COIN – An Inexpensive and Strong Baseline for Predicting Out of Vocabulary Word Embeddings

Andrew Schneider
Near Miss Management
schneider@nearmissmgmt.com

Lihong He
IBM Research, Almaden
lihong.he@ibm.com

Zhijia Chen
Temple University
zhijia.chen@temple.edu

Arjun Mukherjee
University of Houston
arjun@cs.uh.edu

Eduard Dragut
Temple University
edragut@temple.edu

Abstract

Predicting word embeddings for out of vocabulary words remains an important challenge for NLP tools. Word embedding models only include terms that occur a sufficient number of times in their training corpora. Word embedding vector models attempt to approximate information about a word not in their vocabularies. We propose a fast method for predicting vectors for out of vocabulary terms that makes use of the surrounding terms of the unknown term and the hidden context layer of the word2vec model. We propose this method as a *strong baseline* in the sense that 1) while it does not surpass all state-of-the-art methods, it surpasses several techniques for vector prediction on benchmark tasks, 2) even when it underperforms, the margin is small retaining competitive performance in downstream tasks, and 3) it is inexpensive to compute, requiring no additional training stage. We also show that our technique can be incorporated into existing methods to achieve a new state-of-the-art on the word vector prediction problem.

1 Introduction

In recent years, distributive models of lexical semantics, i.e., word embedding models, have proven to be a very useful tool for representing natural language terms (i.e., words and common phrases such as 'New York') as real numbered vectors. These models, such as *word2vec* (Mikolov et al., 2013a,b), *GloVe* (Pennington et al., 2014), and *FastText* (Bojanowski et al., 2017), use a large corpus of documents to learn an embedded vector representation of lexical units based on their co-occurrence within sentences. Word embedding vectors capture semantic features of the terms, for example synonymous terms will be nearby in the embedded vector space. This makes word embeddings more powerful than other lexical representations such as one-hot vector encodings. Furthermore, distributive models are derived using unsupervised algorithms, which are efficient to run on large corpora. A major limitation, however, is that to derive a high quality word embedding vector for a term, it must occur more than a certain times in the training corpus (Bahdanau et al., 2018). Terms that do not meet this threshold are not included in the model's vocabulary.

When using a word embedding model, a user may encounter a large number of Out of Vocabulary (OOV) terms for various reasons. When we encounter OOV terms in a document, a common strategy is to simply ignore the unknown term. Alternately we can represent the term by a zero vector, or by the average of all known word vectors, but these strategies clearly do not extract any meaningful information from the OOV item.

A number of recent efforts propose techniques for predicting high quality word embedding vectors for OOV terms from the information available on the term *in situ* (Lazaridou et al., 2017; Herbelot and Baroni, 2017; Khodak et al., 2018; Li et al., 2017; Luong et al., 2013; Lazaridou et al., 2013; Schick and Schütze, 2019b). An important downside of these methods in practice is that they take an unpredictable amount of time to develop and deploy. A question arise in this context for deployment in an NLP pipeline: *Can one employ (and deploy) a simpler solution (with minimal accuracy loss) until one creates a more advanced solution?* The goal of this work is to seek answers to this question in the context of OOV terms.

In this work, we present a context-based method to predict the word embedding vectors for OOV terms. Our method makes use of context information that is learned internally in the course of training a word embedding model such as *word2vec*. We call this the **COIN** (**Co**ntext **In**formation) method. We advocate COIN as a *competitive baseline* in the following sense:

Accurate (compared to state-of-the-art models). The performance of COIN is close, both in direct comparison and downstream tasks, to state-of-the-

art models. We compare COIN with four such models (ADD, N2V, ALC, and FCM) on the benchmark tasks DefNonce and CRW; COIN vectors meet or exceed the accuracy of all except for FCM.

Inexpensive. Generating a COIN vector prediction requires only a pretrained word2vec model and runs on a regular personal laptop (without the requirement of GPU). The only computational cost of COIN is the equivalent of a few look-ups of word–vector mappings. It is able to generate high quality predicted word vectors for OOV terms from as few as one observed occurrence without any additional training. While the state-of-the-art method FCM requires to train a set of word vectors and then in a second step trains a model for OOV predictions.

Fast. COIN is much faster than other state-of-the-art models. It gives users immediate feedback (in seconds). While most supervised techniques for word embedding for OOV terms require hours of computation before a user can analyze it. In our experiments, the COIN method takes 131 seconds to generate the vectors of OOV terms for evaluation on the DefNonce task, and it only takes 21 seconds to run all of the 2, 4, and 6 sentence evaluations on the Chim task. In comparison, FCM takes nearly 7 hours of training time on the DefNonce task.

Balancing gains with cost. OOV words are important in downstream tasks (Chen et al., 2019; Conneau and Kiela, 2018; Garneau et al., 2019; Lourentzou et al., 2019; Serrà et al., 2017) and not as a standalone exercise. We report a study on 7 SentEval tasks, getting a .002 prediction difference between COIN at 73.9% and FCM at 74.1%. With the compute and energy demands of many modern NLP methods growing exponentially, one needs to consider weighing energy costs with the performance gains of a new model before deploying it (Henderson et al., 2020). In our setting, the gain of energy hungry methods is marginal compared to COIN. However, *many of the alternative approaches require a second step than training the initial word embedding model*. All of the training required for COIN is accomplished with the original word embedding step as a holistic model minimizing total parameters, improved performance and a better model design.

We can draw a parallel from daily life. Toyota Corolla is an excellent means of transportation for millions of people. Ferrari is another excellent means of transportation, but only available to the few who can afford it. One may view COIN (and methods in its class) as the "Corolla" among the methods of predicting embeddings for OOVs and accept that its performance may fall short of the "Ferrari's" (e.g., fastText, ELMo and Bert), one also needs to accept that the "Ferrari's" requires more resources: fuel (i.e., energy) and maintenance (i.e., hardware). Often scientists and engineers need a "Corolla" to deliver a proof of concept. There are times when their hardware infrastructures are ill-suited to run the "Ferrari's," e.g., lacking tens of GPUs. Our goal in this paper is to argue that there is a place for "Corolla's," even though we all seek shiny "Ferrari's."

COIN vectors are suitable on their own for word vector prediction: They can be used to initialize a trained method. We show that using COIN vectors as the first step in existing vector predicting systems can set a new state-of-the-art on benchmark tasks. We also notice that COIN vectors may reduce the training cost for those methods, improving the accuracy of the component techniques. We show these on FCM in Section 5.5.

In summary, the contributions of this paper are:

- propose COIN, a fast and inexpensive method for predicting a word vector for OOV terms based on context information.
- show that COIN is effective and efficient on word vector prediction tasks, working as a strong baseline and occasionally beating other state-of-the-art models.
- show that with COIN vectors as an initialization for existing techniques improves their accuracy.

2 Word Embeddings

Word embeddings are a technique for learning a vector representation of terms from a corpus based on their distributional features, that is, the contexts within which a term is observed (Bengio et al., 2003; Mikolov et al., 2013a,b; Pennington et al., 2014; Bojanowski et al., 2017). Word embedding techniques belong among a class of machine learning algorithms where the model is trained to minimize error on a dummy task and the true purpose of training is to extract an intermediate, internal representation of the input data. In the word embedding case, the dummy task is that of predicting the surrounding terms of a target term (or vice versa) and the true desired output is the internal representation of the input terms, i.e., the word embedding vectors. The various methods for learning word embeddings

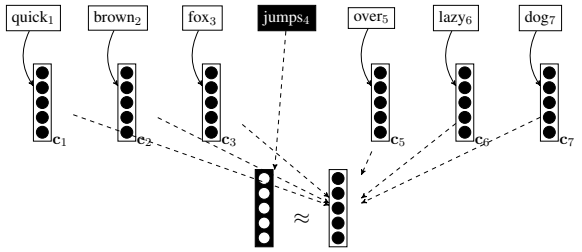

Figure 1: Embeddings maximize the conditional probability of the word vector of the target term ($\mathbf{w}_4$) given the combination of the context vectors of the context terms ($\mathbf{c_v}$). In this example, the window size k_w is 3.

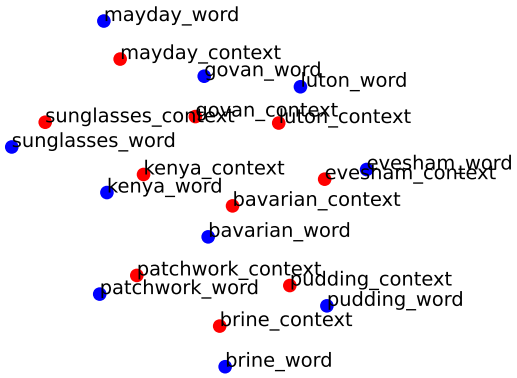

Figure 2: A t-SNE visualization of the relative locations of word and context vectors for a sample vocabulary. The word and context spaces are aligned using absolute orientation with translation and scaling (Dev et al., 2021). For most terms the word and context vectors are respectively well separated.

present variations on a core idea: each term is represented by two latent vectors: a *word vector* and a *context vector*. In training, the word embedding algorithm maximizes the conditional probability of the word vector given the set of context vectors of the terms found in its context window.

Word2vec is one of the most popular technique to learn word embeddings. The goal of training in word2vec is to maximize the log likelihood of the context terms given the target term:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{j\in[-k_w,k_w]\\ j\neq 0}} \log p(term_{t+j}|term_t)$$

For output $term_O$ and input $term_I$, $p(term_O|term_I)$ is defined using the softmax function:

$$p(term_O|term_I) = \frac{\exp(\mathbf{c}_O^\top \mathbf{w}_I)}{\sum_{w=1}^{W}\exp(\mathbf{c}_w^\top \mathbf{w}_I)}$$

Word2vec is one of the most popular technique to learn word embeddings (Mikolov et al., 2013a).

In *word2vec* model, the word vectors are not always close to the context vectors. In Figure 2 we show a visualization of the word and context vectors for some terms from the vocabulary. We find that the context and word vectors are quite far from each other for most terms. This general difference between vectors demonstrates that summing the context vectors will lead to a very different result from summing over the word vectors. This suggests that *context vectors encapsulate information that is not captured by the word vectors*, which may be useful to predict embeddding vectors for OOV terms. We aim to show this in this work. We note here that Figure 2 was generated after the word and context vector were aligned. We align the word and context vectors using absolute orientation with translation and scaling (Dev et al., 2021). Other orientation algorithms are available, but this gave the best average Cosine similarity between the vectors in our experiments. We also note that the context and word vectors as plotted in Figure 2 are even farther apart before alignment was performed.

Following our study of word embeddings, we observe that the word embedding vector $\mathbf{w}_i$ of a word is trained to be similar to the *context vectors* of the words in its context window, $\mathbf{c}_{j\in C_i}$, rather than the *word vectors*, which have previously been used in the prediction task. Intuition suggests that by using the context vectors of the context words we can improve the accuracy of our predicted vectors.

3 Related Work

Previous techniques to predict embedding vectors for OOV terms usually follow two theoretical lines: 1) Using morphological or form features of the OOV term, and 2) Extracting information from the surrounding context terms.

Form based methods in category 1) use some characteristics of the representation of the term, such as letter combinations. (Luong et al., 2013; Lazaridou et al., 2013) use morphological features. (Bojanowski et al., 2017) develops the *FastText* model, which is similar to word2vec in design, which operates on character n-grams rather than whole words. In some tasks, however, an OOV term may not have any character representation, or it may have an arbitrary representation, for example, the OOV term may be represented by underscores: "___" (Herbelot and Baroni, 2017) or by a made-up word form (Lazaridou et al., 2017). In the downstream task there may be no available form

for the term, such as "filling in the missing word."

For the context based works in category 2), (Lazaridou et al., 2017) develop the additive method, which takes the sum of the word vectors of the surrounding terms. The approach of (Herbelot and Baroni, 2017) initializes each OOV term as the sum of the word vectors of the known terms in the context sentences, downsampling frequent words, and then runs a modified word2vec training procedure on the OOV terms only, with a highly accelerated learning rate. (Khodak et al., 2018) finds a linear transformation $\mathbf{A}$ and uses it to transform the sum of the context word vectors to the word vector of OOV terms. Recently (Schick and Schütze, 2019a) propose a hybrid approach using both form and context. It learns a neural network model, which encodes information about the form (n-grams) and likelihood the form will contribute to an understanding of the meaning of the term, as well as a linear transformation similar to (Khodak et al., 2018). (Schick and Schütze, 2019a) further proposes a method to identify high quality contexts for inferring the OOV term vector.

COIN vectors squarely fall into 2), context based methods. Our key observation is that there exists an additional set of vectors, one for each in-vocabulary item, which are trained internally by the word2vec model. In word2vec, each term is represented by two latent vectors: a *word vector* and a *context vector*. These vectors are better suited, by design, for deducing OOV term vectors from the surrounding context words. When the model is being trained, these additional *context vectors* are used to represent the terms when they are found in the context window of the target term. These vectors can be easily extracted from a word2vec model.

Recently *contextualized word vectors* (CWV) methods such as ELMo and BERT have received much attention in the NLP community (Horn, 2017; Peters et al., 2018; Devlin et al., 2018). CWV prediction shares some similarity with OOV vector prediction, but they are intrinsically distinct problems. CWVs derive a vector representation for a term based on the surrounding context in each usage, thus there is not a generalized concept of a word vector for a lexical item as is the case with traditional word vectors. While in traditional word vector models (like word2vec), each term gets one vector, which is used regardless of the surrounding terms. We include them in our empirical study on the OOV benchmark tasks.

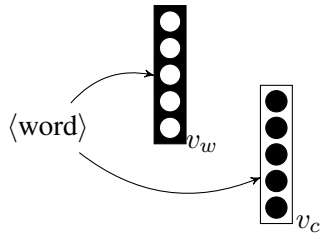

Figure 3: Each term is represented by two latent vectors, a *word vector* $\mathbf{w}_i$ and a *context vector* $\mathbf{c}_i$.

4 Methodology

Based on our observation discussed previously, we suggest that *the context vectors of the context words* can be used to predict the word embedding vectors for OOV terms. We introduce our method, COIN, which uses context vectors only to solve the prediction problem for OOV terms.

4.1 Problem Definition

For some word embedding model M, let $\mathcal{V}$ be the vocabulary of terms for which we have a trained word vector. Then there exists a mapping $\mathbf{w} : \mathcal{V} \rightarrow \mathbb{R}^n$ where n is the dimension of the embedded vectors. For a term $t \notin \mathcal{V}$, let $\hat{\mathbf{w}}(t) \in \mathbb{R}^n$ be the optimal word vector for t based on its semantic properties and the properties of the embedding space. Assuming $\hat{\mathbf{w}}(t)$ is known in our problem. We define the problem as follows:

Given a set of terms $\mathcal{U}$, where $\mathcal{V} \cap \mathcal{U} = \emptyset$, find a function f such that $\forall t \in \mathcal{U}$, $f(t) \approx \hat{\mathbf{w}}(t)$.

4.2 Proposed Approach

To find a COIN vector for an OOV term, we take the sum of the context vectors of the words in its contexts. Let $\mathbf{c} : \mathcal{V} \rightarrow \mathbb{R}^k$ be the mapping of terms to context vectors in M. For the n-th occurrence of term t_i, its context window C_i^n is the set of $2 \times k_w$ terms that are found up to k_w places before and after t_i, excluding t_i itself, for context window size k_w. Then for term t_i we define the COIN vector prediction $\hat{f}_{COIN}(t_i)$ as:

$$\hat{f}_{COIN}(t_i) = \sum_j \sum_{t' \in C_i^j} \mathbf{c}(t')$$

where each C_i^j is one observed context window.

There are some small modifications to the basic COIN pattern that are easily introduced. These include: stop word removal, word weighting, and principal component removal. Stop word removal

(denoted **nsw**) ignores stop words in the vector summation, and generally gives some improvement in accuracy. We apply the stop words provided in python NLTK in our experiments. Word weighting multiplies each vector by a function of its term, so that certain words have more importance in the final sum. Here we consider SIF (Arora et al., 2017) which weights each term by the inverse of its frequency. Principal component removal (Mu and Viswanath, 2018) removes the top principal component from each vector, which has been shown to improve the performance of word vectors on similarity tasks. On the CRW task we show that a combination of all three yields good results.

Additionally, we consider separately how the various setting for learning word embeddings via *word2vec* and the resulting *context vectors* are suited for estimating OOV word vectors. We consider the skip-gram and CBOW architectures, using both HS and NS respectively for updating. Section 5.6 discusses the difference among these models.

5 Evaluation

In this section, we compare our method COIN to a number of state-of-the-art models to show the efficiency and effectiveness of COIN. We also show that COIN vectors can be used in conjunction with an existing, trained method to yield improved results over either method alone

5.1 Models in Comparison

We compare COIN against recent OOV prediction models and CWV methods in the literature:

ADD (Lazaridou et al., 2017) produces an OOV prediction as the sum of the word vectors of all the neighboring terms. We also consider **ADD-nsw**, a same model with stop words removed.

N2V (Herbelot and Baroni, 2017) updates a set of OOV terms initialized by an additive method, by running an accelerated skip-gram training.

ALC (Khodak et al., 2018) learns a linear transformation on the set of ADD vectors that minimizes the ℓ_2 distance to the known word embedding vector for a set of training terms.

FastText (Bojanowski et al., 2017) is a morphological embedding algorithm that extends *word2vec*. It represents each word as an n-gram of characters instead of learning vectors for words directly. We use its implementation in gensim[1].

FCM is the Form Context Model of (Schick and Schütze, 2019b); it is a two part model that learns both a form and context component.

HiCE. HiCE (Hu et al., 2019) is an attention-based hierarchical context encoder that uses both the contexts and morphological features of an OOV word.

BERT. We take the embedding output from the last hidden layer of BERT (Devlin et al., 2018) as the vector representation for a word in our experiment.

ELMo (Peters et al., 2018) is a deep contextualized word representation; the representation for a word depends on the entire context in which it is used.

Model Parameter Settings. For all the methods (e.g., COIN, ADD, N2V, ALC, and FCM) that depend on a pre-trained *word2vec* model, we use the one provided by (Herbelot and Baroni, 2017). We run the experiments using the source codes and settings published in the corresponding referenced paper. For the method FastText, we set embedding size to 400 (same as the embedding size in the *word2vec* model), parameter window is set to 5, the minimum count is set to 1, and the model is trained for 5 epochs. We quote the HiCE experiment results from (Hu et al., 2019). BERT and ELMO are context dependent CWV methods that generate different word embeddings for the same word in different sentences. We use the pre-trained BERT base model from Hugging Face[2] which contains 12 hidden layers, and the pre-trained ELMo model from AllenNLP[3]. The BERT and ELMo embeddings for the OOV words are averaged by their context words' embeddings.

5.2 Benchmark Tasks

We evaluate the quality of our COIN embeddings on the common OOV benchmark tasks: Definitional Nonce (DefNonce), Contextual Rare Words (CRW), and Chimeras (Chim).

5.2.1 DefNonce Task

DefNonce (Herbelot and Baroni, 2017) is a set of sentences harvested from Wikipedia articles which are designed to be maximally informative. The first sentence of articles describing one word topics are extracted. The target term is the article title. Sentences that contain at least 10 words are randomly sampled creating 700 training and 300 test instances. The FastText model is trained on these

[1] https://radimrehurek.com/gensim/

[2] Facehttps://huggingface.co/bert-base-uncased

[3] http://docs.allennlp.org/v0.9.0/api/allennlp.modules.elmo.html

name	DefNonce	enwiki	WWC
dimension	400	400	300
vocab size	259,376	560,881	230,130
min count	50	50	50
source	Wikipedia	Wikipedia	WWC
model type	skip-gram	CBOW skip-gram	CBOW skip-gram
updating method	NS	NS HS	NS HS
window size	5	5	5

Table 1: Specifications of training word2vec embeddings in three datasets. The sample number in negative sampling (NS) is 5. The value of alpha is set to 0.025 in skip-gram, 0.05 in CBOW.

	method	MRR. ($\times 10^{-2}$)	Med. Rank	Time in sec avg (STD)
Part 1	FastText	0.95	2202	30 (0.39)
	FCM-ctxt*	6.56	184	/
	FCM-form*	12.98	404	/
	FCM*	**17.54**	**49**	19,236 (682)
Part 2	N2V*	4.91	623	1,166 (12.99)
	ALC*	7.06	165	842 (12.83)
	ADD	0.95	3881	414 (1.72)
	ADD-nsw	3.62	876	429 (9.83)
	COIN	9.43	100	415 (1.38)
	COIN-nsw	**9.46**	**90**	423 (4.85)
Part 3	BERT	2.34	242	518 (9.74)
	ELMo	4.25	62	1,090 (12.81)

*Performance of these methods are quoted from the referenced paper. Our re-runs may differ slightly.

Table 2: Results on the DefNonce task. Methods are divided in into three groups. Methods in Part 1 make use of context and other information, the ones in Part 2 only use context information, Part 3 is CWV methods.

training instances. The goal for this evaluation task is to learn a word vector for the target term that is close to the known word vector for that term. (Herbelot and Baroni, 2017) provide a set of pre-trained word2vec vectors trained on a $1.6B$ word Wikipedia snapshot using the skip-gram architecture, with negative sampling. The word2vec model parameters are shown in Table 1 (See DefNonce).

Experimental Results on DefNonce. Table 2 gives the results of this evaluation. Accuracy on this task is measured by two values: Mean Reciprocal Rank (MRR), where higher is better, and Median Rank, where lower is better. We show the form and context components of FCM, denoted FCM-form and FCM-ctxt respectively, along with the full FCM model. Of all the models FCM performs the best on this task. However if we limit our focus to methods which only use context information (see Part 2), COIN is more accurate than either ALC or FCM-ctxt, and COIN-nsw is the leader in both measures. With COIN-nsw, for all the terms in the test set, half are among the 90 closest terms, out of a vocabulary size of 259,376.

Comparing FCM with FCM-ctxt and FCM-form, it seems that much of the improvement for FCM comes from the form information, which is suggested by the relative performance of FCM-ctxt and FCM-form. We also observe that while the MRR for FCM-form is high, so is the Median Rank, suggesting that this model performs very well on some terms, presumably morphologically rich examples, but poorly on a good number of other terms.

Between the two CWV methods, ELMo performs better. It gives a competitive Median Rank, the second lowest, better than COIN's, but not as good as FCM's. Its MRR is not as competitive. Our explanation is that ELMo performs well when sentences have informative context for OOV words. But its predicted vector representation of an OOV term is far from the "true" representation when sentences contain less informative context.

Comparison of Runtime. We repeat the experiments of each method 10 times and report their averaged runtimes together with the standard deviations (STD) in the last column of Table 2. We ran the experiments on a PC with an Intel i7-8700K CPU @ 3.70GHz, 64GB RAM @ 2133 MHz, and a NVDIA GeForce GTX 1080 GPU. We find that COIN and its variant COIN-nsw have a good balance between accuracy and runtime. For example, FCM achieves the best performance with 0.1754 MRR and 49 Median Rank, but it requires over 5 hours, which is orders of magnitude larger than that of COIN (415 seconds) and COIN-nsw (423 seconds). FastText runs the fastest (30 seconds) but with a dramatic sacrifice in accuracy (0.0095 MRR and 2202 Median Rank). Put together, the running time and accuracy reported in Table 2, support our claim that COIN is a strong, inexpensive baseline for predicting OOV word embeddings.

5.2.2 CRW Task

CRW (Khodak et al., 2018) is a subset of 562 pairs of words from the Rare Word dataset (Luong et al., 2013) combined with 255 sentences for each rare word sampled from the Westbury Wikipedia Corpus (WWC) (Shaoul and Westbury, 2010). Each pair of words has been manually annotated with

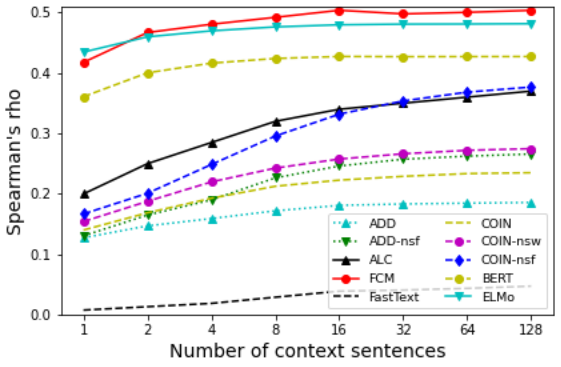

Figure 4: Spearman correlation results for CRW task.

a similarity score. The 255 context sentences are partitioned into eight disjoint subsets of sizes 1, 2, 4, . . . , 64, 128. The goal is to match the human similarity scores between pairs of words in the CRW, measured using Spearman correlation. The authors provide a set of word2vec word embeddings trained on a subset of the WWC from which all sentences containing a rare word have been removed. However they only include the trained word embeddings, without the context vectors needed for our technique. Therefore we train our own set of embeddings using their original training data, trying to follow their parameters as closely as possible (See column WWC in Table 1). We also train FastText embeddings on this training data.

Experimental Results on CRW. Figure 4 gives the Spearman correlation values across the sample sizes, for various model designs. The basic COIN vectors perform better than ADD on this task. Following (Khodak et al., 2018) we explore SIF-weighted vectors, with stop words removed and top principal component removal. The resulting COIN vectors, COIN-nsf, perform about as well as ALC on this task across context sizes. FCM performs the best for all context sizes. (Schick and Schütze, 2019b) points out that the Rare Word dataset was designed to contain many morphologically analyzable words. Form based methods, such as FCM, likely benefit from this design.

5.2.3 Chim Task

Chim (Lazaridou et al., 2017) constructs novel concepts by combining the contexts of pairs of existing terms to derive a "chimera" of their attributes, for example "alligator" and "rattlesnake." Human annotators rank the similarity of the chimeric concept to a set of six probe words (see the example in Table 3). Context sentences are provided for each concept in sizes 2, 4, and 6 (half for each of the existing terms). The goal of this trial is to match the similarities of the probes to the chimeras as given by the human judges, measured by Spearman correlation. The word2vec model used on this task is the same as the one on task DefNonce. The FastText model is trained on WikiText-103 (Merity et al., 2016), following the way in (Hu et al., 2019).

Experimental Results on Chim. Table 4 gives the results. CWV methods cannot be used in this task as there is no sentence for the probe words to learn their embeddings. A big challenge of this task is that the randomly selected sentences are not necessarily informative about the concepts. For the relatively larger sample size of six sentences a selective model such as FCM-AM-ctx (Schick and Schütze, 2019a) is able to focus on the contexts that will be most helpful. Among the context-based methods, ADD-nsw performs the best. But the performance of our method, COIN-nsw, is very close to that of ADD-nsw.

We observe that COIN is much better in the DefNonce task but not in Chim task. A possible explanation is that the sentences for the OOV words in Chim task are randomly selected and thus, are not necessarily informative about the nature of the OOV words. In COIN, we get the vectors of OOV words based on the context words in the sentences and use these vectors to calculate the similarity between words. In contrast, the gold standard similarity between words is from human rankings. We hypothesize that the intrinsic less information of COIN vectors (from these sentences) may be the reason for COIN not performing that well in Chim.

5.3 Downstream Tasks

Following (Schick and Schütze, 2019b), we conduct an experimental study using the SentEval evaluation toolkit (Conneau and Kiela, 2018) which contains various types of sentence classification tasks, including sentiment analysis (MR, SST2, and SST5) (Socher, 2013; Hosseinia et al., 2019; Schneider and Dragut, 2015), product reviews (CR) (Hu and Liu, 2004; Hosseinia et al., 2020), subjectivity/objectivity (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Hosseinia et al., 2021; Tumarada et al., 2021; Wiebe et al., 2005; Yang et al., 2020), and paraphrase detection (MRPC) (Dolan et al., 2004; Aljebreen et al., 2021). We replace the OOV terms with their COIN and FCM predictions, and our results mirror those reported in (Schick and Schütze, 2019b), showing no signif-

input terms:		*probes:*					
alligator	rattlesnake	crocodile	iguana	gorilla	banner	buzzard	shovel
		2.29	3.29	3.43	2.0	3.71	2.14
sentences:							
1. animals such as capybara jaguars jacare ___ and hyacinth macaws are particularly vulnerable							
2. nadirpur stared at it as though it were a ___ his face quite drained							

Table 3: Chimera example for input pairs and probes. Similarities have been determined by human judges.

	method	2 sent.	4 sent.	6 sent.
Part 1	FastText	0.178	0.174	0.129
	FCM-ctx	0.337	0.359	0.422
	FCM-AM-ctx	0.342	0.376	**0.436**
	HiCE	**0.378**	**0.405**	0.431
Part 2	N2V	0.332	0.367	0.389
	ALC	0.363	0.384	0.394
	ADD	0.303	0.340	0.337
	ADD-nsw	**0.354**	**0.379**	**0.416**
	COIN	0.299	0.290	0.359
	COIN-nsw	0.336	0.323	0.395

Table 4: Results on the Chim task reported as Spearman correlation. All methods are grouped into two parts: Part 1 and 2, as in Table 2. Neither ELMo nor BERT can be used in this task as there is no sentence for the probe words to learn their embeddings.

icant change among the OOV prediction methods on the results (COIN 73.9% to FCM 74.1%).

5.4 Efficiency and Effectiveness of COIN

According to the comparison between COIN and other models, our method COIN is very competitive, typically outperforming all but FCM in many of the tasks. However a significant advantage to our technique is that it requires no training beyond learning the initial word embedding model, while FCM requires a significant amount of training time and computing resources. In our experiments, training the FCM model took 5.34 hours on average. We stress that COIN requires no such additional training time. The only computational cost of COIN is the equivalent of a few look-ups of word–vector mappings (in seconds). Specifically on the 300 test words of the DefNonce task, the COIN method takes 131 seconds from loading the word2vec model, generating the COIN vectors to evaluation. The COIN method takes 21 seconds to run all of the 2, 4, and 6 sentence evaluations on the Chim task. *This makes our technique substantially more practical to use, particularly in development stages, giving the user a strong starting point.* In a real world situation when we encounter an OOV term without having previously trained a model for one of the trained methods on a corpus of similar documents, COIN can quickly give a predicted word vector based on the context at hand, with accuracy close to the cutting edge techniques. *In a scenario where it is unknown if a more complicated model may make a difference on the downstream result, a user can use COIN to provide a first look while training another OOV prediction model, whereby the COIN model provides a good basis to extend and build upon.*

method	MRR.	Median Rank
FCM	0.1724 (0.1726)	53.8 (50)
COIN+FCM	**0.1731** (0.1735)	**52.7 (46)**
p-value	0.0003	0.0008

Table 5: Results on the DefNonce task. FCM initialized with COIN vectors improves both MRR. and Median Rank. We give the average and best (in parenthesis) outcomes for both MMR. and Median Rank.

5.5 Initialization with COIN Vectors

The process of deriving COIN vectors is very quick and efficient, therefore they provide a good model for initializing other prediction methods. Typically these methods begin with an additive model for OOV terms, then further refine the embeddings through some training strategy (Herbelot and Baroni, 2017; Khodak et al., 2018; Schick and Schütze, 2019b). COIN vectors can be used in place of the additive vectors as the first level approximation. In this study our goal is to show that COIN vectors can be used in conjunction with an existing, trained method to yield improved results over either method alone. We focus here on the FCM method and investigate a modified design which uses COIN vectors to initialize the model.

On the DefNonce task, COIN achieves better results than the FCM-context piece alone as shown in Table 2. We train the models as described in (Schick and Schütze, 2019b) using the vectors provided by (Herbelot and Baroni, 2017). We train the standard FCM model and our combined model,

dataset	type	update	MRR.	Median Rank
enwiki	CBOW	NS	0.0614	157
	CBOW	HS	**0.1148**	102
	S-G	NS	0.0802	173
	S-G	HS	0.0711	270
WWC	CBOW	NS	0.1037	**72**
	CBOW	HS	**0.1075**	109
	S-G	NS	0.0727	104
	S-G	HS	0.0759	133

Table 6: Results of different choices from model types (CBOW or S-G) and updating methods (HS or NS) for COIN on task DefNonce. S-G is short for skip-gram.

COIN+FCM, on the same training data used in their paper, for 10 epochs. Selecting the best performing epoch for each algorithm on the training set, we then run this model on the test data. We repeat the experiments of FCM and COIN+FCM 10 times, and report their average and best performance in Table 5. We notice that using COIN vectors to initialize the model improves the performance in both MRR and Median Rank. The improvement is slight but consistent in every randomized run. Thus, we calculate the p-value to verify that the performance differences are significant. With a p-value = 0.0003 for the MRR. and p-value = 0.0008 for the Meidan Rank, the results are significant at the 0.05 level.

We note that while the training data is the same as used in (Schick and Schütze, 2019b), the FCM performances here differ slightly with that in Table 2, this is because the results here are averaged over 10 repeats, and the outcome of each run varies due to inherent randomness in the model. We also run the CRW task comparing the FCM model and COIN+FCM model. The results show that COIN+FCM achieves a slight improvement at most context sizes. The largest difference comes when the number of contexts is very small. This makes sense under the assumption that the COIN vectors are closer to the desired target than ADD vectors, and therefore fewer training examples are required to fit the model.

In summary, FCM can be initialized by COIN without additional cost and this combination yields more accurate results on the benchmark tasks.

5.6 Embedding Model Selection

There are various hyperparameters and model choices in training word embeddings which may play a role in how the context vectors relate to the word vectors in OOV prediction. In this section we consider different settings for word2vec models to compare their performance. Specifically we explore the model types (CBOW or skip-gram) and updating methods (HS or NS). We train these vectors on two corpora: a snapshot of Wikipedia from January 2019 and the subset of the WWC used by (Khodak et al., 2018) in their CRW study. The model parameters are described in Table 1 (See enwiki and WWC). We repeat the DefNonce evaluation for comparison between model selection. The results here are not directly comparable to those given in the previous sections.

Results are shown in Table 6. One observation is that CBOW generally yields better vectors for COIN than skip-gram. CBOW with HS on enwiki gives the highest MRR of all COIN models. For this model in particular, the much larger vocabulary of enwiki does not present a liability on this task. The Median Rank is also competitive among all the vector predictions. The lowest Median Rank is achieved on the WWC data with CBOW and NS.

6 Conclusion

In this work we present the COIN method for predicting word embedding vectors for OOV terms using the context vectors of context words. We advertise COIN as an inexpensive and strong baseline. We show that COIN performs close to the existing state-of-the-art techniques, while being much faster as it requires no additional training. We also show how COIN can be used along with existing techniques to give a new state-of-the-art on vector prediction tasks, and how it can be used to help downstream tasks such as sentiment analysis. Besides, we explore different model settings for learning the word vectors. The results of COIN are robust across model choices, with generally better performance from CBOW trained embeddings.

Acknowledgements: This work was supported in part by the U.S. National Science Foundation BIGDATA 1838145 and 1838147 grants, and a gift from NVIDIA Corporation.

References

Abdullah Aljebreen, Weiyi Meng, and Eduard Dragut. 2021. Segmentation of tweets with urls and its applications to sentiment analysis. *AAAI*, 35(14):12480–12488.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR 2017*.

Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzębski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2018. Learning to compute word embeddings on the fly.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR*, 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Hao Chen, Susan McKeever, and Sarah Jane Delany. 2019. The use of deep learning distributed representations in the identification of abusive text. *ICWSM*, 13(01):125–133.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *LREC*.

Sunipa Dev, Safia Hassan, and Jeff M Phillips. 2021. Closed form word embedding alignment. *Knowledge and Information Systems*, 63(3):565–588.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

William Dolan, Chris Quirk, Chris Brockett, and Bill Dolan. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources.

Nicolas Garneau, Jean-Samuel Leboeuf, and Luc Lamontagne. 2019. Predicting and interpreting embeddings for out of vocabulary words in downstream tasks. *arXiv preprint arXiv:1903.00724*.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *JMLR*, 21(248):1–43.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *EMNLP*, pages 304–309.

Franziska Horn. 2017. Context encoders as a simple but powerful extension of word2vec. In *Repl4NLP*, pages 10–14.

Marjan Hosseinia, Eduard Dragut, Dainis Boumber, and Arjun Mukherjee. 2021. On the usefulness of personality traits in opinion-oriented tasks. In *RANLP*, pages 547–556.

Marjan Hosseinia, Eduard Dragut, and Arjun Mukherjee. 2019. Pro/con: Neural detection of stance in argumentative opinions. In *SBP-BRiMS*, pages 21–30, Cham.

Marjan Hosseinia, Eduard Dragut, and Arjun Mukherjee. 2020. Stance prediction for contemporary issues: Data and experiments. In *SocialNLP*, Online.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *ACM SIGKDD*, pages 168–177.

Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-shot representation learning for out-of-vocabulary words. *arXiv preprint arXiv:1907.00505*.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *ACL 2018*, pages 12–22.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive Science*, 41(S4):677–705.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositionally derived representations of morphologically complex words in distributional semantics. In *ACL 2013*, pages 1517–1526.

Quanzhi Li, Sameena Shah, Xiaomo Liu, and Armineh Nourbakhsh. 2017. Data sets: Word embeddings learned from tweets and general data. In *ICWSM*, pages 428–436.

Ismini Lourentzou, Kabir Manghnani, and ChengXiang Zhai. 2019. Adapting sequence to sequence models for text normalization in social media. In *ICWSM*, pages 335–345. AAAI Press.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR 2013*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*, pages 3111–3119.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *ICLR*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*.

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

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

Timo Schick and Hinrich Schütze. 2019b. Learning semantic representations for novel words: Leveraging both form and context. In *AAAI 2019*.

Andrew Schneider and Eduard Dragut. 2015. Towards debugging sentiment lexicons. In *ACL*, pages 1024–1034.

Joan Serrà, Ilias Leontiadis, Dimitris Spathis, Gianluca Stringhini, Jeremy Blackburn, and Athena Vakali. 2017. Class-based prediction errors to detect hate speech with out-of-vocabulary words. In *WOAH6*, pages 36–40.

Cyrus Shaoul and Chris Westbury. 2010. The Westbury Lab Wikipedia Corpus. `http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html`.

Richard et al. Socher. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Kishore Tumarada, Yifan Zhang, Fan Yang, Eduard Dragut, Omprakash Gnawali, and Arjun Mukherjee. 2021. Opinion prediction with user fingerprinting. In *RANLP*, pages 1423–1431.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *LREC*, 39(2):165–210.

Fan Yang, Eduard Dragut, and Arjun Mukherjee. 2020. Predicting personal opinion on future events with fingerprints. In *COLING*, pages 1802–1807.