Rare Words: A Major Problem for Contextualized Embeddings and How to Fix it by Attentive Mimicking

Timo Schick
Sulzer GmbH
Munich, Germany
timo.schick@sulzer.de

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Pretraining deep neural network architectures with a language modeling objective has brought large improvements for many natural language processing tasks. Exemplified by BERT, a recently proposed such architecture, we demonstrate that despite being trained on huge amounts of data, deep language models still struggle to understand rare words. To fix this problem, we adapt Attentive Mimicking, a method that was designed to explicitly learn embeddings for rare words, to deep language models. In order to make this possible, we introduce *one-token approximation*, a procedure that enables us to use Attentive Mimicking even when the underlying language model uses subword-based tokenization, i.e., it does not assign embeddings to all words. To evaluate our method, we create a novel dataset that tests the ability of language models to capture semantic properties of words without any task-specific fine-tuning. Using this dataset, we show that adding our adapted version of Attentive Mimicking to BERT does substantially improve its understanding of rare words.

1 Introduction

Distributed representations of words are a key component of natural language processing (NLP) systems. In particular, deep contextualized representations learned using an unsupervised language modeling objective (Peters et al. 2018) have led to large performance gains for a variety of NLP tasks. Recently, several authors have proposed to not only use language modeling for feature extraction, but to fine-tune entire language models for specific tasks (Radford et al. 2018; Howard and Ruder 2018). Taking up this idea, Devlin et al. (2019) introduced BERT, a bidirectional language model based on the Transformer (Vaswani et al. 2017) that has achieved a new state-of-the-art for several NLP tasks.

As demonstrated by Radford et al. (2019), it is possible for language models to solve a diverse set of tasks to some extent *without* any form of task-specific fine-tuning. This can be achieved by simply presenting the tasks in form of natural language sentences that are to be completed by the model. The very same idea can also be used to test how well a language model understands a given word: we can "ask" it for properties of that word using natural language. For example, a language model that understands the concept of "guilt" should be able to correctly complete the sentence "Guilt is the opposite of __." with the word "innocence".

Q: A *lime* is a __ .	**A:** lime, lemon, fruit
Q: A *bicycle* is a __ .	**A:** bicycle, motorcycle, bike
Q: A *kumquat* is a __ .	**A:** noun, horse, dog
Q: A *unicycle* is a __ .	**A:** structure, unit, chain

Table 1: Example queries and most probable outputs of BERT for frequent (top) and rare words (bottom)

The examples in Table 1 show that, according to this measure, BERT is indeed able to understand frequent words such as "lime" and "bicycle": it predicts, among others, that the former is a fruit and the latter is the same as a bike. However, it fails terribly for both "kumquat" and "unicycle", two less frequent words from the same domains. This poor performance raises the question whether deep language models generally struggle to understand rare words and, if so, how this weakness can be overcome.

To answer this question, we create a novel dataset containing queries like the ones shown in Table 1. This dataset consists of (i) natural language patterns such as

<W> is a __ .

where <W> is a placeholder for a word to be investigated, and (ii) corresponding pairs of keywords (<W>) and targets (fillers for __) obtained using semantic relations extracted from WordNet (Miller 1995).

Using this dataset, we show that BERT indeed fails to understand many rare words. To overcome this limitation, we propose to apply Attentive Mimicking (Schick and Schütze 2019a), a method that allows us to explicitly learn high-quality representations for rare words. A prerequisite for using this method is to have high-quality embeddings for as many words as possible, because it is trained to reproduce known word embeddings. However, many deep language models including BERT make use of byte-pair encoding (Sennrich, Haddow, and Birch 2015), WordPiece (Wu et al. 2016) or similar subword tokenization algorithms. Thus, many words are not represented by a single token but by a sequence of subword tokens and do not have their own embeddings.

To solve this problem, we introduce *one-token approximation* (OTA), a method that approximately infers what the embedding of an arbitrary word would look like if it were represented by a single token. While we apply this method only to BERT, it can easily be adapted for other language modeling architectures.

In summary, our contributions are as follows:

- We introduce *WordNet Language Model Probing* (WNLaMPro), a novel dataset for evaluating the ability of language models to understand specific words.
- Using this dataset, we show that the ability of BERT to understand words depends highly on their frequency.
- We present one-token approximation (OTA), a method that obtains an embedding for a multi-token word that has behavior similar to the sequence of its subword embeddings.
- We apply OTA and Attentive Mimicking (Schick and Schütze 2019a) to BERT and show that this substantially improves BERT's understanding of rare words.
  Our work is the first to successfully apply mimicking techniques to contextualized word embeddings.

2 Related Work

Using language modeling as a task to obtain contextualized representations of words was first proposed by Peters et al. (2018), who train a bidirectional LSTM (Hochreiter and Schmidhuber 1997) language model for this task and then feed the so-obtained embeddings into task-specific architectures. Several authors extend this idea by transferring not only word embeddings, but entire language modeling architectures to specific tasks (Radford et al. 2018; Howard and Ruder 2018; Devlin et al. 2019). Whereas the GPT model proposed by Radford et al. (2018) is strictly unidirectional (i.e., it looks only at the left context to predict the next word) and the ULMFiT method of Howard and Ruder (2018) uses a shallow concatenation of two unidirectional models, Devlin et al. (2019) design BERT as a deep bidirectional model using a Transformer architecture and a masked language modeling task.

There are roughly two types of approaches for explicitly learning high-quality embeddings of rare words: surface-form-based approaches and context-based approaches. The former use subword information to infer a word's meaning; this includes n-grams (Wieting et al. 2016; Bojanowski et al. 2017; Salle and Villavicencio 2018), morphemes (Lazaridou et al. 2013; Luong, Socher, and Manning 2013) and characters (Pinter, Guthrie, and Eisenstein 2017). On the other hand, context-based approaches take a look at the words surrounding a given rare word to obtain a representation for it (e.g., Herbelot and Baroni 2017; Khodak et al. 2018). Recently, Schick and Schütze (2019b) introduced the *form-context model*, combining both approaches by jointly using surface-form and context information. The form-context model and its *Attentive Mimicking* variant (Schick and Schütze 2019a) achieve a new state-of-the-art for high-quality representations of rare words.

Presenting tasks in the form of natural language sentences was recently proposed by McCann et al. (2018) as part of their *Natural Language Decathlon*, for which they frame ten different tasks as pairs of natural language questions and answers. They train models on triples of questions, contexts and answers in a supervised fashion. An alternative, completely unsupervised approach proposed by Radford et al. (2019) is to train a language model on a large corpus, present text specialized for a particular task and then let the model complete this text. They achieve good performance on tasks such as reading comprehension, machine translation and question answering – without any form of task-specific fine-tuning. We use this paradigm for constructing WNLaMPro.

Several existing datasets were designed to analyze the ability of word embeddings to capture semantic relations between words. For example, Baroni and Lenci (2011) compile the BLESS dataset that covers five different semantic relations (e.g., hyponymy) from multiple sources. Weeds et al. (2014) also create a dataset for semantic relations based on hypernyms and hyponyms using WordNet (Miller 1995). However, these datasets differ from WNLaMPro in two important respects. (i) They focus on frequent words by filtering out infrequent ones whereas we explicitly want to analyze rare words. (ii) They do not provide natural language patterns: they either directly evaluate (uncontextualized) word embeddings using a similarity measure such as cosine distance or they frame the task of identifying the relationship between two words as a supervised task.

3 Attentive Mimicking

3.1 Original Model

Attentive Mimicking (AM) (Schick and Schütze 2019a) is a method that, given a set of d-dimensional high-quality embeddings for frequent words, can be used to infer embeddings for infrequent words that are appropriate for the given embedding space. AM is an extension of the *form-context model* (Schick and Schütze 2019b).

The key idea of the form-context model is to compute two distinct embeddings per word, where the first one exclusively uses the word's surface-form and the other the word's contexts, i.e., sentences in which the word was observed. Given a word w and a set of contexts $\mathcal{C}$, the surface-form embedding $v^{\text{form}}_{(w,\mathcal{C})} \in \mathbb{R}^d$ is obtained by averaging over learned embeddings of all n-grams in w; the context embedding $v^{\text{context}}_{(w,\mathcal{C})} \in \mathbb{R}^d$ is the average over the known embeddings of all context words.

The final representation $v_{(w,\mathcal{C})}$ of w is then a weighted sum of form embeddings and transformed context embeddings:

$$v_{(w,\mathcal{C})} = \alpha \cdot A v^{\text{context}}_{(w,\mathcal{C})} + (1-\alpha) \cdot v^{\text{form}}_{(w,\mathcal{C})}$$

where A is a $d \times d$ matrix and α is a function of both embeddings, allowing the model to decide when to rely on the word's surface form and when on its contexts (see Schick and Schütze (2019b) for further details).

While the form-context model treats all contexts equally, AM extends it with a self-attention mechanism that is ap-

plied to all contexts, allowing the model to distinguish informative from uninformative contexts. The attention weight of each context is determined based on the idea that given a word w, two informative contexts C_1 and C_2 (i.e., contexts from which the meaning of w can be inferred) resemble each other more than two randomly chosen contexts in which w occurs. In other words, if many contexts for a word w are similar to each other, then it is reasonable to assume that they are more informative with respect to w than other contexts. Schick and Schütze (2019a) define the similarity between two contexts as

$$s(C_1, C_2) = \frac{(Mv_{C_1}) \cdot (Mv_{C_2})^\top}{\sqrt{d}}$$

with $M \in \mathbb{R}^{d \times d}$ a learnable parameter and v_C denotes the average of embeddings for all words in a context C. The weight of a context is then defined as

$$\rho(C) \propto \sum_{C' \in \mathcal{C}} s(C, C') \,.$$

with $\sum_{C \in \mathcal{C}} \rho(C) = 1$. This results in the final context embedding

$$v^{\text{context}}_{(w,\mathcal{C})} = \sum_{C \in \mathcal{C}} \rho(C) \cdot v_C$$

where again, v_C denotes the average of the embeddings of all words in a context C.

Similar to earlier models (e.g., Pinter, Guthrie, and Eisenstein 2017), the model is trained through *mimicking*. That is, we randomly sample words w and corresponding contexts $\mathcal{C}$ from a large corpus and, given w and $\mathcal{C}$, ask the model to mimic the original embedding of w, i.e., to minimize the squared Euclidean distance between the original embedding and $v_{(w,\mathcal{C})}$.

3.2 AM+CONTEXT

As we found in preliminary experiments that AM focuses heavily on the word's surface form – an observation that is in line with results reported by Schick and Schütze (2019b) –, in addition to the default AM configuration of Schick and Schütze (2019a), we investigate another configuration AM+CONTEXT, which pushes the model to put more emphasis on a word's contexts. This is achieved by (i) increasing the minimum number of sampled contexts for each training instance from 1 to 8 and (ii) introducing n-gram dropout: during training, we randomly remove 10% of all surface-form n-grams for each training instance.

4 One-Token Approximation

As AM is trained through mimicking, it must be given high-quality embeddings of many words to learn how to make appropriate use of form and context information. Unfortunately, as many deep language models make use of subword-based tokenization, they assign embeddings to comparably few words. To overcome this limitation, we introduce *one-token approximation* (OTA). OTA finds an embedding for a multi-token word or phrase w that is similar to the embedding that w would have received if it had been a single token. This allows us to train AM in the usual way by simply mimicking the OTA-based embeddings of multi-token words.

Let Σ denote the set of all characters and $\mathcal{T} \subset \Sigma^*$ the set of all tokens used by the language model. Furthermore, let $t : \Sigma^* \to \mathcal{T}^*$ be the tokenization function that splits each word into a sequence of tokens and $e : \mathcal{T} \to \mathbb{R}^d$ the model's token embedding function, which we extend to sequences of tokens in the natural way as $e([t_1, \ldots, t_n]) = [e(t_1), \ldots, e(t_n)]$.

We assume that the language model internally consists of l_{max} hidden layers and given a sequence of token embeddings $\boldsymbol{e} = [e_1, \ldots, e_n]$, we denote by $h^l_i(\boldsymbol{e})$ the contextualized representation of the i-th input embedding e_i at layer l. Given two additional sequences of left and right embeddings $\boldsymbol{\ell}$ and $\boldsymbol{r}$, we define

$$\tilde{h}^l_i(\boldsymbol{\ell}, \boldsymbol{e}, \boldsymbol{r}) = \begin{cases} h^l_i(\boldsymbol{\ell}; \boldsymbol{e}; \boldsymbol{r}) & \text{if } i \leq |\boldsymbol{\ell}| \\ h^l_{i+|\boldsymbol{e}|}(\boldsymbol{\ell}; \boldsymbol{e}; \boldsymbol{r}) & \text{if } i > |\boldsymbol{\ell}| \end{cases}$$

where $a; b$ denotes the concatenation of sequences a and b. That is, we "cut out" the sequence $\boldsymbol{e}$ and $\tilde{h}^l_i(\boldsymbol{\ell}, \boldsymbol{e}, \boldsymbol{r})$ is then the embedding of the i-th input at layer l, either from $\boldsymbol{\ell}$ (if position i is before $\boldsymbol{e}$) or from $\boldsymbol{r}$ (if position i is after $\boldsymbol{e}$).

To obtain an OTA embedding for an arbitrary word $w \in \Sigma^*$, we require a set of left and right contexts $\mathcal{C} \subset \mathcal{T}^* \times \mathcal{T}^*$. Given one such context $c = (\mathbf{t}_\ell, \mathbf{t}_r)$, the key idea of OTA is to search for the embedding $v \in \mathbb{R}^d$ whose influence on the contextualized representations of $\mathbf{t}_\ell$ and $\mathbf{t}_r$ is as similar as possible to the influence of w's original, multi-token representation on both sequences. That is, when we apply the language model to the sequences $s_1 = [e(\mathbf{t}_\ell); e(t(w)); e(\mathbf{t}_r)]$ and $s_2 = [e(\mathbf{t}_\ell); [v]; e(\mathbf{t}_r)]$, we want the contextualized representations of $\mathbf{t}_\ell$ and $\mathbf{t}_r$ in s_1 to be as similar as possible to those in s_2.

Formally, we define the *one-token approximation* of w as

$$\text{OTA}(w) = \arg\min_{v \in \mathbb{R}^n} \sum_{(\mathbf{t}_\ell, \mathbf{t}_r) \in \mathcal{C}} d(e(t(w)), [v] \mid e(\mathbf{t}_\ell), e(\mathbf{t}_r))$$

where

$$d(\boldsymbol{e}, \tilde{\boldsymbol{e}} \mid \boldsymbol{\ell}, \boldsymbol{r}) = \sum_{l=1}^{l_{\text{max}}} \sum_{i=1}^{|\boldsymbol{\ell}|+|\boldsymbol{r}|} d^l_i(\boldsymbol{e}, \tilde{\boldsymbol{e}} \mid \boldsymbol{\ell}, \boldsymbol{r})$$

$$d^l_i(\boldsymbol{e}, \tilde{\boldsymbol{e}} \mid \boldsymbol{\ell}, \boldsymbol{r}) = \|\tilde{h}^l_i(\boldsymbol{\ell}, \boldsymbol{e}, \boldsymbol{r}) - \tilde{h}^l_i(\boldsymbol{\ell}, \tilde{\boldsymbol{e}}, \boldsymbol{r}))\|^2 \,.$$

That is, given an input sequence $[\boldsymbol{\ell}; \boldsymbol{e}; \boldsymbol{r}]$, $d^l_i(\boldsymbol{e}, \tilde{\boldsymbol{e}} \mid \boldsymbol{\ell}, \boldsymbol{r})$ measures the influence of replacing $\boldsymbol{e}$ with $\tilde{\boldsymbol{e}}$ on the contextualized representation of the i-th word in the l-th layer.

As $d(\boldsymbol{e}, \tilde{\boldsymbol{e}} \mid \boldsymbol{\ell}, \boldsymbol{r})$ is differentiable with respect to $\tilde{\boldsymbol{e}}$, we can use gradient-based optimization to estimate $\text{OTA}(w)$. This idea resembles the approach of Le and Mikolov (2014) to infer paragraph vectors for sequences of arbitrary length.

With regards to the choice of contexts $\mathcal{C}$, we define two variants, both of which do not require any additional information: STATIC and RANDOM. For the STATIC variant, $\mathcal{C}$ consists of a single context

$$(\mathbf{t}_\ell, \mathbf{t}_r) = (\texttt{[CLS]}, \texttt{.[SEP]})$$

Key	Rel.	Targets
new	ANT	old
general	ANT	specific
local	ANT	global
book	HYP	product, publication, ...
basketball	HYP	game, ball, sport, ...
lingonberry	HYP	fruit, bush, berry, ...
samosa	COH+	pizza, sandwich, salad, ...
harmonium	COH+	brass, flute, sax, ...
immorality	COH+	crime, evil, sin, fraud, ...
simluation	COR	simulation
chepmistry	COR	chemistry
pinacle	COR	pinnacle

Table 2: Example entries from WNLaMPro

with [CLS] and [SEP] being BERT's classification and separation token, respectively. We use this particular context because in pretraining, BERT is exposed exclusively to sequences starting with [CLS] and ending with [SEP].

As the meaning of a word can often better be understood by looking at its interaction with other words, we surmise that OTA works better when we provide variable contexts in which different words occur. For this reason, we also investigate the RANDOM variant. In this variant, each pair $(\mathbf{t}_\ell, \mathbf{t}_r) \in \mathcal{C}$ is of the form

$$(\mathbf{t}_\ell, \mathbf{t}_r) = (\texttt{[CLS]}\ t_\ell,\ t_r\ .\ \texttt{[SEP]})$$

where t_ℓ and t_r are uniformly sampled tokens from $\mathcal{T}$, under the constraint that each of them represent an actual word.

5 WordNet Language Model Probing

In order to assess the ability of language models to understand words as a function of their frequency, we introduce the *WordNet Language Model Probing* (WNLaMPro) dataset.[1] This dataset consists of two parts:

- a set of triples (k, r, T) where k is a *keyword*, r is a *relation* and T is a set of *target words*;
- a set of *patterns* $P(r)$ for each relation r, where each pattern is a sequence of tokens that contains exactly one *keyword placeholder* <W> and one *target placeholder* __.

The dataset contains four different kinds of relations: ANTONYM (ANT), HYPERNYM (HYP), COHYPONYM+ (COH+) and CORRUPTION (COR). Examples of dataset entries for all relations are shown in Table 2; the set of patterns for each relation can be seen in Table 3.

We split the dataset into a development and a test set. For each relation, we randomly select 10% of all entries to be included in the development set; the remaining 90% form the test set. We purposefully do not provide a training set as WNLaMPro is meant to be used *without* task-specific fine-tuning. We also define three subsets based on keyword counts in WWC: WNLaMPro-RARE, containing all words that occur less than 10 times, WNLaMPro-MEDIUM, containing all words that occur 10 or more times, but less than 100 times, and WNLaMPro-FREQUENT, containing all remaining words. Statistics about the sizes of these subsets and the mean number of target words per relation are listed in Table 4.

For creating WNLaMPro, we use WordNet (Miller 1995) to obtain triples (k, r, T). To this end, we denote by $\mathcal{V}$ the vocabulary of all words that occur at least once in the Westbury Wikipedia Corpus (WWC) (Shaoul and Westbury 2010) and match the regular expression [a-z.-]*. The set of all tokens in the BERT vocabulary is denoted by $\mathcal{T}$. For all triplets, we restrict the set of target words to single-token words from $\mathcal{T}$. This allows us to measure BERT's performance for each keyword k without the conflating influence of rare or multi-subword words on the target side.

[1] The WNLaMPro dataset is publicly available at https://github.com/timoschick/am-for-bert

ANTONYM	HYPERNYM
<W> is the opposite of __ .	<W> is a __ .
<W> is not __ .	a <W> is a __ .
someone who is <W> is not __ .	"<W>" refers to a __ .
something that is <W> is not __ .	<W> is a kind of __ .
"<W>" is the opposite of "__" .	a <W> is a kind of __ .
CORRUPTION	**COHYPONYM+**
"<W>" is a misspelling of "__" .	<W> and __ .
"<W>" . did you mean "__" ?	"<W>" and "__" .

Table 3: Patterns for all relations of WNLaMPro. The indefinite article "a" used in the HYP patterns is replaced with "an" as appropriate.

	Subset Size			Mean Targets		
Rel.	R	M	F	R	M	F
ANT	41	59	266	1.0	1.0	1.0
HYP	1191	1785	4750	4.0	3.9	4.2
COH+	1960	2740	6126	26.0	26.0	25.0
COR	2880	–	–	1.0	–	–

Table 4: The number of entries and mean number of target words for the RARE (R), MEDIUM (M), and FREQUENT (F) subsets of WNLaMPro

5.1 Antonyms

For each adjective $w \in \mathcal{V}$, we collect all antonyms for its most frequent WordNet sense in a set A and, if $A \cap \mathcal{T} \neq \emptyset$, add $(w, \text{ANTONYM}, A \cap \mathcal{T})$ to the dataset.

5.2 Hypernyms

For each noun $w \in \mathcal{V}$, let H be the set of all hypernyms for its two most frequent senses. As direct hypernyms are sometimes highly specific (e.g., the hypernym of "dog" is "canine"), we include all hypernyms whose path distance to w is at most 3. To avoid the inclusion of very general terms such as "object" or "unit", we restrict H to hypernyms

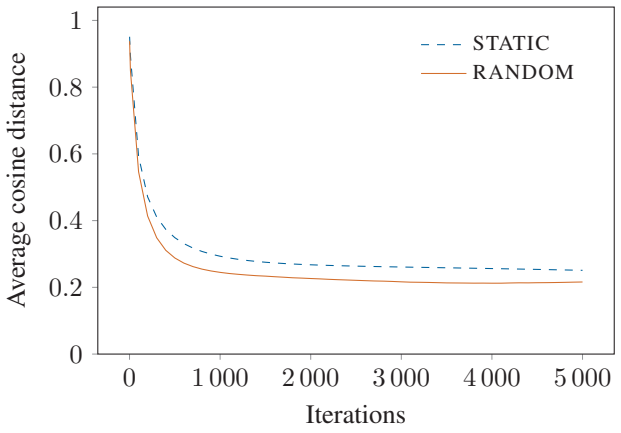

Figure 1: Performance of OTA on 1000 randomly selected one-token words

that have a minimum depth of 6 in the WordNet hierarchy. If $|H \cap \mathcal{T}| \geq 3$, we add $(w, \text{HYPERNYM}, H \cap \mathcal{T})$ to the dataset. However, if $|H \cap \mathcal{T}| > 20$, we keep only the 20 most frequent target words.

5.3 Cohyponyms+

For each noun $w \in \mathcal{V}$, we compute its set of hypernyms H as described above (but with a maximum path distance of 2), and denote by C the union of all hyponyms for each hypernym in H with a maximum path distance of 4.[2] Let $C' = (C \setminus \{w\}) \cap \mathcal{T}$. If $|C'| \geq 10$, we add the corresponding tuple $(w, \text{COHYPONYM+}, C')$ to the dataset. If $|C'| > 50$, we keep only the 50 most frequent target words.

5.4 Corruptions

We include this relation to investigate a model's ability to deal with corruptions of the input that may, for example, be the result of typing errors or errors in optical character recognition. To obtain corrupted words, we take frequent words from $\mathcal{V} \cap \mathcal{T}$ and randomly apply corruptions similar to the ones used by Hill, Cho, and Korhonen (2016) and Lee, Mansimov, and Cho (2018), but we apply them on the character level. Specifically, given a word $w = c_1 \dots c_n$, we create a corrupted version $\tilde{w}$ by either (i) inserting a random character c after a random position $i \in [0, n]$, (ii) removing a character at a random position $i \in [1, n]$ or (iii) switching the characters c_i and c_{i+1} for a random position $i \in [1, n-1]$. We then add $(\tilde{w}, \text{CORRUPTION}, w)$ to the dataset.

6 Experiments

For our evaluation of BERT on WNLaMPro, we use the Transformers library of Wolf et al. (2019). Our implementation of OTA is based on PyTorch (Paszke et al. 2017).[3] For all of our experiments involving AM, we use the original implementation of Schick and Schütze (2019a). As WNLaMPro is based on WordNet, all of our experiments are confined to the English language.

6.1 One-Token Approximation

We first compare the STATIC and RANDOM context variants of OTA and determine the optimal number of training iterations. To this end, we form a development set by randomly selecting 1000 one-token words from the BERT vocabulary. For each word w in this set, we measure the quality of its approximation $\text{OTA}(w)$ by comparing it to its BERT embedding $e(w)$, using cosine distance. We initialize the OTA vector of each word as a zero vector and optimize it using Adam (Kingma and Ba 2015) with an initial learning rate of 10^{-3}. For both context variants, we search for the ideal number of iterations in the range $\{100 \cdot i \mid 1 \leq i \leq 50\}$.

Results can be seen in Figure 1. While for both variants, the average cosine distance between BERT's embeddings and their OTA equivalents is relatively high in the beginning – which is simply due to the fact that all OTA embeddings are initialized randomly – after only a few iterations RANDOM consistently outperforms STATIC.[4] For the RANDOM variant, the average cosine distance reaches its minimum at 4000 iterations. We therefore use RANDOM contexts with 4000 iterations in our following experiments.

6.2 Evaluation on WNLaMPro

To measure the performance of a language model on WNLaMPro, we proceed as follows. Let $x = (k, r, T)$ be a dataset entry, $w \in T$ a target word, $p \in P(r)$ a pattern and $p[k]$ the same pattern where the keyword placeholder <W> is replaced by k. Furthermore, let $(a_1, \dots, a_n)$ be the model's responses (sorted in descending order by their probability) when it is asked to predict a replacement word for the target placeholder in $p[k]$. Then there is some j such that $a_j = w$. We denote with

$$\text{rank}(p[k], w) = j$$

$$\text{precision}_i(p[k], T) = \frac{|\{a_1, \dots, a_i\} \cap T|}{i}$$

the *rank of* w and *precision at* i when the model is queried with $p[k]$.[5] We may then define:

$$\text{rank}(x) = \min_{p \in P(r)} \min_{w \in T} \text{rank}(p[k], w)$$

$$\text{precision}_i(x) = \max_{p \in P(r)} \text{precision}_i(p[k], T)$$

That is, for each triplet x, we compute the *best* rank and precision that can be achieved using any pattern. We do so because our interest is not in testing the model's ability to understand a given pattern, but its ability to understand a given word: by letting the model choose the best pattern for

[2] Cohyponyms are defined to have a common parent. Our more general definition (having a common ancestor) gives us a test that has more coverage than a restriction to cohyponyms in a strict sense would have. We call our generalization "cohyponym+".

[3] Our implementation of OTA is publicly available at https://github.com/timoschick/one-token-approximation

[4] The difference between the best results achieved using RANDOM and STATIC is statistically significant in a two-sided binomial test ($p < 0.05$).

[5] We only look at the first 100 system responses and set $\text{rank}(p[k], w) = \infty$ if $w \notin \{a_1, \dots, a_{100}\}$.

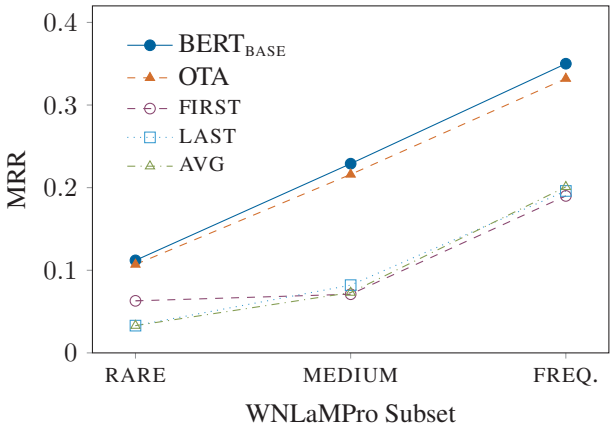

Figure 2: Mean reciprocal rank on WNLaMPro dev+test for BERT$_{\text{BASE}}$, OTA and various baselines

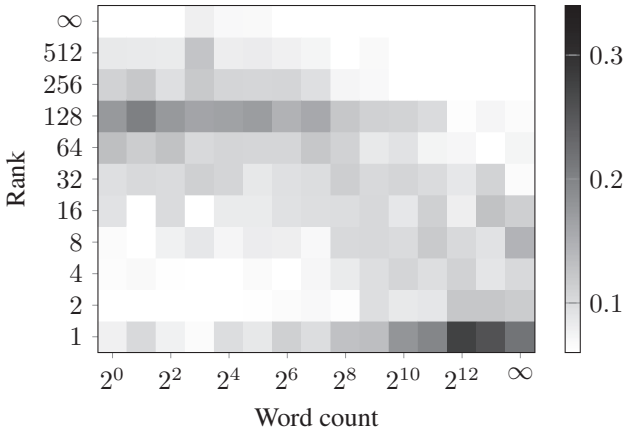

Figure 3: Performance of BERT$_{\text{BASE}}$ for the COHYPONYM+ subset of WNLaMPro. Each cell (i, j) of the heat map is shaded based on the percentage of all dataset entries with keyword counts ("Word count") in the range $(2^{j-1}, 2^j]$ whose rank ("Rank") is in the range $(2^{i-1}, 2^i]$. The values in each column add up to one.

each word, we minimize the probability that its response is of poor quality simply because it did not understand a given pattern.

We evaluate the uncased version of BERT$_{\text{BASE}}$ (Devlin et al. 2019) on WNLaMPro to get an impression of (i) the model's general ability to understand the presented phrases and (ii) the difference in performance for rare and frequent words. To investigate how well OTA does at obtaining single embeddings for multi-token words, we also try a variant of BERT where all multi-token keywords are replaced with their one-token approximations. Furthermore, we compare OTA against the following baseline strategies for obtaining single embeddings for multi-token words $w = t_1, \ldots, t_n$:

- FIRST: We use the embedding of the first token, $e(t_1)$.
- LAST: We use the embedding of the last token, $e(t_n)$.
- AVG: We use the average over the embeddings of all tokens, $\frac{1}{n}\sum_{i=1}^{n} e(t_i)$.

We choose these particular baselines because they are natural choices for obtaining a word embedding from a sequence of subword embeddings without any advanced computation.

The mean reciprocal rank (MRR) over WNLaMPro can be seen in Figure 2 for BERT$_{\text{BASE}}$, OTA and all baselines. We can see that for all models, the score depends heavily on the word frequency. Notably, OTA performs much better than all of the above baselines, regardless of word frequency. Furthermore, the difference in performance between OTA's single embeddings and BERT's original, multi-token embeddings is only marginal, allowing us to conclude that OTA is indeed able to infer single-token embeddings of decent quality for multi-token words.

Of course, OTA by itself does not improve the embedding quality compared to using BERT as is – and we never apply OTA to words that have single-token BERT representations in the following experiments. The purpose of OTA is to allow us to train our attentive mimicking model for BERT: OTA provides us with the single-token embeddings that we require to train AM.

	MRR	
Model	5 Epochs	10 Epochs
AM	0.258	0.253
AM+CONTEXT	**0.262**	**0.276**
AM − OTA	0.219	0.220
AM − form	0.138	0.133
AM − context	0.227	0.225

Table 5: Results on WNLaMPro dev for various configurations of AM trained on embeddings from and integrated into BERT$_{\text{BASE}}$

The general trend that the understanding of a word increases with its frequency becomes even more obvious when looking at Figure 3, where the distribution of ranks for the COHYPONYM+ subset of WNLaMPro is shown as a function of WWC word counts. The distribution of ranks is computed independently for each interval of word counts considered. That is, the values in each column are normalized so that they add up to one. This was done to prevent the diagram from being distorted because certain word count intervals contain more words than others. As can be seen, for words that occur at most 256 (2^8) times in WWC, the most probable rank interval is $[64, 128)$. With more observations, BERT's understanding of words drastically improves: more than 50% of all words with more than 256 (2^8) observations achieve a rank of at most 16.

6.3 Attentive Mimicking

We train two variants of Attentive Mimicking: the default configuration of Schick and Schütze (2019a) and the AM+CONTEXT configuration (§3.2) that puts more emphasis on contexts. To decide which method to apply and to

		RARE			MEDIUM			FREQUENT		
Set	**Model**	MRR	P@3	P@10	MRR	P@3	P@10	MRR	P@3	P@10
ANT	$\text{BERT}_{\text{BASE}}$	0.149	0.065	0.025	0.089	0.044	0.021	0.390	0.170	0.061
	$\text{BERT}_{\text{BASE}}$ + AM	**0.449**	**0.167**	**0.075**	**0.511**	**0.176**	**0.064**	**0.482**	**0.195**	**0.074**
	$\text{BERT}_{\text{LARGE}}$	0.234	0.083	0.044	0.218	0.088	0.036	0.541	0.209	0.081
	$\text{BERT}_{\text{LARGE}}$ + AM	**0.529**	**0.194**	**0.075**	**0.558**	**0.195**	**0.068**	**0.570**	**0.228**	**0.088**
HYP	$\text{BERT}_{\text{BASE}}$	0.276	0.122	0.066	0.327	0.151	0.077	**0.416**	**0.204**	**0.109**
	$\text{BERT}_{\text{BASE}}$ + AM	**0.300**	**0.135**	**0.074**	**0.343**	**0.158**	**0.081**	0.377	0.181	0.096
	$\text{BERT}_{\text{LARGE}}$	0.284	0.128	0.065	**0.350**	**0.169**	**0.086**	**0.462**	**0.226**	**0.117**
	$\text{BERT}_{\text{LARGE}}$ + AM	**0.299**	**0.137**	**0.074**	0.323	0.149	0.079	0.401	0.193	0.101
COH+	$\text{BERT}_{\text{BASE}}$	0.147	0.065	0.054	0.177	0.089	0.070	**0.294**	**0.150**	**0.116**
	$\text{BERT}_{\text{BASE}}$ + AM	**0.213**	**0.106**	**0.082**	**0.213**	**0.110**	**0.090**	0.262	0.136	0.108
	$\text{BERT}_{\text{LARGE}}$	0.174	0.085	0.067	0.210	**0.109**	**0.091**	**0.337**	**0.183**	**0.143**
	$\text{BERT}_{\text{LARGE}}$ + AM	**0.227**	**0.110**	**0.087**	**0.216**	0.106	0.089	0.292	0.153	0.121
COR	$\text{BERT}_{\text{BASE}}$	0.020	0.007	0.004	–	–	–	–	–	–
	$\text{BERT}_{\text{BASE}}$ + AM	**0.254**	**0.095**	**0.038**	–	–	–	–	–	–
	$\text{BERT}_{\text{LARGE}}$	0.062	0.022	0.012	–	–	–	–	–	–
	$\text{BERT}_{\text{LARGE}}$ + AM	**0.261**	**0.095**	**0.038**	–	–	–	–	–	–

Table 6: Performance of BERT with and without AM for WNLaMPro test, subdivided by relation and keyword count. Underlined numbers indicate a significant difference between BERT and BERT+AM in a two-sided binomial test ($p < 0.05$).

determine the optimal number of training epochs, we use WNLaMPro dev. As evaluating AM on WNLaMPro is a time-consuming operation, the only values we try are 5 and 10 epochs; furthermore, we perform hyperparameter optimization only on $\text{BERT}_{\text{BASE}}$. To understand the influence of one-token approximation on the performance of AM, in addition to the two configurations described above – both of which make use of OTA – we also try a variant without OTA, where the training set contains only one-token words. To see whether we actually need both form and context information, we additionally investigate the influence of dropping either the context or form parts of AM.

As proposed by Schick and Schütze (2019b), we train AM on all words that occur at least 100 times in WWC; for each word that is represented by multiple tokens in the BERT vocabulary, we use its OTA as a target vector to be mimicked. Importantly, we train AM on contexts from WWC (containing slightly fewer than 10^9 words), whereas the original BERT model was trained on the concatenation of BooksCorpus (Zhu et al. 2015) (containing $0.8 \cdot 10^9$ words) and a larger version of Wikipedia (containing $2.5 \cdot 10^9$ words). Each occurrence of a word can contribute to obtaining a high-quality representation, especially for rare words. Therefore, BERT has a clear advantage over our proposed method due to its larger training corpus.

Table 5 shows results for all model variants on WNLaMPro dev. We can see that OTA is indeed helpful for training the model, substantially improving its score. Results for the model variants using only form or context are in line with the findings of Schick and Schütze (2019b): it is essential for good performance to use both form and context. Furthermore, AM+CONTEXT improves upon the default configuration of AM and training it for 10 epochs performs better than 5 epochs. Based on these findings, we only apply AM+CONTEXT trained for 10 epochs using OTA on WNLaMPro test.

For both the base and large configurations of BERT, Table 6 compares BERT's performance with and without AM on WNLaMPro; MRR as well as precision at 3 and 10 are shown for each relation and frequency. AM substantially improves the score for rare words, both for $\text{BERT}_{\text{BASE}}$ and for $\text{BERT}_{\text{LARGE}}$. The difference between BERT with and without AM is significant according to a two-sided binomial test ($p < 0.05$). This demonstrates that AM helps BERT get a better understanding of rare words. The benefit of applying AM for medium frequency words depends largely on the model being used: for $\text{BERT}_{\text{LARGE}}$, using AM only brings a consistent improvement for the ANTONYM relation, whereas for $\text{BERT}_{\text{BASE}}$, using AM is always helpful. The fact that BERT performs better than AM for frequent words is not surprising, considering that our model both has less capacity and was trained on considerably less data. However, the strong results for rare and – in some cases – medium frequency words suggest that to obtain the best of both worlds, one can simply replace BERT's embeddings for rare words using AM while keeping its original embeddings for frequent words. As AM is trained using mimicking as an objective, embeddings induced by AM are well aligned with the embedding space it was trained on. Thus, BERT's original embeddings and AM-based embeddings can seamlessly be employed together.

To better understand for what kinds of words adding AM to BERT is especially helpful, we finally analyze the predictions of BERT with and without AM for a few selected words (Table 7). As exemplified by these examples, the inability of BERT to understand rare words is often

Query:	something that is *una·cc·ess·ible* is not __ .
BERT:	possible, impossible, true, allowed
BERT+AM:	accessible, allowed, possible, available
Query:	*un·ic·y·cle* and __ .
BERT:	bridge, body, base, chain
BERT+AM:	bicycle, pedestrian, walking, pedestrians
Query:	a *sal·si·fy* is a __ .
BERT:	cocktail, toilet, noun, boat
BERT+AM:	shrub, flower, plant, noun
Query:	" *resign·tai·on* " is a misspelling of " __ " .
BERT:	king, john, son, death
BERT+AM:	resignation, resign, resigned, resigning

Table 7: Example queries from WNLaMPro and most probable outputs of $\text{BERT}_{\text{BASE}}$ and BERT+AM. The tokenization of keywords used by BERT is indicated by · characters.

due to the tokenization algorithm splitting words in a suboptimal way ("una·cc·ess·ible" and "un·ic·y·cle" instead of "un·access·ible" and "uni·cycle"). As AM uses overlapping n-grams to represent a word's surface form and thus does not need to choose a single tokenization, it does not suffer from that problem. While BERT's tokenization problem could potentially also be addressed by replacing WordPiece with a morphology-aware tokenization algorithm, other words – such as "salsify" – simply cannot be decomposed into smaller meaningful units. BERT also struggles with spelling errors (e.g., "resigntaion") and rare spellings (e.g., "bulghur", "kidnaper").

7 Conclusion

We have introduced WNLaMPro, a new dataset that allows us to explicitly investigate the ability of language models to understand rare words. Using this dataset, we have shown that BERT struggles with words if they are too rare. To address this problem, we proposed to apply Attentive Mimicking (AM). For AM to work, we introduced one-token approximation (OTA), an effective method to obtain "single-token" embeddings for multi-token words. Using this method, we showed that AM is able to substantially improve BERT's understanding of rare words.

Future work might investigate whether more complex architectures than AM can bring further benefit to deep language models; it would also be interesting to see whether training AM on a larger corpus – such as the one used for training BERT by Devlin et al. (2019) – is beneficial. Furthermore, it would be interesting to see the impact of integrating AM on downstream tasks.

Acknowledgments

This work was funded by the European Research Council (ERC #740516). We would like to thank the anonymous reviewers for their helpful comments and their willingness to engage with our author response.

References

Baroni, M., and Lenci, A. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, 1–10. Association for Computational Linguistics.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Herbelot, A., and Baroni, M. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 304–309. Association for Computational Linguistics.

Hill, F.; Cho, K.; and Korhonen, A. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1367–1377. San Diego, California: Association for Computational Linguistics.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Howard, J., and Ruder, S. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 328–339. Melbourne, Australia: Association for Computational Linguistics.

Khodak, M.; Saunshi, N.; Liang, Y.; Ma, T.; Stewart, B.; and Arora, S. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12–22. Association for Computational Linguistics.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*.

Lazaridou, A.; Marelli, M.; Zamparelli, R.; and Baroni, M. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1517–1526. Association for Computational Linguistics.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, II–1188–II–1196. JMLR.org.

Lee, J.; Mansimov, E.; and Cho, K. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1173–1182. Brussels, Belgium: Association for Computational Linguistics.

Luong, T.; Socher, R.; and Manning, C. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 104–113.

McCann, B.; Keskar, N. S.; Xiong, C.; and Socher, R. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv* abs/1806.08730.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 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)*, 2227–2237. New Orleans, Louisiana: Association for Computational Linguistics.

Pinter, Y.; Guthrie, R.; and Eisenstein, J. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 102–112. Association for Computational Linguistics.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. Technical report.

Salle, A., and Villavicencio, A. 2018. Incorporating subword information into matrix factorization word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, 66–71. Association for Computational Linguistics.

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

Schick, T., and Schütze, H. 2019b. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Sennrich, R.; Haddow, B.; and Birch, A. 2015. Neural machine translation of rare words with subword units. *CoRR* abs/1508.07909.

Shaoul, C., and Westbury, C. 2010. The westbury lab wikipedia corpus.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

Weeds, J.; Clarke, D.; Reffin, J.; Weir, D.; and Keller, B. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2249–2259. Dublin City University and Association for Computational Linguistics.

Wieting, J.; Bansal, M.; Gimpel, K.; and Livescu, K. 2016. Charagram: Embedding words and sentences via character n-grams. *CoRR* abs/1607.02789.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; and Brew, J. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv* abs/1910.03771.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; Klingner, J.; Shah, A.; Johnson, M.; Liu, X.; Łukasz Kaiser; Gouws, S.; Kato, Y.; Kudo, T.; Kazawa, H.; Stevens, K.; Kurian, G.; Patil, N.; Wang, W.; Young, C.; Smith, J.; Riesa, J.; Rudnick, A.; Vinyals, O.; Corrado, G.; Hughes, M.; and Dean, J. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* abs/1609.08144.

Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, 19–27.