metric is accuracy. For the token level, we use two NER datasets: general CoNLL-03 (Sang and De Meulder, 2003) and biomedical BC2GM (Smith et al., 2008). The metric is the entity-level F1 score. As before, we select the best model among five different learning rates $\{5e-3, 3e-3, 1e-3, 8e-4, 5e-4\}$ on the development set and then report the model score on the test set.

C Additional Analyses

C.1 Qualitative Analysis

To better understand the clusterings produced by LOVE, we chose 15 words from the AP dataset (Almuhareb, 2006), covering three topics (Chemical Substance, Illness, and Occupation). We added 3 corrupted words, `oxgen`, `archiitect` and `leukamia`. Figure A2 shows how LOVE, BoS, and KVQ-FH cluster these words (using a PCA projection and K-means). All approaches space out the clusters to some degree. In particular, BoS and KVQ-FH have trouble separating professions and chemical substances. For the corrupted words, only LOVE is able to embed them close enough to their original form, so that they appear in the correct cluster.

C.2 Effect of Augmentation for Text Classification

Figure A4 shows the performance of five augmentation strategies on the text classification task SST2. We observe that synonym is the most effective methods. The first four methods have a weaker effect, but the keyboard replacement can bring a certain degree of improvement. The results on RareWord are similar (Figure A3).

C.3 Effect of τ in Contrastive Loss

As discussed in Chen et al. (2020), a proper temperature can yield better representation in the visual area because τ is able to weigh the negatives by their relative hardness. As shown in Table A3, we attempt different values of temperature and find that there is no consistent τ that makes a model work well both on intrinsic and extrinsic datasets. Hence, we choose the best performer on average, i.e., $\tau = 0.07$.

C.4 Visualization of Encoder

As mentioned before, we combine two types of attention heads (self-attention and positional attention) to encode a subword sequence. Here, we vi-

	parameters		RareWord	SST2
	embedding	non-embedding		
ℓ_{cl} ($\tau = 0.03$)	9M	200K	35.0	**81.6**
ℓ_{cl} ($\tau = 0.07$)	9M	200K	39.8	81.3
ℓ_{cl} ($\tau = 0.12$)	9M	200K	**39.9**	81.1
ℓ_{cl} ($\tau = 0.20$)	9M	200K	37.6	81.5
ℓ_{cl} ($\tau = 0.50$)	9M	200K	38.3	80.6

Table A3: Performances of contrastive loss with various temperature τ, measured as Spearman's ρ and accuracy respectively.

sualize the attention weights on each side and show how they work. Figure A5 shows the position-dependent weights. We use sinusoidal functions to generate positional embeddings, and the weights are the dot product between these embeddings. We observe the positional weights tend to the left and right subwords in addition to themselves, which yields trigram representations.

Figure A6 shows the self-attention weights which are computed from the trigram subwords of positional attention. Hence, each subword in this figure is a trigram representation instead of a single subword representation. As we see, self-attention can capture global features regardless of distance. We take the first token `[CLS]` as an example, and this self-attention assigns high weights for the token `e` and `[SEP]`, which constructs a representation like this: `[CLS]b` + `me[SUB]` + `##me[SEP]`. This segment tells us this word starts with `b` and ends with `me`.

Mimicking Word Embeddings using Subword RNNs

Yuval Pinter **Robert Guthrie** **Jacob Eisenstein**

School of Interactive Computing
Georgia Institute of Technology
{uvp,rguthrie3,jacobe}@gatech.edu

Abstract

Word embeddings improve generalization over lexical features by placing each word in a lower-dimensional space, using distributional information obtained from unlabeled data. However, the effectiveness of word embeddings for downstream NLP tasks is limited by out-of-vocabulary (OOV) words, for which embeddings do not exist. In this paper, we present MIMICK, an approach to generating OOV word embeddings compositionally, by learning a function from spellings to distributional embeddings. Unlike prior work, MIMICK does not require re-training on the original word embedding corpus; instead, learning is performed at the type level. Intrinsic and extrinsic evaluations demonstrate the power of this simple approach. On 23 languages, MIMICK improves performance over a word-based baseline for tagging part-of-speech and morphosyntactic attributes. It is competitive with (and complementary to) a supervised character-based model in low-resource settings.

1 Introduction

One of the key advantages of word embeddings for natural language processing is that they enable generalization to words that are unseen in labeled training data, by embedding lexical features from large unlabeled datasets into a relatively low-dimensional Euclidean space. These low-dimensional embeddings are typically trained to capture distributional similarity, so that information can be shared among words that tend to appear in similar contexts.

However, it is not possible to enumerate the entire vocabulary of any language, and even large unlabeled datasets will miss terms that appear in later applications. The issue of how to handle these *out-of-vocabulary* (OOV) words poses challenges for embedding-based methods. These challenges are particularly acute when working with low-resource languages, where even unlabeled data may be difficult to obtain at scale. A typical solution is to abandon hope, by assigning a single OOV embedding to all terms that do not appear in the unlabeled data.

We approach this challenge from a quasi-generative perspective. Knowing nothing of a word except for its embedding and its written form, we attempt to learn the former from the latter. We train a recurrent neural network (RNN) on the character level with the embedding as the target, and use it later to predict vectors for OOV words in any downstream task. We call this model the MIMICK-RNN, for its ability to read a word's spelling and mimick its distributional embedding.

Through nearest-neighbor analysis, we show that vectors learned via this method capture both word-shape features and lexical features. As a result, we obtain reasonable near-neighbors for OOV abbreviations, names, novel compounds, and orthographic errors. Quantitative evaluation on the Stanford RareWord dataset (Luong et al., 2013) provides more evidence that these character-based embeddings capture word similarity for rare and unseen words.

As an extrinsic evaluation, we conduct experiments on joint prediction of part-of-speech tags and morphosyntactic attributes for a diverse set of 23 languages, as provided in the Universal Dependencies dataset (De Marneffe et al., 2014). Our model shows significant improvement across the board against a single *UNK*-embedding backoff method, and obtains competitive results against a supervised character-embedding model, which is trained end-to-end on the target task. In low-resource settings, our approach is particularly effective, and is complementary to supervised character embeddings trained from labeled data. The MIMICK-RNN therefore provides a useful new tool for tagging tasks in settings where there is limited labeled data. Models and code are available at `www.github.com/yuvalpinter/mimick`.

2 Related Work

Compositional models for embedding rare and unseen words. Several studies make use of morphological or orthographic information when training word embeddings, enabling the prediction of embeddings for unseen words based on their internal structure. Botha and Blunsom (2014) compute word embeddings by summing over embeddings of the morphemes; Luong et al. (2013) construct a recursive neural network over each word's morphological parse; Bhatia et al. (2016) use morpheme embeddings as a prior distribution over probabilistic word embeddings. While morphology-based approaches make use of meaningful linguistic substructures, they struggle with names and foreign language words, which include out-of-vocabulary morphemes. Character-based approaches avoid these problems: for example, Kim et al. (2016) train a recurrent neural network over words, whose embeddings are constructed by convolution over character embeddings; Wieting et al. (2016) learn embeddings of character n-grams, and then sum them into word embeddings. In all of these cases, the model for composing embeddings of subword units into word embeddings is learned by optimizing an objective over a large unlabeled corpus. In contrast, our approach is a post-processing step that can be applied to any set of word embeddings, regardless of how they were trained. This is similar to the "retrofitting" approach of Faruqui et al. (2015), but rather than smoothing embeddings over a graph, we learn a function to build embeddings compositionally.

Supervised subword models. Another class of methods learn task-specific character-based word embeddings within end-to-end supervised systems. For example, Santos and Zadrozny (2014) build word embeddings by convolution over characters, and then perform part-of-speech (POS) tagging using a local classifier; the tagging objective drives the entire learning process. Ling et al. (2015) propose a multi-level long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997), in which word embeddings are built compositionally from an LSTM over characters, and then tagging is performed by an LSTM over words. Plank et al. (2016) show that concatenating a character-level or bit-level LSTM network to a word representation helps immensely in POS tagging. Because these methods learn from labeled data, they can cover only as much of the lexicon as appears in their labeled training sets. As we show, they struggle in several settings: low-resource languages, where labeled training data is scarce; morphologically rich languages, where the number of morphemes is large, or where the mapping from form to meaning is complex; and in Chinese, where the number of characters is orders of magnitude larger than in non-logographic scripts. Furthermore, supervised subword models can be combined with MIMICK, offering additive improvements.

Morphosyntactic attribute tagging. We evaluate our method on the task of tagging word tokens for their morphosyntactic attributes, such as gender, number, case, and tense. The task of morpho-syntactic tagging dates back at least to the mid 1990s (Oflazer and Kuruöz, 1994; Hajič and Hladká, 1998), and interest has been rejuvenated by the availability of large-scale multilingual morphosyntactic annotations through the Universal Dependencies (UD) corpus (De Marneffe et al., 2014). For example, Faruqui et al. (2016) propose a graph-based technique for propagating type-level morphological information across a lexicon, improving token-level morphosyntactic tagging in 11 languages, using an SVM tagger. In contrast, we apply a neural sequence labeling approach, inspired by the POS tagger of Plank et al. (2016).

3 MIMICK Word Embeddings

We approach the problem of out-of-vocabulary (OOV) embeddings as a **generation** problem: regardless of how the original embeddings were created, we assume there is a generative wordform-based protocol for creating these embeddings. By training a model over the existing vocabulary, we can later use that model for predicting the embedding of an unseen word.

Formally: given a language $\mathcal{L}$, a vocabulary $\mathcal{V} \subseteq \mathcal{L}$ of size V, and a pre-trained embeddings table $\mathcal{W} \in \mathbb{R}^{V \times d}$ where each word $\{w_k\}_{k=1}^{V}$ is assigned a vector $\boldsymbol{e}_k$ of dimension d, our model is trained to find the function $f : \mathcal{L} \rightarrow \mathbb{R}^d$ such that the projected function $f|_{\mathcal{V}}$ approximates the assignments $f(w_k) \approx \boldsymbol{e}_k$. Given such a model, a new word $w_{k^*} \in \mathcal{L} \setminus \mathcal{V}$ can now be assigned an embedding $\boldsymbol{e}_{k^*} = f(w_{k^*})$.

Our predictive function of choice is a **Word Type Character Bi-LSTM**. Given a word with character sequence $w = \{c_i\}_1^n$, a forward-LSTM and a backward-LSTM are run over the corresponding character embeddings sequence $\{\boldsymbol{e}_i^{(c)}\}_1^n$. Let $\boldsymbol{h}_f^n$ represent the final hidden vector for the forward-LSTM, and let $\boldsymbol{h}_b^0$ represent the final hidden vector for the backward-LSTM. The word embedding is computed by a multilayer perceptron:

$$f(w) = \mathbf{O}_T \cdot g(\mathbf{T}_h \cdot [\boldsymbol{h}_f^n; \boldsymbol{h}_b^0] + \boldsymbol{b}_h) + \boldsymbol{b}_T, \quad (1)$$

where $\mathbf{T}_h, \boldsymbol{b}_h$ and $\mathbf{O}_T, \boldsymbol{b}_T$ are parameters of affine transformations, and g is a nonlinear elementwise function. The model is presented in Figure 1.

The training objective is similar to that of Yin and Schütze (2016). We match the predicted embeddings $f(w_k)$ to the pre-trained word embeddings e_{w_k}, by minimizing the squared Euclidean distance,

$$\mathcal{L} = \|f(w_k) - \boldsymbol{e}_{w_k}\|_2^2. \quad (2)$$

By backpropagating from this loss, it is possible to obtain local gradients with respect to the parameters of the LSTMs, the character embeddings, and the output model. The ultimate output of the training phase is the character embeddings matrix $\mathbf{C}$ and the parameters of the neural network: $\mathcal{M} = \{\mathbf{C}, \mathbf{F}, \mathbf{B}, \mathbf{T}_h, \boldsymbol{b}_h, \mathbf{O}_T, \boldsymbol{b}_T\}$, where $\mathbf{F}, \mathbf{B}$ are the forward and backward LSTM component parameters, respectively.

3.1 MIMICK Polyglot Embeddings

The pretrained embeddings we use in our experiments are obtained from Polyglot (Al-Rfou et al., 2013), a multilingual word embedding effort. Available for dozens of languages, each dataset contains 64-dimension embeddings for the 100,000 most frequent words in a language's training corpus (of variable size), as well as an *UNK* embedding to be used for OOV words. Even with this vocabulary size, querying words from respective UD corpora (train + dev + test) yields high

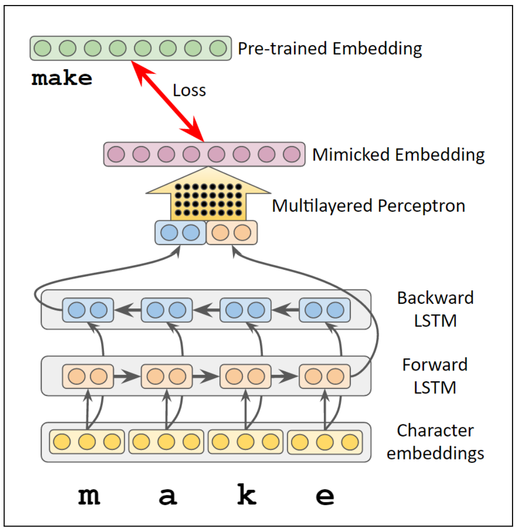


Figure 1: MIMICK model architecture.

OOV rates: in at least half of the 23 languages in our experiments (see Section 5), 29.1% or more of the word types do not appear in the Polyglot vocabulary. The token-level median rate is 9.2%.[1]

Applying our MIMICK algorithm to Polyglot embeddings, we obtain a prediction model for each of the 23 languages. Based on preliminary testing on randomly selected held-out development sets of 1% from each Polyglot vocabulary (with error calculated as in Equation 2), we set the following hyper-parameters for the remainder of the experiments: character embedding dimension = 20; one LSTM layer with 50 hidden units; 60 training epochs with no dropout; nonlinearity function $g = \tanh$.[2] We initialize character embeddings randomly, and use DyNet to implement the model (Neubig et al., 2017).

Nearest-neighbor examination. As a preliminary sanity check for the validity of our protocol, we examined nearest-neighbor samples in languages for which speakers were available: English, Hebrew, Tamil, and Spanish. Table 1 presents selected English OOV words with

[1]Some OOV counts, and resulting model performance, may be adversely affected by tokenization differences between Polyglot and UD. Notably, some languages such as Spanish, Hebrew and Italian exhibit **relational synthesis** wherein words of separate grammatical phrases are joined into one form (e.g. Spanish *del* = *de* + *el*, 'from the-masc.-sg.'). For these languages, the UD annotations adhere to the sub-token level, while Polyglot does not perform subtokenization. As this is a real-world difficulty facing users of out-of-the-box embeddings, we do not patch it over in our implementations or evaluation.

[2]Other settings, described below, were tuned on the supervised downstream tasks.

OOV word	Nearest neighbors	OOV word	Nearest neighbors
MCT	AWS OTA APT PDM SMP	compartmentalize	formalize rationalize discern prioritize validate
McNeally	Howlett Gaughan McCallum Blaney	pesky	euphoric disagreeable horrid ghastly horrifying
Vercellotti	Martinelli Marini Sabatini Antonelli	lawnmower	tradesman bookmaker postman hairdresser
Secretive	Routine Niche Turnaround Themed	developiong	compromising inflating shrinking straining
corssing	slicing swaying pounding grasping	hurtling	splashing pounding swaying slicing rubbing
flatfish	slimy jerky watery glassy wrinkle	expectedly	legitimately profoundly strangely energetically

Table 1: Nearest-neighbor examples for the English MIMICK model.

their nearest in-vocabulary Polyglot words computed by cosine similarity. These examples demonstrate several properties: (a) word shape is learned well (acronyms, capitalizations, suffixes); (b) the model shows robustness to typos (e.g., *develop**iong***, *c**or**ssing*); (c) part-of-speech is learned across multiple suffixes (*pesky – euphoric, ghastly*); (d) word compounding is detected (e.g., *lawnmower – bookmaker, postman*); (e) semantics are not learned well (as is to be expected from the lack of context in training), but there are surprises (e.g., *flatfish – slimy, watery*). Table 2 presents examples from Hebrew that show learned properties can be extended to nominal morphosyntactic attributes (gender, number – first two examples) and even relational syntactic subword forms such as genetive markers (third example). Names are learned (fourth example) despite the lack of casing in the script. Spanish examples exhibit word-shape and part-of-speech learning patterns with some loose semantics: for example, the plural adjective form *prenatales* is similar to other family-related plural adjectives such as *patrimoniales* and *generacionales*. Tamil displays some semantic similarities as well: e.g. *enjineer* ('engineer') predicts similarity to other professional terms such as *kalviyiyal* ('education'), *thozhilnutpa* ('technical'), and *iraanuva* ('military').

Stanford RareWords. The Stanford RareWord evaluation corpus (Luong et al., 2013) focuses on predicting word similarity between pairs involving low-frequency English words, predominantly ones with common morphological affixes. As these words are unlikely to be above the cutoff threshold for standard word embedding models, they emphasize the performance on OOV words.

For evaluation of our MIMICK model on the RareWord corpus, we trained the Variational Embeddings algorithm (VarEmbed; Bhatia et al., 2016) on a 20-million-token, 100,000-type Wikipedia corpus, obtaining 128-dimension word embeddings for all words in the test corpus. VarEmbed estimates a prior distribution over word embeddings, conditional on the morphological composition. For in-vocabulary words, a posterior is estimated from unlabeled data; for out-of-vocabulary words, the expected embedding can be obtained from the prior alone. In addition, we compare to FastText (Bojanowski et al., 2016), a high-vocabulary, high-dimensionality embedding benchmark.

The results, shown in Table 3, demonstrate that the MIMICK RNN recovers about half of the loss in performance incurred by the original Polyglot training model due to out-of-vocabulary words in the "All pairs" condition. MIMICK also outperforms VarEmbed. FastText can be considered an upper bound: with a vocabulary that is 25 times larger than the other models, it was missing words from only 44 pairs on this data.

4 Joint Tagging of Parts-of-Speech and Morphosyntactic Attributes

The Universal Dependencies (UD) scheme (De Marneffe et al., 2014) features a minimal set of 17 POS tags (Petrov et al., 2012) and supports tagging further language-specific features using attribute-specific inventories. For example, a verb in Turkish could be assigned a value for the evidentiality attribute, one which is absent from Danish. These additional morphosyntactic attributes are marked in the UD dataset as optional per-token attribute-value pairs.

Our approach for tagging morphosyntactic attributes is similar to the part-of-speech tagging model of Ling et al. (2015), who attach a projection layer to the output of a sentence-level bidirectional LSTM. We extend this approach to morphosyntactic tagging by duplicating this projection layer for each attribute type. The input to our multilayer perceptron (MLP) projection network is the hidden state produced for each token in the sentence by an underlying LSTM, and the output is

OOV word	Nearest neighbors
TTGFM '(s/y) will come true',	TPTVR '(s/y) will solve', TBTL '(s/y) will cancel', TSIR '(s/y) will remove'
GIAVMTRIIM 'geometric(m-pl)'$_2$	ANTVMIIM 'anatomic(m-pl)', GAVMTRIIM 'geometric(m-pl)'$_1$
BQFTNV 'our request'	IVFBIHM 'their(m) residents', XTAIHM 'their(m) sins', IRVFTV 'his inheritance'
RIC'RDSVN 'Richardson'	AVISTRK 'Eustrach', QMINQA 'Kaminka', GVLDNBRG 'Goldenberg'

Table 2: Nearest-neighbor examples for Hebrew (Transcriptions per Sima'an et al. (2001)). 's/y' stands for 'she/you-m.sg.'; subscripts denote alternative spellings, standard form being 'X'$_1$.

	Emb. dim	Vocab size	Polyglot in-vocab $N = 862$	All pairs $N = 2034$
VarEmbed	128	100K	41.9	25.5
Polyglot	64	100K	40.8	8.7
MIMICK	64	0	17.9	17.5
Polyglot +MIMICK	64	100K	40.8	27.0
Fasttext	300	2.51M		47.3

Table 3: Similarity results on the RareWord set, measured as Spearman's $\rho \times 100$. VarEmbed was trained on a 20-million token dataset, Polyglot on a 1.7B-token dataset.

attribute-specific probability distributions over the possible values for each attribute on each token in the sequence. Formally, for a given attribute a with possible values $v \in V_a$, the tagging probability for the i'th word in a sentence is given by:

$$\Pr(a_{w_i} = v) = (\text{Softmax}(\phi(\boldsymbol{h}_i)))_v \ , \quad (3)$$

with

$$\phi(\boldsymbol{h}_i) = \mathbf{O}_W^a \cdot \tanh(\mathbf{W}_h^a \cdot \boldsymbol{h}_i + \boldsymbol{b}_h^a) + \boldsymbol{b}_W^a, \quad (4)$$

where $\boldsymbol{h}_i$ is the i'th hidden state in the underlying LSTM, and $\phi(\boldsymbol{h}_i)$ is a two-layer feedforward neural network, with weights $\mathbf{W}_h^a$ and $\mathbf{O}_W^a$. We apply a softmax transformation to the output; the value at position v is then equal to the probability of attribute v applying to token w_i. The input to the underlying LSTM is a sequence of word embeddings, which are initialized to the Polyglot vectors when possible, and to MIMICK vectors when necessary. Alternative initializations are considered in the evaluation, as described in Section 5.2.

Each tagged attribute sequence (including POS tags) produces a loss equal to the sum of negative log probabilities of the true tags. One way to combine these losses is to simply compute the **sum loss**. However, many languages have large differences in sparsity across morpho-syntactic attributes, as apparent from Table 4 (rightmost column). We therefore also compute a **weighted sum loss**, in which each attribute is weighted by the proportion of training corpus tokens on which it is assigned a non-*NONE* value. Preliminary experiments on development set data were inconclusive across languages and training set sizes, and so we kept the simpler sum loss objective for the remainder of our study. In all cases, part-of-speech tagging was less accurate when learned jointly with morphosyntactic attributes. This may be because the attribute loss acts as POS-unrelated "noise" affecting the common LSTM layer and the word embeddings.

5 Experimental Settings

The morphological complexity and compositionality of words varies greatly across languages. While a morphologically-rich agglutinative language such as Hungarian contains words that carry many attributes as fully separable morphemes, a sentence in an analytic language such as Vietnamese may have not a single polymorphemic or inflected word in it. To see whether this property is influential on our MIMICK model and its performance in the downstream tagging task, we select languages that comprise a sample of multiple morphological patterns. Language family and script type are other potentially influential factors in an orthography-based approach such as ours, and so we vary along these parameters as well. We also considered language selection recommendations from de Lhoneux and Nivre (2016) and Schluter and Agić (2017).

As stated above, our approach is built on the Polyglot word embeddings. The intersection of the Polyglot embeddings and the UD dataset (version 1.4) yields 44 languages. Of these, many are under-annotated for morphosyntactic attributes; we select twenty-three sufficiently-tagged languages, with the exception of Indonesian.[3] Table 4 presents the selected languages and their typological properties. As an additional proxy for mor-

[3]Vietnamese has no attributes by design; it is a pure analytic language.

	Language	Branch	Script type	Morpho.	Tokens w/ attr.		Language	Branch	Script type	Morpho.	Tokens w/ attr.
vi	Vietnamese	Vietic	alphabetic*	Analytic	00.0%	fa	Persian	Iranian	consonantal	Agglutin.	65.4%
hu	Hungarian	Finno-Ugric	alphabetic	Agglutin.	83.6%	hi	Hindi	Indo-Aryan	alphasyllab.	Fusional	92.4%
id	Indonesian	Malayic	alphabetic	Agglutin.	—	lv	Latvian	Baltic	alphabetic	Fusional	69.2%
zh	Chinese	Sinitic	ideographic	Isolating	06.2%	el	Greek	Hellenic	alphabetic	Fusional	64.8%
tr	Turkish	Turkic	alphabetic	Agglutin.	68.4%	bg	Bulgarian	Slavic	alphabetic	Fusional	68.6%
kk	Kazakh	Turkic	alphabetic	Agglutin.	20.9%	ru	Russian	Slavic	alphabetic	Fusional	69.2%
ar	Arabic	Semitic	consonantal	Fusional	60.6%	cs	Czech	Slavic	alphabetic	Fusional	83.2%
he	Hebrew	Semitic	consonantal	Fusional	62.9%	es	Spanish	Romance	alphabetic	Fusional	67.1%
eu	Basque	Vasconic	alphabetic	Agglutin.	59.2%	it	Italian	Romance	alphabetic	Fusional	67.3%
ta	Tamil	Tamil	syllabic	Agglutin.	78.8%	ro	Romanian	Romance	alphabetic	Fusional	87.1%
						da	Danish	Germanic	alphabetic	Fusional	72.2%
						en	English	Germanic	alphabetic	Analytic	72.8%
						sv	Swedish	Germanic	alphabetic	Analytic	73.4%

Table 4: Languages used in tagging evaluation. Languages on the right are Indo-European. *In Vietnamese script, whitespace separates syllables rather than words.

phological expressiveness, the rightmost column shows the proportion of UD tokens which are annotated with any morphosyntactic attribute.

5.1 Metrics

As noted above, we use the UD datasets for testing our MIMICK algorithm on 23 languages[4] with the supplied train/dev/test division. We measure part-of-speech tagging by overall token-level accuracy.

For morphosyntactic attributes, there does not seem to be an agreed-upon metric for reporting performance. Dzeroski et al. (2000) report per-tag accuracies on a morphosyntactically tagged corpus of Slovene. Faruqui et al. (2016) report macro-averages of F1 scores of 11 languages from UD 1.1 for the various attributes (e.g., part-of-speech, case, gender, tense); recall and precision were calculated for the full set of each attribute's values, pooled together.[5] Agić et al. (2013) report separately on parts-of-speech and morphosyntactic attribute accuracies in Serbian and Croatian, as well as precision, recall, and F1 scores per tag. Georgiev et al. (2012) report token-level accuracy for exact all-attribute tags (e.g. 'Ncmsh' for "Noun short masculine singular definite") in Bulgarian, reaching a tagset of size 680. Müller et al. (2013) do the same for six other languages. We report **micro F1**: each token's value for each attribute is compared separately with the gold labeling, where a correct prediction is a matching non-*NONE* attribute/value assignment. Recall and precision are calculated over the entire set, with F1 defined as their harmonic mean.

[4]When several datasets are available for a language, we use the unmarked corpus.

[5]Details were clarified in personal communication with the authors.

5.2 Models

We implement and test the following models:

No-Char. Word embeddings are initialized from Polyglot models, with unseen words assigned the Polyglot-supplied *UNK* vector. Following tuning experiments on all languages with cased script, we found it beneficial to first back off to the lowercased form for an OOV word if its embedding exists, and only otherwise assign *UNK*.

MIMICK. Word embeddings are initialized from Polyglot, with OOV embeddings inferred from a MIMICK model (Section 3) trained on the Polyglot embeddings. Unlike the No-Char case, backing off to lowercased embeddings before using the MIMICK output did not yield conclusive benefits and thus we report results for the more straightforward no-backoff implementation.

CHAR→TAG. Word embeddings are initialized from Polyglot as in the No-Char model (with lowercase backoff), and appended with the output of a character-level LSTM updated during training (Plank et al., 2016). This additional module causes a threefold increase in training time.

Both. Word embeddings are initialized as in MIMICK, and appended with the CHAR→TAG LSTM.

Other models. Several non-Polyglot embedding models were examined, all performed substantially worse than Polyglot. Two of these are notable: a random-initialization baseline, and a model initialized from FastText embeddings (tested on English). FastText supplies 300-dimension embeddings for 2.51 million lowercase-only forms, and no *UNK* vector.[6] Both of these embedding models were attempted with and without CHAR→TAG concatenation. Another model, initialized from only MIMICK output embeddings, performed well only on the language with smallest Polyglot training corpus (Latvian). A Polyglot model where OOVs were initialized using an averaged embedding of all Polyglot vectors, rather than the supplied *UNK* vector, performed worse than our No-Char baseline on a great majority of the languages.

Last, we do not employ type-based tagset restrictions. All tag inventories are computed from the training sets and each tag selection is performed over the full set.

5.3 Hyperparameters

Based on development set experiments, we set the following hyperparameters for all models on all languages: two LSTM layers of hidden size 128, MLP hidden layers of size equal to the number of each attribute's possible values; momentum stochastic gradient descent with 0.01 learning rate; 40 training epochs (80 for 5K settings) with a dropout rate of 0.5. The CHAR→TAG models use 20-dimension character embeddings and a single hidden layer of size 128.

6 Results

We report performance in both low-resource and full-resource settings. Low-resource training sets were obtained by randomly sampling training sentences, without replacement, until a predefined token limit was reached. We report the results on the full sets and on $N = 5000$ tokens in Table 5 (part-of-speech tagging accuracy) and Table 6 (morphosyntactic attribute tagging micro-F1). Results for additional training set sizes are shown in Figure 2; space constraints prevent us from showing figures for all languages.

MIMICK as OOV initialization. In nearly all experimental settings on both tasks, across languages and training corpus sizes, the MIMICK embeddings significantly improve over the Polyglot *UNK* embedding for OOV tokens on both POS and morphosyntactic tagging. For POS, the largest margins are in the Slavic languages (Russian, Czech, Bulgarian), where word order is relatively free and thus rich word representations are imperative. Chinese also exhibits impressive improvement across all settings, perhaps due to the large character inventory ($>$ 12,000), for which a model such as MIMICK can learn well-informed embeddings using the large Polyglot vocabulary dataset, overcoming both word- and character-level sparsity in the UD corpus.[7] In morphosyntactic tagging, gains are apparent for Slavic languages and Chinese, but also for agglutinative languages — especially Tamil and Turkish — where the stable morpheme representation makes it easy for subword modeling to provide a type-level signal.[8] To examine the effects on Slavic and agglutinative languages in a more fine-grained view, we present results of multiple training-set size experiments for each model, averaged over five repetitions (with different corpus samples), in Figure 2.

MIMICK vs. CHAR→TAG. In several languages, the MIMICK algorithm fares better than the CHAR→TAG model on part-of-speech tagging in low-resource settings. Table 7 presents the POS tagging improvements that MIMICK achieves over the pre-trained Polyglot models, with and without CHAR→TAG concatenation, with 10,000 tokens of training data. We obtain statistically significant improvements in most languages, even when CHAR→TAG is included. These improvements are particularly substantial for test-set tokens outside the UD training set, as shown in the right two columns. While test set OOVs are a strength of the CHAR→TAG model (Plank et al., 2016), in many languages there are still considerable improvements to be obtained from the application of MIMICK initialization. This suggests that with limited training data, the end-to-end CHAR→TAG model is unable to learn a sufficiently accurate representational mapping from orthography.

7 Conclusion

We present a straightforward algorithm to infer OOV word embedding vectors from pre-trained,

[6]Vocabulary type-level coverage for the English UD corpus: 55.6% case-sensitive, 87.9% case-insensitive.

[7]Character coverage in Chinese Polyglot is surprisingly good: only eight characters from the UD dataset are unseen in Polyglot, across more than 10,000 unseen word types.

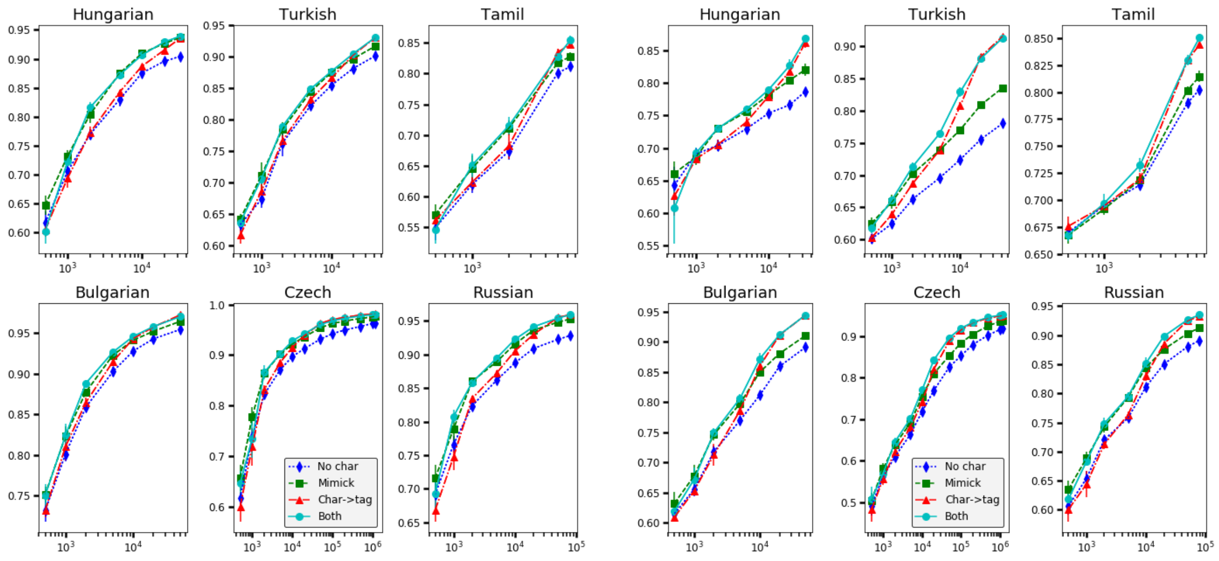
[8]Persian is officially classified as agglutinative but it is mostly so with respect to derivations. Its word-level inflections are rare and usually fusional.

	$N_{train} = 5000$				Full data					
	No-Char	MIMICK	CHAR →TAG	Both	N_{train}	No-Char	MIMICK	CHAR →TAG	Both	PSG 2016*
kk	—	—	—	—	4,949	81.94	83.95	83.64	84.88	
ta	82.30	81.55	84.97	85.22	6,329	80.44	**82.96**	84.11	84.46	
lv	80.44	**84.32**	84.49	**85.91**	13,781	85.77	**87.95**	89.55	89.99	
vi	85.67	*84.22*	84.85	85.43	31,800	89.94	90.34	90.50	90.19	
hu	82.88	**88.93**	85.83	**88.34**	33,017	91.52	**93.88**	94.07	93.74	
tr	83.69	**85.60**	84.23	**86.25**	41,748	90.19	**91.82**	93.11	92.68	
el	93.10	**93.63**	94.05	**94.64**	47,449	97.27	**98.08**	98.09	98.22	
bg	90.97	**93.16**	93.03	**93.52**	50,000	96.63	**97.29**	97.95	97.78	98.23
sv	90.87	**92.30**	92.27	**93.02**	66,645	95.26	**96.27**	96.69	96.87	96.60
eu	82.67	**84.44**	86.01	**86.93**	72,974	91.67	**93.16**	94.46	94.29	95.38
ru	87.40	**89.72**	88.65	**90.91**	79,772	92.59	**95.21**	95.98	95.84	
da	89.46	90.13	89.96	90.55	88,980	94.14	**95.04**	96.13	96.02	96.16
id	89.07	89.34	89.81	90.21	97,531	92.92	93.24	93.41	**93.70**	93.32
zh	80.84	**85.69**	81.84	**85.53**	98,608	90.91	**93.31**	93.36	93.72	
fa	93.50	93.58	93.53	93.71	121,064	96.77	**97.03**	97.20	97.16	97.60
he	90.73	**91.69**	91.93	91.70	135,496	95.65	**96.15**	96.59	96.37	96.62
ro	87.73	**89.18**	88.96	**89.38**	163,262	95.68	**96.72**	97.07	97.09	
en	87.48	**88.45**	88.89	88.89	204,587	93.39	**94.04**	94.90	94.70	95.17
ar	89.01	**90.58**	90.49	90.62	225,853	95.51	**95.72**	96.37	96.24	98.87
hi	87.89	87.77	87.92	88.09	281,057	96.31	96.45	96.64	96.61	96.97
it	91.35	**92.50**	92.45	**93.01**	289,440	97.22	97.47	97.76	97.69	97.90
es	90.54	**91.41**	91.71	91.78	382,436	94.68	94.84	95.08	95.05	95.67
cs	87.97	**90.81**	90.17	**91.29**	1,173,282	96.34	**97.62**	98.18	*97.93*	98.02

Table 5: POS tagging accuracy (UD 1.4 Test). **Bold** (*Italic*) indicates significant improvement (degradation) by McNemar's test, $p < .01$, comparing MIMICK to "No-Char", and "Both" to CHAR→TAG.
* For reference, we copy the reported results of Plank et al. (2016)'s analog to CHAR→TAG. Note that these were obtained on UD 1.2, and without jointly tagging morphosyntactic attributes.

	$N_{train} = 5000$				Full data			
	No-Char	MIMICK	CHAR →TAG	Both	No-Char	MIMICK	CHAR →TAG	Both
kk	—	—	—	—	21.48	20.07	28.47	20.98
ta	80.68	**81.96**	84.26	**85.63**	79.90	**81.93**	84.55	85.01
lv	56.98	**59.86**	64.81	**65.82**	66.16	66.61	76.11	75.44
hu	73.13	**76.30**	73.62	**76.85**	80.04	80.64	86.43	84.12
tr	69.58	**75.21**	75.81	**78.93**	78.31	**83.32**	91.51	90.86
el	86.87	*86.07*	86.40	**87.50**	94.64	**94.96**	96.55	**96.76**
bg	78.26	**81.77**	82.74	**84.93**	91.98	**93.48**	96.12	95.96
sv	82.09	**84.12**	85.26	**88.16**	92.45	**94.20**	96.37	**96.57**
eu	65.29	**66.00**	70.67	*70.27*	82.75	**84.74**	90.58	**91.39**
ru	77.31	**81.84**	79.83	**83.53**	88.80	**91.24**	93.54	93.56
da	80.26	**82.74**	83.59	82.65	92.06	**94.14**	96.05	95.96
zh	63.29	**71.44**	63.50	**74.66**	84.95	85.70	84.86	85.87
fa	84.73	**86.07**	85.94	81.75	95.30	**95.55**	96.90	96.80
he	75.35	68.57	81.06	75.24	90.25	**90.99**	93.35	93.63
ro	84.20	**85.64**	85.61	**87.31**	94.97	**96.10**	97.18	97.14
en	86.71	**87.99**	88.50	**89.61**	95.30	**95.59**	96.40	96.30
ar	84.14	84.17	81.41	*81.11*	94.43	**94.85**	95.50	95.37
hi	83.45	**86.89**	85.64	85.27	96.15	96.21	96.59	**96.67**
it	89.96	**92.07**	91.27	**92.62**	97.32	**97.80**	98.18	98.31
es	88.11	**89.81**	88.58	**89.63**	94.84	**95.44**	96.21	**96.84**
cs	68.66	**72.65**	71.02	**73.61**	91.75	**93.71**	95.29	95.31

Table 6: Micro-F1 for morphosyntactic attributes (UD 1.4 Test). **Bold** (*Italic*) type indicates significant improvement (degradation) by a bootstrapped Z-test, $p < .01$, comparing models as in Table 5. Note that the Kazakh (*kk*) test set has only 78 morphologically tagged tokens.



Part-of-speech tagging (accuracy) Morpho-syntactic attribute tagging (micro-F1)

Figure 2: Results on agglutinative languages (top) and on Slavic languages (bottom). X-axis is number of training tokens, starting at 500. Error bars are the standard deviations over five random training data subsamples.

Test set	Missing embeddings	Full vocabulary		OOV (UD)	
CHAR→TAG		w/o	with	w/o	with
Persian	2.2%	0.03	**0.41**	**0.83**	**0.81**
Hindi	3.8%	**0.59**	0.21	**3.61**	0.36
English	4.5%	**0.83**	0.25	**3.26**	0.49
Spanish	5.2%	0.33	-0.26	1.03	-0.66
Italian	6.6%	**0.84**	0.28	**1.83**	0.21
Danish	7.8%	0.65	**0.99**	**2.41**	**1.72**
Hebrew	9.2%	**1.25**	**0.40**	**3.03**	0.06
Swedish	9.2%	**1.50**	**0.55**	**4.75**	**1.79**
Bulgarian	9.4%	**0.96**	0.12	**1.83**	-0.11
Czech	10.6%	**2.24**	**1.32**	**5.84**	**2.20**
Latvian	11.1%	**2.87**	**1.03**	**7.29**	**2.71**
Hungarian	11.6%	**2.62**	**2.01**	**5.76**	**4.85**
Turkish	14.5%	**1.73**	**1.69**	**3.58**	**2.71**
Tamil*	16.2%	**2.52**	0.35	2.09	1.35
Russian	16.5%	**2.17**	**1.82**	**4.55**	**3.52**
Greek	17.5%	**1.07**	0.34	**3.30**	1.17
Indonesian	19.1%	**0.46**	0.25	**1.19**	0.75
Kazakh*	21.0%	2.01	1.24	**5.34**	**4.20**
Vietnamese	21.9%	0.53	**1.18**	1.07	**5.73**
Romanian	27.1%	**1.49**	**0.47**	**4.22**	**1.24**
Arabic	27.1%	**1.23**	**0.32**	**2.15**	0.22
Basque	35.3%	**2.39**	**1.06**	**5.42**	**1.68**
Chinese	69.9%	**4.19**	**2.57**	**9.52**	**5.24**

Table 7: Absolute gain in POS tagging accuracy from using MIMICK for 10,000-token datasets (all tokens for Tamil and Kazakh). **Bold** denotes statistical significance (McNemar's test,$p < 0.01$).

limited-vocabulary models, without need to access the originating corpus. This method is particularly useful for low-resource languages and tasks with little labeled data available, and in fact is task-agnostic. Our method improves performance over word-based models on annotated sequence-tagging tasks for a large variety of languages across dimensions of family, orthography, and morphology. In addition, we present a Bi-LSTM approach for tagging morphosyntactic attributes at the token level. In this paper, the MIMICK model was trained using characters as input, but future work may consider the use of other subword units, such as morphemes, phonemes, or even bitmap representations of ideographic characters (Costa-jussà et al., 2017).

8 Acknowledgments

We thank Umashanthi Pavalanathan, Sandeep Soni, Roi Reichart, and our anonymous reviewers for their valuable input. We thank Manaal Faruqui and Ryan McDonald for their help in understanding the metrics for morphosyntactic tagging. The project was supported by project HDTRA1-15-1-0019 from the Defense Threat Reduction Agency.

References

Željko Agić, Nikola Ljubešić, and Danijela Merkler. 2013. Lemmatization and morphosyntactic tagging of Croatian and Serbian. In *4th Biennial International Workshop on Balto-Slavic Natural Language Processing (BSNLP 2013)*.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 183–192, Sofia, Bulgaria.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In (**?**).

Marta R Costa-jussà, David Aldón, and José AR Fonollosa. 2017. Chinese–spanish neural machine translation enhanced with character and word bitmap fonts. *Machine Translation*, pages 1–13.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 4585–4592.

Saso Dzeroski, Tomaz Erjavec, and Jakub Zavrel. 2000. Morphosyntactic tagging of slovene: Evaluating taggers and tagsets. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16.

Georgi Georgiev, Valentin Zhikov, Petya Osenova, Kiril Simov, and Preslav Nakov. 2012. Feature-rich part-of-speech tagging for morphologically complex languages: Application to Bulgarian. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 492–502, Avignon, France.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 483–490.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Miryam de Lhoneux and Joakim Nivre. 2016. Ud treebank sampling for comparative parser evaluation. In *The Sixth Swedish Language Technology Conference (SLTC)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 322–332.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Kemal Oflazer and Ìlker Kuruöz. 1994. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*, pages 144–149. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In (**?**).

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In (**?**), pages 1818–1826.

Natalie Schluter and Željko Agić. 2017. Empirically sampling universal dependencies. In *The NoDaLiDa Workshop on Universal Dependencies (UDW 2017)*.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a tree-bank of modern hebrew text. *Traitement Automatique des Langues*, 42(2):247–380.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In (**?**).