Estimator Vectors: OOV Word Embeddings based on Subword and Context Clue Estimates

Raj Patel
Department of Computer Science
George Mason University
Fairfax, VA, USA
rpatel17@masonlive.gmu.edu

Carlotta Domeniconi
Department of Computer Science
George Mason University
Fairfax, VA, USA
cdomenic@gmu.edu

***Abstract*—Semantic representations of words have been successfully extracted from unlabeled corpuses using neural network models like word2vec. These representations are generally high quality and are computationally inexpensive to train, making them popular. However, these approaches generally fail to approximate out of vocabulary (OOV) words, a task humans can do quite easily, using word roots and context clues. This paper proposes a neural network model that learns high quality word representations, subword representations, and context clue representations jointly. Learning all three types of representations together enhances the learning of each, leading to enriched word vectors, along with strong estimates for OOV words, via the combination of the corresponding context clue and subword embeddings. Our model, called Estimator Vectors (EV), learns strong word embeddings and is competitive with state of the art methods for OOV estimation.**

***Index Terms*—Deep learning, Natural language processing, Knowledge representation**

I. Introduction

Semantic representations of words are useful for many natural language processing (NLP) tasks. While there exists many ways to learn them, models like word2vec [13] and GloVe [17] have been shown to be very efficient at producing high quality word embeddings. These embeddings not only capture similarity between words, but also capture some algebraic relationships between words. These models, though, also have some downsides. One major drawback is that they can only learn embeddings for words in the vocabulary, determined by the corpus they were trained on. Although common words are typically captured, most existing approaches are unable to learn the meaning of new words, known as out of vocabulary (OOV) words, a task humans can do easily. Unknown words could be new words or domain specific words, both of which could be very important for NLP tasks. Therefore, finding good representations for these words poses a relevant challenge. Some attempts have been made to estimate representations of OOV words, generally based on how humans learn a new word. One way is to use external auxiliary information, like definitions of the word [2]. One downside of this is that it requires external information, which may not be accessible. Another way that gets around this problem is to estimate OOV word representations using word roots or subwords [3], [19]. This approach can work well, but struggles on words that have less meaningful word roots. Another strategy is to use the context the OOV word appears in (in human learning, these are known as *context clues*). These methods estimate OOV representations by adding the context word representations [8], [11] or by training the representation with these words [7]. Context words are generally good for estimating an unknown word, but these methods can struggle with weighing the important context clues over the less important ones.

In this paper, we propose Estimator Vectors (EV), a new neural network approach that learns three types of embeddings: word, context clue, and subword embeddings. The word embeddings are similar to other word embedding methods, while the context clue and subword ones are used to estimate word embeddings when they are encountered. This approach learns the embeddings jointly, enhancing the quality of each.

The major contribution of this work is a novel and effective approach (Estimator Vectors, or EV) to word embedding and out-of-vocabulary estimation with the following distinctive features: (1) EV learns and uses three sets of vectors, context clue embeddings, subword embeddings, *and* word embeddings, each for its own specific purpose; (2) EV learns the embeddings at the same time, in order to learn the three sets of vectors effectively. EV learns context clue embeddings and subword embeddings such that their individual averages estimate the representation of the target word. At the same time, it uses the target estimate in the target context pair to learn word embeddings. The interplay of the three embeddings enhances the learning of all three, leading to strong estimates.

The rest of the paper is organized as follows. Section II discusses the background and related work. Section III defines EV in detail. Section IV describes the experimental setup, and Section V discusses the results. Finally, Section VI concludes the paper.

II. Related Work

In this section we discuss relevant previous work. First, we discuss word embeddings in general, then focus on strategies for estimating OOV words.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

A. Word Embeddings

Word2vec [13] is a popular approach for computing semantic representations of words. It relies on training a shallow neural network model based on predicting the context of words, using backpropagation. The original model had two versions; continuous bag of words, which uses the context of a word to predict the word itself, and skipgram, which uses the word to predict its context [13]. Both of these models input a 1-hot encoding of each word, and output a softmax probability distribution over the vocabulary. The word vectors are the first layer weights connected to a word's index. Due to the large amount of calculations needed to compute the softmax distribution over the entire vocabulary (usually a very large number), Mikolov et al. [14] proposed a more efficient method, known as negative sampling. Instead of calculating the probability of every word in the vocabulary, negative sampling tries to maximize the probability of a target word co-occurring with its context words, while also minimizing the probability of the target word co-occurring with randomly selected words, known as negative samples. In the negative sampling version of the skipgram model, the probability of a target word w_t and a context word w_c occurring together is calculated as:

$$\sigma(u_{w_t} \cdot v_{w_c}) \tag{1}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the logistic function, u_{w_t} is the word vector representation for the target word w_t, and v_{w_c} is the word vector representation for the context word w_c. In this formulation, target context pairs with similar representations will have high dot products, leading to a high value (closer to 1), while different representations will have a low dot product, leading to low values (closer to 0). The overall loss function for negative sampling with the skipgram model is the negative log likelihood:

$$E = -log\sigma(u_{w_t} \cdot v_{w_c}) - \sum_{n \in N} log\sigma(-u_{w_t} \cdot v_n) \tag{2}$$

where N is the set of negative samples.

To demonstrate how the skipgram model is trained, we use the example sentence "The yellow car sped up quickly". If the target word w_t was "car", one context word w_c could be "sped". Skipgram learns a representation such that $\sigma(u_{car} \cdot v_{sped})$ is a large value, while the probability of co-occurrence with negative samples, like $\sigma(u_{car} \cdot v_{coffee})$, is small.

Continuous bag of words also has a negative sampling version [14]. Continuous bag of words pairs the sum of the context vectors with the target vector, and learns a probability that is high when the sum of the context is paired with its target and low when paired with negative samples. The probability is calculated as:

$$\sigma(\sum_{c \in C} v_{w_c} \cdot u_{w_t}) \tag{3}$$

where C is the set of all the context words for the target.

Continuous bag of words trains all the context words at once. For the example "The yellow car sped up quickly", continuous bag of words learns representations such that $\sigma((v_{the} + v_{yellow} + v_{sped} + v_{up}) \cdot u_{car})$ is a large value, while with a negative sample like "coffee", $\sigma((v_{the} + v_{yellow} + v_{sped} + v_{up}) \cdot u_{coffee})$ is a small value.

Both the skipgram model and the continuous bag of words model learn embeddings on a subsampled version of the training corpus, in order to reduce the influence of overly frequent words. Mikolov et al. [14] found that removing some instances of very frequent words improves the general quality of the word embeddings. Therefore, word instances are removed with a probability based on their frequency, where more frequent words have a higher probability of being removed.

B. Out-Of-Vocabulary Embeddings

Models like word2vec lead to effective word representations, but only for words in the vocabulary of the original training corpus. Therefore, models using word2vec representations struggle when encountering OOV words. There have been attempts to estimate OOV words' vector representations. These approaches tend to mirror human strategies for learning new words; by using the word's roots/subwords or using the context the word was found in.

1) Subword Based Approaches: One way to estimate an OOV word's embedding is to use its subword information. Bojanowski et al. [3] train both word vectors and subword vectors at the same time, such that the sum of the subwords' vectors approximates the target word. Their model, known as fastText, uses a similar approach to the negative sampling skipgram model, but replaces the target representation u_{w_t} with the sum of its n-gram vectors, so the probability (of w_t and w_c co-occuring) becomes:

$$\sigma(\sum_{g \in G_{w_t}} z_g \cdot v_{w_c}) \tag{4}$$

where G_{w_t} is the set of character n-grams (the subwords) of the target word w_t, and z is the embedding of the subwords.

The subword model is similar to skipgram, except it estimates the target word's embedding with the sum of its subword vectors. In the recurring example, if $n = 2$ for the character n-grams, the probability $\sigma(u_{car} \cdot v_{sped})$ of skipgram is replaced with $\sigma((z_{<c} + z_{ca} + z_{ar} + z_{r>} + z_{<car>}) \cdot v_{sped})$, where the "<" and ">" denote the beginning and end of a word respectively, and "<car >" is a special token added to the subword set.

Subword methods are powerful, but do have some flaws. They struggle with words that have weak or unknown word roots. For example, subword methods may struggle with foreign words, as they may not learn the subwords required for a good estimate.

2) Context Based Approaches: Other methods use the context of an OOV word to estimate its representation. Some methods simply sum the existing context word embeddings to get an estimate of the OOV word [8], [11]. This summation method works due to the algebraic property of word2vec, which states that simple algebraic operations can be applied to word vectors in order to mimic the operation semantically (for example, u("King") - u("Man") + u("Woman") $\approx$

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

u("Queen")) [13]. This suggests that adding relevant words like the context words should give a good estimate of the OOV word representation, which is demonstrated in [8], [11]. Other OOV estimation methods refine the estimate using various techniques. Nonce2vec [7] trains the estimate in the existing skipgram model using a very high learning rate. Another technique, *à la carte*, [10] refines the estimate with a linear transformation learned from the original training corpus and set of word vectors.

The above models do have limitations when it comes to estimating OOV embeddings. They tend to be based on the summation of their word embeddings, which may have weaknesses, depending on the original embedding method. The word2vec models only focus on learning word embeddings, which may hinder their ability to estimate OOV words. The skipgram model learns with one target context pair at a time, and therefore is not capable of learning how multiple context words relate to each other in terms of impact, nor what each context word says about the target word relative to the others. This means when the context words are summed to estimate an OOV word embedding, the estimate will be less accurate, as each words' relative importance (how much it affects the summed vector) and relative contribution (what it adds when being summed with other context words) are not captured by a skipgram model. The continuous bag of words model, on the other hand, does learn how word vectors relate to each other by summing the context words, as it is trained on this sum. However, it does not learn infrequent words' embeddings well [16]. This is because it learns all of its vectors at once (the sum of the context vectors, being paired with the target), diluting infrequent words (as the model does not learn infrequent words individually, decreasing how much it learns per target word). This may interfere with how well the sum of context words can estimate an OOV word, as we expect less frequent words to be informative for an OOV word.

3) Combined Approaches: The Form-Context model [22] combines subwords and context words to estimate OOV words. This approach takes previously trained word embeddings and a text corpus, and trains a model that can estimate an OOV word embedding using subword information and context information. For the subword representation (which they call the form representation), it learns a subword embedding for each character ngram, similar to fastText [3]. For the context representation, it takes the average of the context words, and learns a linear transformation to estimate the OOV word, similar to *à la carte* [10]. It then combines the embeddings in a weighted sum:

$$v = \alpha \cdot v_{context} + (1 - \alpha) \cdot v_{form} \tag{5}$$

where α decides how much to weigh the context estimate against the subword estimate. Schick and Schütze [22] propose two different ways to calculate α. The first approach, known as single parameter, simply learns α as one value for every subword context pair. The second approach, known as the gated model, learns parameters to calculate α with the following function:

$$\alpha = \sigma(w^T[v_{context}, v_{form}] + b) \tag{6}$$

where σ is the sigmoid function, and $w \in \mathbb{R}^{2k}$ and $b \in \mathbb{R}$ are learnable parameters (k is the embedding dimensionality). The gated model learns to weigh how important the context is compared to the subwords in each scenario.

Other combined approaches use attention mechanisms in order to enhance the context estimates of OOV words. Attention is used to help the model decide which contexts (when there are multiple) to weigh more when approximating an unknown word. One such model is Attentive Mimicking [21]. Attentive Mimicking is an extension of the Form-Context model mentioned above. While the Form-Context model weighs each context equally, Attentive Mimicking weighs contexts based on how much they agree with other contexts of the OOV word. The more similar each context is to the others, the higher the weight towards that context is (as it is more likely to be relevant).

Another attention based approach is HiCE (Hierarchical Context Encoder) [9]. HiCE is a deep model that uses self-attention blocks [25] to encode each context of a OOV word, and then uses another set of self-attention blocks to combine each encoded context into a final context estimate. It then creates a subword estimate using a character based convolutional neural network. These estimates are then concatenated and then inputted to an output layer that outputs the final estimate. Like the Form-Context model, it trains to predict already existing embeddings, using a large text corpus. In addition, HiCE improves its embedding estimates by adapting the trained model to newer data sets using a technique known as Model Agnostic Meta-Learning (MAML) [6].

The combined models are very good at estimating OOV words, but do have some drawbacks. First, they depend on the quality of word embeddings they are trained on, as they do not learn their own. Because both the subword and context sections of the models are trained to estimate the word embeddings directly, the word embeddings do not learn from subword or combined context information, which could enrich the quality of the embeddings.

III. Estimator Vectors

A. Model

We present Estimator Vectors (EV), a word2vec based model that learns three types of representations: word embeddings, context clue embeddings, and subword embeddings. EV can easily create an embedding for an OOV word as it is encountered. Like the Form-Context model, Estimator Vectors combine both context and subword information. However, EV has some key differences. First, it learns word embeddings, subword embeddings, and context embeddings at the same time. This joint training leads to stronger representations. Second, unlike other methods' context estimations, EV learns a unique set of context vectors, which we call context clue embeddings. Unlike word embeddings, these embeddings are

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

trained in a sum, and therefore not only learn the meaning of each context word, but also how "informative" it is. This means that more informative words will have a greater impact on the sum, leading to a strong context estimate.

Our model takes a similar approach as the skipgram [13] and fastText [3] models mentioned above. EV trains on word co-occurrence pairs, but replaces the first word embedding (u_{w_t} in (1)) with a context clue estimate, which is the average of the context clue embeddings for the context words of w_t:

$$cc_{w_t} = \frac{1}{|Q_{w_t}|} \sum_{q \in Q_{w_t}} h_q \tag{7}$$

where Q_{w_t} is the set of context clues for w_t and h is the set of context clue embeddings.

In addition, it replaces the first word with the subword estimate as well, which is the average of the character n-gram embeddings for w_t:

$$sub_{w_t} = \frac{1}{|G_{w_t}|} \sum_{g \in G_{w_t}} z_g \tag{8}$$

where G_{w_t} is the set of character n-grams (the subwords) of the target word w_t, and z is the embedding of the subwords.

EV maximizes both probabilities for words that co-occur and minimizes both probabilities for the negative samples. The equations for the context clue probability is

$$\sigma(cc_{w_t} \cdot v_{w_c}) \tag{9}$$

where v is the set of word embeddings. Similarly, the probability for the subwords is

$$\sigma(sub_{w_t} \cdot v_{w_c}) \tag{10}$$

EV optimizes both probabilities at the same time, through the following error function:

$$\begin{aligned} E = &-log\sigma(cc_{w_t} \cdot v_{w_c}) - log\sigma(sub_{w_t} \cdot v_{w_c}) \\ &- \sum_{n \in N} [log\sigma(-cc_{w_t} \cdot v_n) + log\sigma(-sub_{w_t} \cdot v_n)] \end{aligned} \tag{11}$$

where N is the set of negative samples.

Note that two major components are being learned by this model; (1) the overall semantic space is being learned by the word vectors via the skipgram pairings and negative samples, and (2) the ability to estimate any word in the space is also being learned by the context clue and subword vectors. In addition, since these are all being learned at the same time, they are enhanced by each other. This means the word embeddings v learn from both context clue embeddings h and subword embeddings z, while h and z both learn from v. In addition, since both impact v, h indirectly learns from z and vice versa. This interplay between each type of vectors leads to high quality vectors of each type.

As an example, we return to the sentence "The yellow car sped up quickly". Given the target context pair "car" and "sped", Estimator Vectors would train on the following probabilities: $\sigma(\frac{1}{4}(h_{the} + h_{yellow} + h_{sped} + h_{up}) \cdot v_{sped})$ and $\sigma(\frac{1}{4}(z_{<c} + z_{ca} + z_{ar} + z_{r>}) \cdot v_{sped})$. In this example, the EV model learns three things: a semantic representation for "sped", how to estimate a semantic representation for "car" by learning representations for its context clues, and how to estimate "car" by learning representations for its subwords. In addition, it learns all of these representations based on the fact that "car" and "sped" co-occur.

The learned word vectors v can be used as normal word embeddings for downstream tasks. When an OOV word w_o is encountered, the context clue representation is calculated as:

$$cc(w_o) = \frac{1}{|Q_{w_o}|} \sum_{q \in Q_{w_o}} h_q \tag{12}$$

and the subword representation is calculated as

$$sub(w_o) = \frac{1}{|G_{w_o}|} \sum_{g \in G_{w_o}} z_g. \tag{13}$$

These estimates can then be combined for a final estimate of the OOV word:

$$est(w_o) = cc(w_o) + sub(w_o). \tag{14}$$

B. Postprocessing Context Clues

One advantage of word embeddings is that they can be summed to estimate a new word. However, sets of word vectors tend to share a few common directions, and summing multiple vectors can amplify these directions. This can harm the sum's representation, as the uncommon directions tend to carry more meaning. In order to reduce this problem, Mu and Viswanath [15] and Arora et al. [1] propose removing the top PCA components from the vectors.

In order to improve the context clue representations, we remove the top three components based on the word representations from the sum of the context clues. This is done before a context clue representation is combined with a subword representation. We denote the postprocessed context clue representation as $cc'(w_o)$, which leads to the final equation:

$$est(w_o) = cc'(w_o) + sub(w_o). \tag{15}$$

IV. Experiments

A. Baseline and Hyperparameters

We compare EV's word embeddings to the word2vec skipgram model and fastText. Both word2vec and fastText are trained using the gensim library, a very efficient embedding toolkit in Python [20]. EV's implementation[1] is also based on the gensim library. All models, including EV, use the same hyperparameters: embeddings of size 300, minimum frequency of 100, sampling with a rejection threshold (for reducing overly frequent words) of .0001, window sizes between 1 and 5 (uniformly sampled), and 5 negative samples. The models are trained with a learning rate of 0.025, which linearly decays to a minimum of .0001, as training goes on. Word2vec and fastText were trained for 15 epochs, and EV was trained for 20 (these were chosen from 5, 10, 15 and 20, based on each models' performance on the validation set for the definitional

[1] https://github.com/rajicon/Estimator_Vectors

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

nonce task (see Section IV-C). Since fastText isn't compatible with the definitional nonce set, it was set to 15 epochs to match word2vec. For fastText, character ngrams from size 3 to 5 were selected. For EV, the context clues of a target are taken from a window size of 3 (3 before and 3 after). If a word with less than 100 frequency occurs as a context clue, it is ignored. Additionally, the subwords used in EV were chosen as any ngram from size 3 to 5, and only those that occur in at least 3 words in the vocabulary (in order to compare to the Form-Context model, mentioned below).

In addition to word embedding quality, we also show EV's ability to estimate OOV words effectively. To this end, we compare to context based methods, subword methods, and combined methods. For context models, EV is compared to simple summation methods of the skipgram vectors and *à la carte* embeddings [10] (which are based on the trained word2vec embeddings). We train *à la carte* embeddings on the skipgram vectors mentioned above, with a minimum word count of 500 for training the linear transformation. For subword methods, we compare to fastText [3]. Finally, we compare to the state of the art combined methods, the Form-Context model and its extension, the Attentive Mimicking model [21], [22]. We train both of these models on the skipgram embeddings mentioned above, using the hyper parameters mentioned in [22]. These include a minimum word count of 100, and character ngrams from size 3 to 5 (ngrams only taken if they occur in at least 3 different words[2]) . For weights, we examine the gated model, as it generally had stronger results than the single parameter model. We slightly alter the tokenization method, and we do not shuffle the corpus when training on the form context model, both in order to match more closely to EV. Each model was trained for 20 epochs, with a version saved each epoch. The best models (on the definitional nonce validation set) were selected.

B. Data Set

All models were trained on the Westbury Wikipedia Corpus (WWC) [23]. We use a modified version provided by Khodak et al. [10] where sentences with certain rarewords removed for purpose of testing OOV estimation.

C. Testing

The EV model trains word embeddings along with context clue embeddings and subword embeddings, leading to two goals. The first goal is for the word embeddings (v) trained by this model to be high quality. To verify this, the trained embeddings are tested on an analogy task and a similarity to human judgement task. The analogy task, first shown in [13], tests the embeddings on how well they can solve an analogy, like the u("King") - u("Man") + u("Woman") $\approx$ u("Queen") example mentioned earlier. Three words (the left side of the equation) are used to estimate a new vector. Then, this vector is compared to all word embeddings, and the most similar (by cosine similarity) is chosen. The score is the percentage of correct words found. The task is split into two parts: semantic (which captures meaningful relationships) and syntactic (which captures structural relationships). The quality of the word embeddings is also evaluated using the WS353 task, created by Finkelstein et al. [5]. This task contains 353 word pairs with human created similarity scores for each pair. These scores are compared to the cosine similarity between the corresponding word embeddings, using Spearman's rank order correlation coefficient [24]. Better embeddings should have a higher correlation coefficient. For both the analogy and WS353 tasks, any analogy or pair involving words not in the vocabulary is ignored.

The second goal of EV is for the context clue embeddings (h) and subword embeddings (z) to find good estimates of OOV word embeddings. This is evaluated by two tasks for OOV estimation; the definitional nonce task, created by Herbelot and Baroni [7] and the Contextualized Rare Word (CRW) task, created by Khodak et al. [10]. The definitional nonce task contains 300 sentences, each being the first sentence of the Wikipedia page for a nonce word. The goal of this test is to pretend the nonce word is unknown, estimate it using the sentence, and then compare the estimated embedding to its original embedding. Because the sentences are definitions, their contexts are known to be informative. Note that the definitional nonce task compares the nonce estimate to its real location, and therefore must have a real embedding for the nonce. All models trained from WWC are missing 5 nonce words, and therefore only evaluate based on 295 nonces. The second test is the CRW set. The goal of CRW is to estimate OOV words given the word and a set of contexts. It is built on the Rare Word (RW) dataset [12], which has a list of rare words, pairs them with other words, and contains similarity scores for the pair based on human judgements. The goal is to try to estimate the rare words such that their similarity to the paired word correlates with the human scores. CRW extends this, by adding sets of contexts to each rare word. CRW estimates the rare word with different amounts of contexts and judges how well their pair similarity correlates with human judgements. Unlike the definitional nonce task, the CRW task does not require the words to have existing embeddings.

V. Results

Each result we present is the average of 10 trained versions of the corresponding model. Statistical significance is assessed using a one-way ANOVA with a post-hoc Tukey HSD test with a p-value threshold equal to 0.05. For each task, boldface indicates the technique with the statistically significant best performance score.

For the results, we denote the whole Estimator Vectors model as EV, with its word vectors as EV-word, its subword vectors as EV-s, and its context clue vectors as EV-c. We denote the Form-Context model as FCM and Attentive Mimicking as AM, and similarly denote the subword and context only models as FCM-s/AM-s and FCM-c/AM-c respectively. In addition, skipgram is denoted as sg.

[2]EV counts slightly differently than Form-Context and Attentive Mimicking, leading to subword counts of 111968 for the former and 111976 for the latter models. We do not expect this to make a large difference in analysis.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE I
ANALOGY TASK AND WS353 TASK JUDGEMENT

	Semantic	Syntactic	WS353 (ρ)
sg	80.72%	73.66%	0.7147
fastText	82.99%	**76.58%**	0.7132
EV-word	**84.86%**	67.49%	**0.7233**

A. Word Embedding Quality

We compare gensim implementations of skipgram and fastText to the word vectors trained by our model. Note that FCM, AM, and *à la carte* embeddings are based on the skipgram embeddings, so the word embedding quality is particularly important. The results for the analogy test and the WS353 task are shown in Table I.

These results show that EV word vectors are stronger embeddings, due to being trained jointly with subwords and context clues. EV outperforms skipgram and fastText in the Semantic analogy test, along with the WS353 semantic similarity task. However, it performs worse on the Syntactic task. This shows that joint training with subwords and context clues at least enhances the semantic information contained in embeddings, although it may decrease the structural information. This may be mitigated by adding the EV-s vectors, as subword vectors tend to be good at capturing syntactic information [3]. This is shown by fastText, a subword model, which has the strongest Syntactic task score.

B. OOV Embedding Estimation

We also investigate how well EV performs at OOV estimation. We perform this test on the definitional nonce task and the CRW task. We show the results for the definition nonce task in Table II, and for the CRW task in Figure 1. The definitional nonce task measures the rank of the 'real' embedding of the OOV word in the list of nearest neighbors of the estimated embedding. The ranks of each nonce are aggregated using two metrics: MRR and Median Rank. MRR is the Mean Reciprocal Rank, i.e. the average inverse rank, and higher values are better. Since Median Rank measures rank, lower is better (with the vocabulary size 145741 being the worst rank). For the CRW task, each method's correlation with human scores is shown across multiple context sizes, to show how they perform in both low and high context settings.

We compare EV to context based, subword based, and combined methods mentioned earlier. Note that fastText is incompatible with the definitional nonce task as a subword estimation method. This is because fastText's word embeddings are already the sum of its subwords. Therefore, its "real" embedding is the same as its subword estimation, which makes it unable to be judged by the definitional nonce task. As a result, fastText is omitted from the definitional nonce task.

The results for the definitional nonce task (shown in Table II) show that EV performs strongly, although it is not the best model for the task. For the combined subword/context methods (which ideally should perform the best), the Form-Context and Attentive Mimicking models outperform everything, including

TABLE II
DEFINITIONAL NONCE TASK

	MRR	Median Rank
sg (additive)	0.0322	161.6
à la carte	0.0921	44.8
FCM-c	0.0971	52.9
FCM-s	**0.9533**	**1**
FCM	0.8554	**1**
AM-c	0.0985	52.3
AM-s	**0.9532**	**1**
AM	0.8652	**1**
EV-c	0.0848	41.6
EV-s	0.9174	**1**
EV	0.7830	**1**

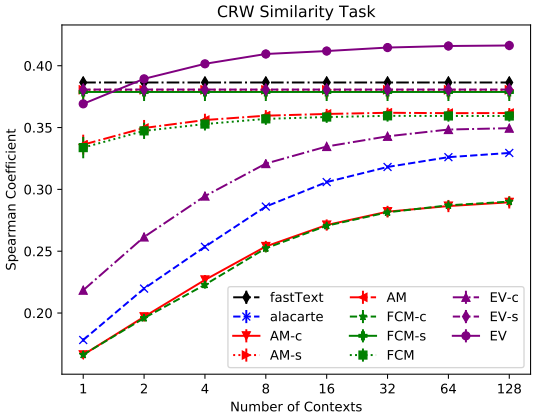

Fig. 1. CRW Task

EV, in MRR. For median rank, they all get the best possible score with a median rank of 1, demonstrating that the Form-Context model, Attentive Mimicking model, and EV estimate a vector that is very close to the correct one. However, interestingly, using the subword only strategies performs better on both measures than their combined strategies (EV-s, FCM-s, and AM-s are better than every other model). This seems unusual, as we would expect context to help the subwords, and a combined representation to be better than just a subword representation. However, this issue can be answered by looking deeper into the definitional nonce task. The definitional nonce task tests words the model already knows (it compares the estimate to the real embedding). Therefore, in any subword model, the subwords of the nonce words were trained with the nonce word specifically, and therefore will generally be good at estimating it. This shows a flaw in the definitional nonce task; although it is supposed to measure how well OOV words can be estimated, it uses already learned words to do it, which may mislead on how well it performs on real OOV words. A "better" task for this is the CRW task, which compares OOV words to other words, and therefore tests words that the models have truly never seen before.

This flaw in the definitional nonce task mainly applies to subword strategies. As such, we also compare each context

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

only strategy. EV perform well using just the context, with a fairly strong rank of 41.6, tied with *à la carte*'s 44.8 (in terms of statistical significance), and beating Form Context model's 52.9 and Attentive Mimicking model's 52.3. The MRR scores between all the context models are statistically the same. This means EV-c and *à la carte* have stronger ranks, but similar MRR to the other context models. We suspect this means that EV context clues have a more 'stable' estimation than the other context estimation, where the competitors' context estimations are sometimes much better but can also be much worse. MRR is affected more by individual ranks, which could explain why the other models maintain an equal MRR but worse median rank compared to EV-c.

Next, we look at the CRW task, shown in Figure 1. Like the above tasks, 10 trials of each method were analyzed, with the figure showing the average results. Statistical significance is assessed as before, using a one-way ANOVA with a post-hoc Tukey HSD test for each context group. All differences in performance are significant, except for fastText, FCM-s, AM-s, and EV-s in all contexts; FCM and AM in all contexts; FCM-c and AM-c in all contexts; and EV with EV-s, fastText, FCM-s, and AM-s in context size 2.

For the CRW task, EV outperforms all other methods when using at least 4 contexts. This demonstrates EV's strong capabilities at estimating OOV words. As mentioned earlier, the CRW task tests a model's ability to estimate words it truly hasn't seen before, and as such we consider this task a better test of OOV word estimation. This shows Estimator Vectors are competitive at OOV estimation.

Like the definitional nonce task, the subword only strategies perform extremely well on the CRW task, (although not as well as the full EV this time). For the Form-Context model, the subword only model once again outperforms the combined Form-Context model, suggesting using only subwords is better than using both context and subwords with the Form-Context model. This finding is also demonstrated by Schick and Schütze [22], where subword based strategies also perform extremely well (better than all other strategies). Schick and Schütze suggest this is due to the fact that CRW was built from the Rare Word dataset, which was originally contructed on words with strong morphologies. Therefore, the rare words in this set have highly meaningful subword context, which means subword estimation strategies should do well.

When looking at context only methods, EV-c is by far the best performing method, with higher scores than FCM-c, AM-c, and *à la carte* in any amount of contexts. This suggests that learning a separate set of context clue embeddings (for the purpose of estimating words) seems to be an effective strategy for better OOV estimation.

Finally, we observe that attention (used in AM) does not seem to help in the definitional nonce task and the CRW task, as AM performs very similarly to FCM, the equivalent model without attention.

Overall, the CRW task shows EV is extremely effective at estimating OOV words.

VI. Conclusion

We propose Estimator Vectors (EV), a word2vec inspired model that learns high quality word embeddings, and allows for good OOV estimates without requiring separate training. The model learns three distinct sets of embeddings: the word embedding itself, along with context clue embeddings and subword embeddings, used for estimating OOV words. We show this model has promising results in both word embedding quality and OOV estimation.

We plan to extend this work. First, we plan to experiment with various weighting strategies, in order to effectively combine context clue and subword words. In addition, we plan to incorporate other information, like position information, to enhance the context clue representations even more.

Furthermore, we plan to investigate ways of combining EV's word, subword, and context clue embeddings to create stronger representations for all words, not just OOV words. With EV's context clue vectors, we can create contextualized embeddings for words using the summation of EV's various embeddings. We plan to investigate how strong these contextualized representations can be, and how well they compare to more complex, deep contextualized representations like those generated by ELMo [18] and BERT [4].

References

[1] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[2] D. Bahdanau, T. Bosc, S. Jastrzebski, E. Grefenstette, P. Vincent, and Y. Bengio, "Learning to compute word embeddings on the fly," *arXiv preprint arXiv:1706.00286*, 2017.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association of Computational Linguistics*, vol. 5, no. 1, pp. 135–146, 2017.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 406–414.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135.

[7] A. Herbelot and M. Baroni, "High-risk learning: acquiring new word vectors from tiny data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 304–309.

[8] F. Horn, "Context encoders as a simple but powerful extension of word2vec," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 10–14.

[9] Z. Hu, T. Chen, K.-W. Chang, and Y. Sun, "Few-shot representation learning for out-of-vocabulary words," *arXiv preprint arXiv:1907.00505*, 2019.

[10] M. Khodak, N. Saunshi, Y. Liang, T. Ma, B. Stewart, and S. Arora, "A la carte embedding: Cheap but effective induction of semantic feature vectors," *arXiv preprint arXiv:1805.05388*, 2018.

[11] A. Lazaridou, M. Marelli, and M. Baroni, "Multimodal word meaning induction from minimal exposure to natural text," *Cognitive science*, vol. 41, pp. 677–705, 2017.

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

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
[15] J. Mu and P. Viswanath, "All-but-the-top: Simple and effective post-processing for word representations," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.
[16] M. Naili, A. H. Chaibi, and H. H. B. Ghezala, "Comparative study of word embedding methods in topic segmentation," *Procedia Computer Science*, vol. 112, pp. 340–349, 2017.
[17] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
[18] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL-HLT*, 2018, pp. 2227–2237.
[19] Y. Pinter, R. Guthrie, and J. Eisenstein, "Mimicking word embeddings using subword RNNs," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 102–112.
[20] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.
[21] T. Schick and H. Schütze, "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)*, 2019, pp. 489–494.
[22] ——, "Learning semantic representations for novel words: Leveraging both form and context," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6965–6973.
[23] C. Shaoul, "The Westbury lab Wikipedia corpus," *Edmonton, AB: University of Alberta*, p. 131, 2010.
[24] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 15, no. 1, pp. 72–101, 1904.
[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE