Beyond Context: A New Perspective for Word Embeddings

Yichu Zhou
School of Computing
University of Utah
flyaway@cs.utah.edu

Vivek Srikumar
School of Computing
University of Utah
svivek@cs.utah.edu

Abstract

Most word embeddings today are trained by optimizing a language modeling goal of scoring words in their context, modeled as a multi-class classification problem. Despite the successes of this assumption, it is incomplete: in addition to its context, orthographical or morphological aspects of words can offer clues about their meaning. In this paper, we define a new modeling framework for training word embeddings that captures this intuition. Our framework is based on the well-studied problem of multi-label classification and, consequently, exposes several design choices for featurizing words and contexts, loss functions for training and score normalization. Indeed, standard models such as CBOW and FASTTEXT are specific choices along each of these axes. We show via experiments that by combining feature engineering with embedding learning, our method can outperform CBOW using only 10% of the training data in both the standard word embedding evaluations and also text classification experiments.

1 Introduction

The distributional hypothesis (Firth, 1935; Harris, 1954) has been a cornerstone in NLP. For example, Firth (1935) writes:

> ...the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously.

Operationally, in modern NLP, word embeddings capture this idea and are typically trained using neural language models or word collocations (e.g. Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013b; Pennington et al., 2014; Peters et al., 2018; Devlin et al., 2018).

Is word meaning exclusively defined by its context? In this paper, we argue that while the word usage plays a crucial role in defining its meaning (perhaps, centrally so), it is not the only mechanism that endows meaning to words. Indeed, Firth writes in the paragraph before the above quote:

> ...a certain component of the meaning of a word is described when you say what sort of a word it is, that is when you identify it morphologically...

The composition of a word, (i.e., its orthography and morphology) may offer cues about its meaning even if the word is not commonly used, thus allowing us to understand unseen words. For example, we can elide over misspellings of words (e.g., *Bbeijing*) because we observe the similarities in the orthography between words.

By ignoring word-level information, many existing off-the-shelf word embedding approaches suffer from two shortcomings. First, they need a great amount of training data to get high quality word embeddings. Second, even with large amounts of training data, some words (e.g., neologisms, misspellings, technical terms) will not be seen frequently enough to provide statistical support for good embeddings.

In this paper, we are motivated by the observation that both the context of a word and its own internal information contribute to word meaning. To model this in an easy-to-extend manner, we need a new perspective about training word embeddings that not only admits arbitrary word and context features, but also supports conceptual tools to systematically reason about the various model design aspects in terms of familiar modeling techniques.

A common method for training word embeddings is to construct a word prediction problem, and obtain the word embeddings as a side effect. One instantiation of the word prediction task, namely CBOW (Mikolov et al., 2013a), frames it

as the multi-class classification problem of predicting a word given a context. We argue that the task is more appropriately framed as multi-label classification — multiple words can fit in the same context. Moreover, since the label set (all words) is massive, word prediction is an instance of *eXtreme* Multi-label Learning (XML) (Balasubramanian and Lebanon, 2012; Bhatia et al., 2015; Bi and Kwok, 2013, inter alia).

Framing word prediction as an XML problem allows us to define a unifying framework for word embeddings. Consequently, we can systematically analyze the problem of training word embeddings using lessons from the XML literature. In particular, we can featurize both inputs and outputs — in our case, contexts and words. Apart from featurization, loss functions and normalization of probability are also design choices available. We show that our approach subsumes several standard word embedding learning methods: specific design choices give us familiar models such as CBOW (Mikolov et al., 2013a) and FASTTEXT (Bojanowski et al., 2017)[1].

Our experiments study the interplay between the amount of data needed to train embeddings, and the features for words and contexts. We show that, when trained on the same amount of data, using word and context features outperforms the original CBOW and FASTTEXT on both the standard analogy evaluation and a variant where words have introduce typographical errors. Featurizing words and contexts reduces data dependency for training and can achieve similar results as CBOW and FASTTEXT trained on a 10x larger corpus. Finally, we also show that the trained embeddings offer better representations for an text classification evaluation.

In summary, the contributions of this work are: (i) We propose a new family of models for word embeddings that allow both word orthography *and* context to inform its embeddings via user designed features. (ii) We show that our model family generalizes several well-known methods such as CBOW and FASTTEXT. (iii) Our experiments show that exploiting word and context features gives better embeddings using significantly lower amounts of training data. (iv) Our experiments also show that while global normalization is the more appropriate formulation, in practice, the average number of words in a context is too small for global normalization to prove advantageous.

[1]In order to have a fair comparison, we always use the CBOW variant of FASTTEXT in this paper.

2 Preliminaries & Notation

In this section, we will briefly look at the tasks of word prediction and extreme multi-label classification with the goal of defining notation.

Word prediction The task of word prediction is commonly used to train word embeddings (e.g. Bengio et al., 2003; Mikolov et al., 2013b). A typical example of this class of models (that includes CBOW and several others) frames the problem as using the context around a word to predict it. The context is defined via a fixed-size window around the word to be predicted.

Suppose y denotes a target word and x represents its context. Then CBOW embeddings are trained by minimizing the log loss:

$$\arg\min_{\theta} \sum_{(x,y)\in\mathcal{D}} -\log P(y|x;\theta) \qquad (1)$$

Here, $\mathcal{D}$ is the set of all training documents and θ are the parameters which defined the probability distribution and are learned. The trained word embeddings are part of the learned parameters.

Extreme Multi-label Learning Suppose we have a classification task with a set of labels $\mathcal{L}$. A multi-label classification problem (e.g. Zhang and Zhou, 2014) is one where inputs can be associated with more than one label. Given an input x, the goal is to predict a subset $\mathcal{Y}$ of the label set. One general strategy to model a multi-label problem is to define a scoring function $f(x, \mathcal{Y}; \theta)$ which uses parameters θ to score sets $\mathcal{Y} \subseteq \mathcal{L}$ as being correct for the input x. We can train such a model by minimizing a loss function ℓ over a training set $\mathcal{D}$:

$$\arg\min_{\theta} \sum_{(x,\mathcal{Y})\in\mathcal{D}} \ell\left(f(x, \mathcal{Y}; \theta)\right) \qquad (2)$$

When the label set $\mathcal{L}$ is large, both training and prediction can become problematic because enumerating all subsets of $\mathcal{L}$ is infeasible. We call such problems eXtreme multi-label learning (XML) problems (e.g. Bhatia et al., 2015).

3 Rethinking Word Embeddings

In this section, we will look at assumptions that underlie the use of word prediction to train word embeddings.

Words & Features Many embedding methods are built on the assumption that the context defines the meaning of a word, thus accounting for the pervasiveness of the word prediction task to train word embeddings. However, we argue this assumption is incomplete.

The meaning of a word is defined not only by its context, but also the word itself. For example, consider the word *googlize*. Such made-up words may have only limited or no context. Yet, we may be able to infer their meaning (devoid of context) by appealing to our understanding of their parts. In our example, the word is composed of *google* and *-ize*, both of which have their own meanings and the composition (*google* + *-ize*) gives cues as to what *googlize* may mean. A reader may use their understanding of the word *google* and the fact that *-ize* is a common suffix to create verbs to hypothesize the meaning of the word.

The above example illustrates the following principle: *A word is not the smallest meaning unit, but the most common one.* We argue that we should utilize the internal information of words when we train word embeddings.

Some recent work (e.g. Pinter et al., 2017; Kim et al., 2018; Bojanowski et al., 2017; Schick and Schütze, 2018) applies our assumptions implicitly by using character-level information to embed words. While character-based features help capture the internal structure of the word, several other aspects may be helpful, e.g. linguistically motivated prefixes and suffixes, the shape of the word and other possible features. §4.4 describes the various choices we explore.

Word Prediction as XML The second inherent assumption in word prediction is that we can frame the problem of predicting a word that fits a context as a multi-class classification problem. However, in nearly all contexts, more than one word could fit. For example, consider the sentence *The running _____ is chasing after a rabbit*. It can be completed with many words filling the blank, such as *fox*, *dog*, *hound*.

Seen this way, word prediction models are better framed as the multi-label classification problem of *using the context to predict all words that could occur in the context*. In the example above, we would use the sentence with the blank to jointly predict all the words such as *fox, dog, hound* that can occupy the blank. Using the notation from §2, the input x to the problem is a context and all words that could occur in the context form the label set $\mathcal{Y}$ for that input. The label set is a subset of all labels $\mathcal{L}$, i.e., the entire vocabulary. Following this intuition, in the rest of the paper, we will use labels and words interchangeably. Note that since the vocabulary is large, we have an extreme multi-label learning problem at hand.

4 A unifying framework

In this section, we will formalize the intuition illustrated in §3. We will see that this effort reveals several design choices involving normalization, loss functions, label costs, and featurization.

4.1 Modeling Words in Context

Our goal is to frame word prediction as a multi-label classification problem to (i) predict a subset of words for a context, and, (ii) generate embeddings for each word.

Suppose we have a word y in a context x. Let the functions ϕ and ψ denote any feature functions that operate on the word and context respectively. We model the score of the word y given the context x by projecting their feature representations into a common d dimensional space with two matrices V and W. The matrices are the parameters to be learned during training. Using these projections, the scoring function of a pair (x, y) is defined as:

$$S(x, y) = [W\phi(y)]^T [V\psi(x)] \tag{3}$$

Using the score for a pair (x, y), we can now assign scores to a set of words for a context. In any context, some words are more frequent than others. Suppose we denote the frequency of a label y in a context x as $n_x(y)$. We can then define the score of a set of words $\mathcal{Y}$ as the weighted sum of each word in the set:

$$Score(x, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} n_x(y) S(x, y) \tag{4}$$

The matrices V and W will be used to compute word embeddings; we will discuss this further in the comparison to CBOW in §5.2.

4.2 Normalization and Loss Functions

As a prelude to defining loss functions for training, we need to decide how to normalize the scoring functions defined above: we have the choice of local or global normalization. With local normalization, we contrast each label in the set of true labels against all other labels individually; with

global normalization, we contrast the true label set against other possible *subsets* of labels.

As an illustration, suppose the label set $\mathcal{Y}$ for a context x contains two labels y_1 and y_2. With local normalization, we seek parameters that simultaneously make $S(x, y_1)$ higher than $S(x, y_2)$ *and* $S(x, y_2)$ higher than $S(x, y_1)$. Moreover, local normalization does not prevent a third label y_3 from having a positive score as long as it is less than the scores of the valid labels. As a result, the set of highest scoring labels could be invalid even though all the local constraints are satisfied. To fix this, we can design a globally normalized loss function that demands the score of valid subset $\mathcal{Y}$ to be higher than all other subsets of $\mathcal{L}$.

Irrespective of whether the scores are locally or globally normalized, for the XML problem of word prediction, we can use several loss functions for training. To compare to CBOW and FASTTEXT, we will focus on log loss here. We refer the interested reader to the supplementary material for details about the global and local hinge loss for the problem.

For a locally normalized log loss, the probability of a pair (x, y) can be defined as:

$$P(y|x) = \frac{e^{S(x,y)}}{\sum_{y' \in \mathcal{L}} e^{S(x,y')}} \tag{5}$$

The local log loss is defined as the negative log of this probability for a word y that occurs in a context x. Note that each valid word contrasts against all other words in the vocabulary $\mathcal{L}$.

For a globally normalized log loss, the probability of a label $\mathcal{Y}$ for an input x is:

$$P(\mathcal{Y}|x) = \frac{e^{Score(x,\mathcal{Y})}}{\sum_{\overline{\mathcal{Y}} \subseteq \mathcal{L}} e^{Score(x,\overline{\mathcal{Y}})}} \tag{6}$$

The global log loss is the negative log of this probability for a set $\mathcal{Y}$ of words that are valid in a context x. Note that the valid set $\mathcal{Y}$ contrasts against all other possible subsets of the vocabulary.

4.3 Training

A final consideration for training concerns the fact that frequent words can dominate the loss function unless special care is taken. Sub-sampling of frequent words is commonly used to deal with this problem, where a word y will be retained with probability $g(y)$, which is inversely proportional to its frequency. In this work, we adopt the strategy of *cost-sensitive classification* which lets directly augment the loss functions. Different labels (i.e., words) are assigned different costs based on their frequency using a cost function $c_x(y)$.

Suppose we have a word y in a context x, whose frequency is $n_x(y)$. We will use the word subsampling probability $g(y)$ above to define the cost for the pair (x, y) as the expected frequency of the pair in the training set. That is,

$$c_x(y) = n_x(y)g(y) \tag{7}$$

We use the subsampling frequency from word2vec to define $g(y)$ as follows:[2]

$$g(y) = \min\left(1, \left(\sqrt{\frac{n(y)}{\alpha}} + 1\right) \cdot \frac{\alpha}{n(y)}\right) \tag{8}$$

Here, $n(y)$ is the context-independent frequency of label y in the whole corpus and α is a hyperparameter.

Using these costs, the final form of local log loss can be shown to be:

$$\begin{aligned}
\ell_{ll}(x, \mathcal{Y}) &= -\sum_{y \in \mathcal{Y}} c_x(y) \log P(y|x) \\
&= -\sum_{y \in \mathcal{Y}} c_x(y) \log \frac{e^{S(x,y)}}{\sum_{y' \in \mathcal{L}} e^{S(x,y')}} \\
&= -\sum_{y \in \mathcal{Y}} c_x(y) S(x, y) + \sum_{y \in \mathcal{Y}} c_x(y) \log \sum_{y' \in \mathcal{L}} e^{S(x,y')}
\end{aligned} \tag{9}$$

For global loss, we can achieve a similar effect by rewriting the weighting term in Eq. 4 with the cost. The global log loss is:

$$\begin{aligned}
\ell_{gl}(x, \mathcal{Y}) &= -\log P(\mathcal{Y}|x) \\
&= -\log \frac{\prod_{y \in \mathcal{Y}} \exp\left(c_x(y) S(x, y)\right)}{\sum_{\overline{\mathcal{Y}} \subseteq \mathcal{L}} \prod_{y' \in \overline{\mathcal{Y}}} \exp\left(c_x(y') S(x, y')\right)}
\end{aligned} \tag{10}$$

Direcly computing the denominator is expensive. However, we know that:

$$\begin{aligned}
&\sum_{\overline{\mathcal{Y}} \subseteq \mathcal{L}} \prod_{y' \in \overline{\mathcal{Y}}} \exp\left(c_x(y') S(x, y')\right) \\
&= \exp(0) + \exp\left(c_x(y_1) S(x, y_1)\right) + \exp\left(c_x(y_2) S(x, y_2)\right) + \dots \\
&+ \exp\left(c_x(y_1) S(x, y_1)\right) \exp\left(c_x(y_2) S(x, y_2)\right) \dots \\
&+ \exp\left(c_x(y_1) S(x, y_1)\right) \dots \exp\left(c_x(y_{|\mathcal{L}|}) S(x, y_{|\mathcal{L}|})\right) \\
&= \prod_{y \in \mathcal{L}} \left(\exp\left(c_x(y) S(x, y)\right) + 1\right)
\end{aligned} \tag{11}$$

[2]The implementation and the description in the paper are different. We are following the implementation and not the paper. The implementation can be found here:https://code.google.com/archive/p/word2vec/

Then, the global log loss can be rewritten as:

$$\ell_{gl}(x, \mathcal{Y}) = -\sum_{y \in \mathcal{Y}} c_x(y) S(x, y) + \sum_{y' \in \mathcal{L}} \log\left(\exp\left(c_x(y') S(x, y')\right) + 1\right) \tag{12}$$

Note that both local and global models are dominated by the $O(|\mathcal{L}|)$ summation, which suggests they have same computational cost.

4.4 Featurizing Words and Contexts

In the scoring function of a pair (x, y), i.e., Eq. 3, we use two feature functions ϕ and ψ to extract features from the label and context respectively. This design choice dictates the information we wish to provide to the model about words and contexts. CBOW and word2vecf uses indicators for the target words, while FASTTEXT uses both the words and their constituent character ngrams. For context, CBOW and FASTTEXT aggregate the same features as the target word, but over the context words. Word2vecf (Levy and Goldberg, 2014) uses dependency information to featurize the context. We generalize these by allowing user or domain dependent features. The output of feature functions ϕ and ψ is a sparse vector and each dimension is a binary value, which indicates the existence of corresponding feature.

Though the ϕ and ψ functions, we can easily incorporate extra information into word embeddings from other resources. For example, we can use hand-crafted gazetteers to indicate whether two words can belong to the same type. If both *Beijing* and *Paris* are in a list of locations, we can identify similarity between the words without any context. Such resources may provide information that we can not learn from the context.

5 Analysis of Modeling Choices

In this section, we discuss the advantages of the framework described in §4 and its connections to CBOW embeddings.

5.1 Word Embeddings

In the framework described in §4, we are not learning the word embeddings, but feature embeddings. Each column of W and V represents the embedding of a certain feature. Given features for a word, we can compute its the embeddings of a word w by computing either $W\phi(w)$ or $V\psi(w)$. This perspective of feature embeddings presents three advantages.

First, we provide a mechanism for incorporating human knowledge into word embeddings. Feature engineering can be combined with our model naturally. Moreover, using informative features can help produce high quality embeddings even with smaller document collections (e.g., all documents related to a specific project in a company).

Second, because a feature can be shared by different words, each training update for a feature will update all the word embeddings containing this feature. This can lead to better generalization.

Third, it presents an elegant solution for the out-of-vacabulary (OOV) problem. Once we define the feature template, we can extract features of any word, then we can compute the embedding for it. Some recent work (Pinter et al., 2017; Kim et al., 2018; Zhao et al., 2018; Artetxe et al., 2018) address the OOV problem using pre-trained embeddings and mimicking them by training a second model using substrings of a given word. Instead, here we can use arbitrary features and do not need pre-trained embeddings.

5.2 CBOW: A Specific Instance

In this subsection, we will show that CBOW is an instance of our framework. We can rewrite the overall loss on a given dataset D using the local log loss function from Eq. 9 as:

$$\begin{aligned}\ell_{ll}(D, \theta) &= \sum_{(x, \mathcal{Y}) \in D} \sum_{y \in \mathcal{Y}} -c_x(y) \log P(y|x) \\ &= \sum_{(x, \mathcal{Y}) \in D} \sum_{y \in \mathcal{Y}} -n_x(y) g(y) \log P(y|x)\end{aligned} \tag{13}$$

As before, $n_x(y)$ is the frequency of label y for the context x, $g(y)$ is the probability of keeping this label y and θ denotes the matrices V and W.

Now, suppose we have a different dataset D' that is constructed from D as follows: for every $(x, \mathcal{Y}) \in D$, for every $y \in \mathcal{Y}$, add $n_x(y)$ copies of (x, y) to the new dataset D'. Now, D' is a *multi-class* classification dataset, where each x is associated with only one label y in a single example. More importantly, D' is exactly the input-output pairing used to train CBOW. With this new dataset representation, we can write the total loss over the dataset D' as:

$$\ell_{\text{CBOW}}(D', \theta) = \sum_{(x, y) \in D'} -\log P(y|x) \tag{14}$$

In both $\ell_{ll}(D, \theta)$ and $\ell_{\text{CBOW}}(D', \theta)$, $P(y|x)$ is given by Equation 5. If the label features $\phi(y)$ consists only of indicators for the label (i.e., the target word) and the context features $\psi(x)$ is the average of the indicators for the words in the context, then these two loss functions are identical. In other words, CBOW is an instance of our model that minimizes local log loss, and uses these specific features. A similar argument applies for FASTTEXT and word2vecf as well.

There are two important differences: First, CBOW used sub-sampling to reduce the impact of frequent words, while we use costs $c_x(y)$ for this purpose. Second, in CBOW, the matrix V is used as word embeddings. As mentioned above, in fact, both V and W generate word embeddings for a word w as $W\phi(w)$ and $V\psi(w)$. Based on preliminary experiments, we observed that concatenating $W\phi(w)$ and $V\psi(w)$ produces the best embeddings. In this work, we use this concatenation strategy to embed words.

5.3 Local vs. Global Normalization

Given the two normalization methods (§4.2), which one should we pick? In theory, local normalization for word prediction can be problematic as described in §4.2. However, in practice, CBOW, FASTTEXT and word2vecf all use multiclass classification (i.e., local normalization) and work well. This apparent gap between the theory and practice can be explained by the observation that while many words may indeed fit in a given context, the key criterion is the *label density* — that is, the ratio of the number of valid labels ($|\mathcal{Y}|$) to the number of all possible labels ($|\mathcal{L}|$). For the XML problem of word prediction, the label density is small enough that the problem can be approximated as a multiclass problem. In other words, because $|\mathcal{L}|$ is large, the average value of $\frac{|\mathcal{Y}|}{|\mathcal{L}|}$ is small enough that it is close to $\frac{1}{|\mathcal{L}|}$. Unless we have dense labels in specialized document collections, we do not expect globally normalized models to outperform locally normalized ones.

6 Experiments

In this section, we empirically verify that our approach: (i) achieves similar performance as CBOW using the same training set and features, (ii) can outperform CBOW and FASTTEXT with only 10% of the training data if extra features are used, (iii) creates embeddings that generalize better by evaluating analogies on datasets with misspellings, and, (iv) offers a better feature representation for an extrinsic evaluation of text classification. The overall goal of our experiments is to understand the dependence between dataset size, features, and the quality of embeddings produced.

We conduct our experiments on the training set of the 1 billion language model benchmark corpus (Chelba et al., 2014), consisting of 700M words (with 550K unique words). For all experiments, the embedding size of W and V is 300 and the context window size is set to five words to the left and right of a word. The hyper-parameter α from Eq. 8 is 0.001 (following CBOW's implementation). To compare to CBOW and FASTTEXT, we use log loss on all our experiments. We optimized the loss using Adam (Kingma and Ba, 2015) and used dropout with keep probability 0.6. Based on preliminary trials, we projected the matrix W onto a unit ball. For efficient learning, we also used the negative sampling approach from word2vec. More experiment setup details are available in supplementary material.

Table 1 summarizes all the feature templates we used. In the table, the feature Gazetteers is a set of lists containing names of entities from Wikipedia, grouped by category. Each list represents an entity type such as cities, organizations, days of the week, etc. If the current word matches one of the words in the list, the corresponding feature is activated. The Quirk feature is a collection of prefixes and suffixes types from Quirk et al. (1987). For example, *un-* is a negative prefix and *-ness* is a noun suffix. For the context feature function ψ, we summed the above features over all the context words.

We implement our model using Pytorch[3]. We train our model on a Nvidia DGX machine using one Tesla (16G video memory) GPU. We train 70 epochs for both local log loss and global log loss. For all experiments, prediction accuracy is used as the evaluation metric.

6.1 Analogy Evaluation

In this subsection, we evaluate our models on these traditional analogy evaluation tests (Mikolov et al., 2013a), in particular the Google and the MSR analogy tests.[4] These evaluations focus on an

[3] https://pytorch.org/

[4] Although the Google and Microsoft analogy datasets have been shown to be problematic (Linzen, 2016), they are the most commonly used evaluation datasets.

Features	Description
Word	word itself. e.g. <Beijing>
Prefix and suffix	Prefix and suffix up to length 3. e.g. 0#1#B, 0#2#Be, 0#3#Bei
substring	word substrings from length 3 to 6 e.g. n-grams@Bei, n-grams@eij
Quirk	Known prefix and suffix types from Quirk et al. (1987), e.g. quirk@Conversion
Word shape	word shape of the word. e.g. shape@Xxxx
Gazetteers	Indicators for gazetteer matches e.g. gaz@Locations
MISC	isalpha? isPrintable? isdigit? e.g. special@alpha special@printable
Default	The default feature for every word, functions as a bias feature.

Table 1: Extra feature templates for learning word embeddings. This table uses the word *Beijing* as an example. We use gazetteers from the EDISON package (Sammons et al., 2016).

Model	Features	Google	MSR
CBOW	word	0.398	0.463
Fasttext	subwords	0.424	0.584
local model	word	0.416	0.426
global model	word	0.431	0.410
local model	all	**0.494**	**0.724**
global model	all	**0.480**	**0.692**

Table 2: Comparison between different models trained on 10% corpus using a closed vocabulary. The last two rows use features from Table 1.

analogy question of the form *A:B::C:?*, and the goal is to use word embeddings to find the word that best fills the question slot. Because these results are highly related to the vocabulary used to search for the answers, we divide this evaluation into two different vocabularies: closed and open. Closed vocabulary means it comes directly from the training set; while open vocabulary means it is composed of words coming from the training set and evaluation set. Open vocabulary can ensure there are no OOV words during the evaluation. Our model and FASTTEXT can generate embeddings for OOV words, while CBOW can not. We evaluate CBOW only on the closed vocabulary.

First, we compare our model with CBOW and FASTTEXT trained only on 10% of the corpus with a closed vocabulary (Table 2). Using only words as features, our model achieves better performance than CBOW on the Google analogy set, and close performance on the MSR set. This difference might be caused by the different optimization algorithms. The global model is close in performance to the local model — this is is expected because the global model is approximated by local model when the density of label is small, and the global model is optimizing for a more stringent goal. With all features, our models (both local and global) outperform both CBOW and FASTTEXT by a large margin.

Model	Features	Google	MSR
Fasttext-10%	subwords	0.310	0.507
Fasttext-100%	subwords	**0.490**	0.686
local model	all	0.415	**0.763**
global model	all	0.395	0.719

Table 3: Comparison between our models and FASTTEXT using open vocabulary. FASTTEXT is trained on 10% and 100% corpus; our models are all trained on 10% corpus with extra features defined in Table 1.

Second, we compare our model (trained on 10% of the data) with FASTTEXT using an open vocabulary (Table 3). The first row FASTTEXT is trained on 10% of the data. Our model significantly outperforms it. To understand the impact of the extra features, we compare it with FASTTEXT trained on the entire corpus (second row). On the MSR set, our 10% model with extra feature still outperforms FASTTEXT. We believe the underperformance of our 10% model against the 100% FASTTEXT embeddings on the Google data may be due to the fact that the data contains more complex relations between words, and our feature templates may not be expressive enough.

6.2 Analogies with Typographical Errors

One important advantage of our model is it can dynamically produce embeddings for unseen words using the feature embeddings. To verify its usefulness, we need a set of words that do not have any context in a training corpus, but are still meaningful. Misspelt words are a natural set of such words. Such words usually do not occur in the training set and their meanings are defined by their orthographical similarity to the correct word. Using only word features, CBOW can not deal with OOV words; we only compare against FASTTEXT for this task.

	Google			MSR		
	degree 1	degree 2	degree 3	degree 1	degree 2	degree 3
Fasttext-10%	0.212	0.159	0.035	0.374	0.259	0.073
Fasttext-100%	0.348	0.217	0.036	0.503	0.323	0.080
local model	**0.373**	**0.263**	**0.066**	**0.685**	**0.531**	**0.185**
global model	0.341	0.227	0.054	0.634	0.449	0.134

Table 4: Misspelling evaluation results. Different degrees mean how many words in the quadruple has been changed into a misspelling word. FASTTEXT is trained on 10% and 100% corpus while our model trained on 10% corpus with extra features. CBOW can not generate embeddings for OOV words, which means we can not compare with CBOW on this task.

We create new misspelling datasets by randomly replacing, deleting or inserting one character of each word in MSR and Google analogy datasets. Then we apply the standard analogy test on these misspelling datasets. Table 4 shows the misspelling analogy test results. Different degrees indicate how many words in the analogy quadruple have been changed into a misspelling word. The results show that our model can outperform FASTTEXT in every degree, indicating extra features can capture these similarities between misspelling words and their corrections.

6.3 Extrinsic Task: Dataless Classification

In this subsection, we report the results of an extrinsic evaluation of our trained embeddings. In most extrinsic tasks, embeddings are usually used as a representation of examples which are the inputs of a classifier. As a consequence, the performance on these extrinsic tasks is determined by two factors: the quality of representation and the quality of that classifier. Assigning credit to these two factors for any changes in classifier performance can be difficult.

What we need is a task where performance depends only on the quality of the feature representation. Dataless text classification (Chang et al., 2008) has this characteristic, where the goal is to predict a label for a document without any labeled data. Following Chang et al. (2008), we use the 20 Newsgroup dataset (Lang, 1995) to construct ten binary classification problems. Each label in this dataset is mapped to a short list of words that describe the label, as specified by the original work.

We frame this task as a nearest neighbor task. Each word in the documents and label expansions will be assigned an embedding. We use the average of all the words in the documents and expansions as their embeddings and measure Euclidean distance between the labels and a document. For each document, closest label will be chosen as the predicted label.

Models	Features	Accuracy
CBOW-10%	word	0.524
CBOW-100%	word	0.569
Fastext-10%	subwords	0.509
Fastext-100%	subwords	0.537
local model	all	**0.593**
global model	all	0.567

Table 5: Dataless task evaluation results. FASTTEXT and CBOW are trained on 100% corpus while our model trained on only 10% corpus.

Table 5 shows the average accuracy of these ten binary classification problem. By using extra features, our model can substantially outperform FASTTEXT and CBOW, even when they were trained on 100% of the corpus.

7 Discussion and Related Work

Word and Label Embeddings Word embeddings are ubiquitous across NLP and word prediction is a common approach to train them. Our modeling unifies this task with multi-label classification. There are other approaches for training word embeddings, such as skipgram (Mikolov et al., 2013b), and Glove (Pennington et al., 2014). A similar formalization of such approaches is a direction for future research.

The idea of embedding labels in a classification task has been previously explored (e.g. Amit et al., 2007; Weston et al., 2011; Srikumar and Manning, 2014). In this paper, we make a formal connection between these lines of work and the word embedding literature.

Feature Engineering The feature functions ϕ and ψ in the scoring function (Eq. 3) provide a systematic way to combine feature engineering and embedding learning. Consequently, the rich history of feature engineering in NLP becomes applicable for constructing word embeddings.

In this work, we use context independent features such as prefix, suffix and word shape. Such character-level features have been used for other NLP tasks (e.g. Sapkota et al., 2015) However, contextual features can also be incorporated into this framework (e.g. Akbik et al., 2018). Furthermore, such contextual features could be informed by traditional feature functions such as POS tags of neighboring words.

eXtreme Multi-label Learning (XML) Embedding-based methods are widely used in extreme multi-label classification (e.g. Bhatia et al., 2015; Balasubramanian and Lebanon, 2012; Bi and Kwok, 2013). However, all these embedding-based methods are not used in word prediction context. In this paper, we point out that essentially, they are the same problem. This paper is the first attempt to combine these two areas; fruitful exchange of ideas between the them may lead both to better predictors and embeddings.

8 Conclusion

In this paper, we argue that assumption that context defines meaning, which is used by most word embedding models is incomplete. Besides the context, the internal information of a word also characterizes its meaning. Using this assumption, we reframe the word prediction task as a multi-label classification problem. This new perspective reveals a family of embedding learning models, which allows different featurizations, loss functions and normalizations. We show that CBOW is one particular instance of our framework, with specific choices for these options. Our experiments demonstrate the value of word and context features for constructing word embeddings.

Acknowledgments

We would like to thank members of the Utah NLP group for several valuable discussions and the anonymous reviewers for their feedback. This work was supported in part by a grant from the US Israel Binational Science Foundation (BSF 2016257) and a hardware gift from NVIDIA Corporation.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Yonatan Amit, Michael Fink, Nathan Srebro, and Shimon Ullman. 2007. Uncovering shared structures in multiclass classification. In *Proceedings of the 24th international conference on Machine learning*, pages 17–24. ACM.

Mikel Artetxe, Gorka Labaka, Inigo Lopez-Gazpio, and Eneko Agirre. 2018. Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 282–291.

Krishnakumar Balasubramanian and Guy Lebanon. 2012. The landmark selection method for multiple output prediction. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 283–290. Omnipress.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, pages 730–738.

Wei Bi and James Kwok. 2013. Efficient multi-label classification with many labels. In *International Conference on Machine Learning*, pages 405–413.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: dataless classification. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 2*, pages 830–835. AAAI Press.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. 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*.

John Rupert Firth. 1935. The technique of semantics. *Transactions of the philological society*, 34(1):36–73.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

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

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1987. A comprehensive grammar of the english language. *Studies in Second Language Acquisition*, 9(1):109111.

Mark Sammons, Christos Christodoulopoulos, Parisa Kordjamshidi, Daniel Khashabi, Vivek Srikumar, Paul Vijayakumar, Mazin Bokhari, Xinbo Wu, and Dan Roth. 2016. EDISON: Feature Extraction for NLP, Simplified. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102.

Timo Schick and Hinrich Schütze. 2018. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the 33rd national conference on Artificial intelligence*.

Vivek Srikumar and Christopher D Manning. 2014. Learning distributed representations for structured output prediction. In *Advances in Neural Information Processing Systems*, pages 3266–3274.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

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

A Local and Global Hinge Loss

We model the score of a word y given the context x by projecting them into a common space with two matrices V and W. The scoring function of pair (x, y) is defined as:

$$S(x, y) = [W\phi(y)]^T [V\psi(x)] \qquad (15)$$

where ϕ and ψ are two feature functions. This scoring function is the same as we saw for log loss in the body of the paper.

A.1 Local Hinge Loss

In this subsection, we will define local hinge loss for word embedding learning under multi-label formulation. Local hinge loss means we want to contrast the target word against all other words in the vocabulary. In our modeling framework, the local hinge loss can be defined as:

$$\ell_{lh}(x, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} \left[c_x(y) \max_{y' \in L} \left[S(x, y') + \Delta(y, y') - S(x, y) \right] \right] \tag{16}$$

Here, $\Delta(y, y')$ represents the Hamming distance between the ground truth y and any other label y'. That is, it takes the value 1 when $y \neq y'$ and 0 otherwise.

A.2 Global Hinge Loss

For the local hinge loss above, we treat each label associated with the input x as being independent. In contrast, for the global hinge loss, we compare scores for subsets of labels. Given a set of ground truth labels for an example, our goal is to maximize the gap between this set and *all* other subsets of labels.

Let us formalize this intuition. As before, let x denote a context. Let $\mathcal{Y}$ denote its corresponding gold label set. Every set of labels that does not agree with this subset $\mathcal{Y}$ will be penalized. We define the global hinge loss to be:

$$\ell_{gh}(x, \mathcal{Y}) = \max_{\overline{\mathcal{Y}} \subseteq \mathcal{L}} \left[\sum_{y \in \mathcal{Y}} S(x, y) + \Delta(\mathcal{Y}, \overline{\mathcal{Y}}) - \sum_{y \in \overline{\mathcal{Y}}} S(x, y) \right] \tag{17}$$

Here, Δ denotes the cost-sensitive Hamming distance between the true set of labels $\mathcal{Y}$ and any other set of labels $\overline{\mathcal{Y}}$. It can be written as:

$$\Delta(\mathcal{Y}, \overline{\mathcal{Y}}) = \sum_{y \in \mathcal{L}} \mathbb{1}[y \in \mathcal{Y}] \oplus \mathbb{1}[y \in \overline{\mathcal{Y}}] c_x(y)$$

where $\mathbb{1}[\cdot]$ is the indicator function and $\oplus$ represents the XOR operation. Essentially, the loss adds up the costs $c_x(y)$ for all labels that are mispredicted, either by mistakenly including a label that is not in the gold set, or by missing one that is in it. The costs $c_x(y)$ are the same as defined in the main body of the paper.

Combining the equations above and reorganizing the summations, we can get the final form of the global hinge loss:

$$\ell_{gh}(x, \mathcal{Y}) = \sum_{y \in \mathcal{Y}} \max\left(c_x(y) - S(x, y), 0\right) + \sum_{y \notin \mathcal{Y}} \max(c_x(y) + S(x, y), 0) \tag{18}$$

The global hinge loss can be interpreted as a sum of two terms: the first term penalizes labels that should have been predicted, but have scores less than their costs, and the second term penalizes labels that should *not* have been predicted, but have scores that are more than the negative costs.

As in the log loss in the body of the paper, with hinge loss as well, both global and local normalization require $O(|\mathcal{L}|)$ computation.

B Hyper-parameters for experiments

Table 6 shows the hyper-parameters for our experiments.

Hyper-parameters	Setting
10% corpus vocabulary size	155, 525
100% corpus vocabulary size	552, 402
Window size	5
α in subsamping	0.001
embedding dimension	300
minimum word frequency	5
negative sampling size	6
dropout probability	0.6

Table 6: Hyper-parameters in the all experiments.