Embedding Senses via Dictionary Bootstrapping

Byungkon Kang
byungkon@ajou.ac.kr
Ajou University
Suwon, Korea

Kyung-Ah Sohn
kasohn@ajou.ac.kr
Ajou University
Suwon, Korea

Abstract

This paper addresses the problem of embedding senses of a plain word according to its context. Natural language is inherently ambiguous, due to the presence of many multi-sensed words. Such ambiguity might have undesirable influence over many text-mining tools, including word embedding. Traditional word embedding techniques have focused on identifying which words tend to co-occur with one another in order to derive close embeddings for such words. However, the effectiveness of this approach is largely susceptible to the validity and neutrality of the training corpus. To address this problem, we propose to use the dictionary as the authoritative corpus for computing the word embeddings. The basic idea is to simultaneously embed the definition sentence while disambiguating words. Since dictionaries list a set of disambiguated senses, the proposed procedure yields sense embeddings which exhibit semantic characteristics comparable to plain word embeddings.

1 INTRODUCTION

Recent advances in neural network technology have influenced the field of natural language processing (NLP) to progress towards using distributed representations for solving its problems. One notable example is the concept of *word embedding*, or word vectors, popularized by Mikolov et al. [2013]. This framework allows one to represent the semantics of each word in the vocabulary as a Euclidean vector. The basic concept is to have words with similar semantics be "embedded" as nearby points in the Euclidean space. The initial success of Mikolov et al. [2013] has led to the proposal of many variants (Hill et al. [2016], Iacobacci et al. [2015], Pennington et al. [2014]).

The advantage of operating with embedding vectors is that they allow for continuous operations, which are prevalent in most machine learning algorithms, to be applied to discrete objects. For example, one many take a dot product between any two word vectors to measure the similarity.

However, word embeddings are likely to suffer from corrupted samples. To be more concrete, most current algorithms used to embed words are heavily dependent on the corpora they use. The embedding algorithms work by taking co-occurrence statistics of words, and computes a set of vectors that best reflects such co-occurrences. Naturally, words that co-occur frequently will tend to have close-by vector representations. In other words, socio-linguistic norms are likely to influence the embeddings. While this approach may bear intuitive justification, it can sometimes have detrimental effects: Report by Bolukbasi et al. [2016] indicates that such word embeddings inadvertently exhibit sexist relationships.

Another shortcoming of word embedding is related to the processing of out-of-vocabulary (OOV) words. OOV words are words that have not been seen during the training phase of the embedding. Since OOV words lack sufficient amount of context on which embedding computation is based, there is not enough information to derive the embedding. Algorithms such as Skipthought (Kiros et al. [2015]) attempt to work around this issue by performing linear regression to extrapolate new vectors for the OOV word. Techniques motivated by morphological analysis exist as well (Zhang et al. [2015]), where convolution is performed on character level of the word to extract meaningful representation inherent in the morphemes. However, such methods are either approximate, or language dependent, and hence fail to capture the precise semantics of the embedding.

Finally, typical word embedding frameworks largely ignore the issue of word sense ambiguity. Most of the frequently-used words have multiple senses that are only discernible by their uses in context. Therefore, blindly using the corpora might yield embeddings that contain such unresolved ambiguities. There exists a work by Vilnis and McCallum [2015] that embeds each word as a multivariate Gaussian distribution, whose covariance matrix represents the degree of ambiguity. But it fails to mention how such ambiguities

might be resolved.
In this work, we propose to use a dictionary to improve the above-mentioned problems of traditional word embedding frameworks. Since a dictionary is considered to contain precise, authoritative description of each word's semantics, using it will alleviate problems associated with corrupt samples. We also introduce a method to simultaneously learn a sentence embedding while disambiguating plain words.

Related Works This section serves as a brief survey of previous efforts that are best related to our proposal. We begin with the dictionary embedding by Hill et al. [2016]. In this work, the authors use dictionary definitions to enhance previous embeddings. However the focus is mainly on improving phrase representation. In fact, they use pre-existing word embeddings to learn phrase representation. Also, they do not distinguish various senses of the plain words, which is the greatest difference from our work.
Sense embedding has been pursued by Iacobacci et al. [2015] as well. In this SensEmbed framework, the embedding of individual senses is computed in the same manner as skip-gram or continuous bag-of-word embeddings (Mikolov et al. [2013]) are computed. One important preprocessing is to run an external word sense diambiguation (WSD) algorithm to label the senses of interest in the plain-word corpora. However, this requires that the WSD algorithm be accurate enough to be of good use. In contrast, our work combines the process of WSD with the embedding computation.

1.1 CONTRIBUTIONS

Although there have been previous works such as those by Hill et al. [2016] and Iacobacci et al. [2015] that resemble ours, we propose the following contributions that differ from theirs.

- Leveraging small-sized corpus to learn embeddings: Many word embedding algorithms such as Pennington et al. [2014] and Mikolov et al. [2013] require massive amount of text data in order to successfully gather co-occurrence statistics. However, this approach becomes infeasible when we wish to deal with a minor language that lacks electronic corpora. Our method uses a dictionary, which is considered to be the minimal text required to learn the basics of a language. A dictionary is much smaller than the corpora required by other embedding algorithms, hence it is more economical to use one.

- Automatic sense-disambiguation: When embedding a sentence, one often encounters multi-sensed words with ambiguity, such as "bank" or "paper". Such words can be disambiguated via use of context, or surrounding words in the sentence. Our model is able to learn which sense of the given ambiguous word carries most weight in assigning a proper semantic to the sentence embedding.

- Out-of-vocabulary (OOV) word processing: One of the difficulties involved with previous word-embedding techniques is dealing with words that do not appear in the known vocabulary. Skip-gram, for example, require a large amount of text that contain the OOV word in question in order to learn the embedding. However, since our approach to embedding a word uses a dictionary, all we need is the definition sentence of the OOV word. Intuitively, this is the minimally-required information one uses to learn the meaning of a new word.

2 PRELIMINARY

In our setting, we are given a vocabulary $V = \{v_1, \cdots, v_N\}$ with N words, and each word v_i is paired with a definition string $d(v_i) = \{d_i^1, d_i^2, \cdots, d_i^k | d_i^j \in O\}$. We assume that the words comprising the definition sentences belongs to a different set O. All words $v_i \in V$ are *disambiguated*, and are expressed using the WordNet (Miller [1995]) format to denote various senses of a word. That is, each sense of a word w is expressed as: w.POS.sense_num. POS is one of 'n', 'v', 'a', 'r' for noun, verb, adjective, and adverb, and 'sense_num' is a natural number indicating the count of the word. For example, the fifth noun sense of the word 'bank' is written as `bank.n.5`. On the other hand, the words $d_i^j \in O$ that make up the definition are *ambiguous*. In the following, the terms "word" and "sense" refer to disambiguated words (*e.g.*, `bank.n.5`), and "plain word" to ambiguous words (*e.g.*, "bank").
The triple $(V, O, d())$ forms a *dictionary* in a traditional sense. This dictionary is *closed*, meaning that all plain words that appear in the definition string must also have their corresponding senses present in V.
The types of definition strings $d(v_i)$ need not necessarily be confined to dictionary definition, as long as the contents contain sufficient information to describe the target word v_i. It could be as large as Wikipedia articles, but we settle for WordNet definitions for this work in order to quickly demonstrate the feasibility. Our goal is to compute, or refine a vector representation $e_i \in \mathbb{R}^D$ for each $v_i \in V$ using only the dictionary - hence the name *Dictionary bootstrapping*.
The most intuitive way to compute the embedding e_i of word v_i is to assign the embedding of its definition $d(v_i)$. The embedding of $d(v_i)$ is acquired from a recurrent neural network (RNN: Hochreiter and Schmidhuber [1997]). However, we will see in Section 3.2 that such a sequential RNN can be replaced by a simpler RNN.

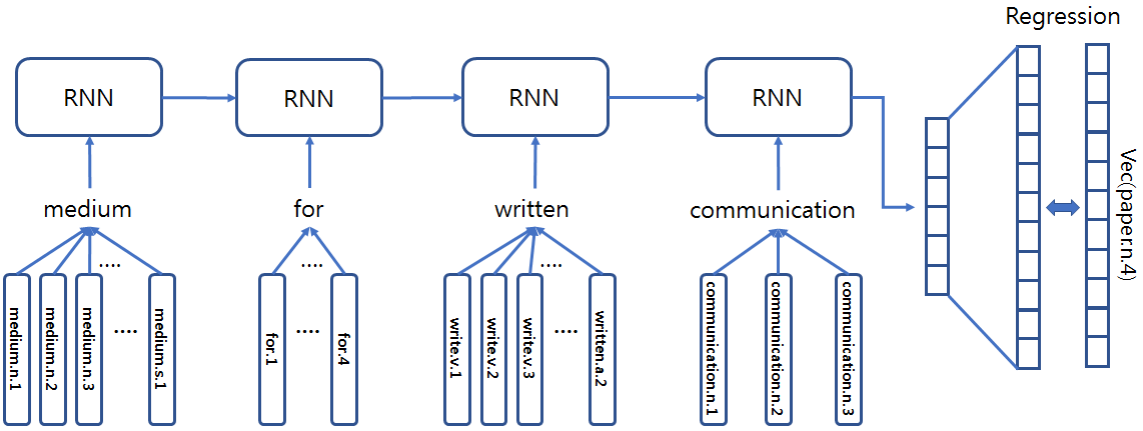

Figure 1: Proposed architecture for Dictionary Bootstrapping

3 ALGORITHM

One shortcoming of this sequeuntial RNN approach is that we only use the plain words of the definition sentence to derive an embedding. This might not pose much problem when we are dealing only with dictionaries, where the definitions of the words are given in a precise and unambiguous manner. But it would be better to devise a method that is capable of computing an embedding regardless of the ambiguity of the input sentence.

3.1 RESOLVING AMBIGUITIES

In order to reflect the ambiguity of the plain words, we let each plain word be modeled as a convex combination of its individual disambiguations, where the coefficients are determined by the context. For a plain word d_i, let $s(d_i) = \{v_i^1, \cdots, v_i^k\}$ denote the set of all disambiguated words of d_i. For example, $s(\text{'apple'}) = \{\texttt{apple.n.1}, \texttt{apple.n.2}\}$. With abuse of notation, we will also use $s(d_i)$ as the set of vectors corresponding to the disambiguated words. Then we represent a plain word d_i as

$$d_i = \sum_{v_j \in s(d_i)} \alpha_{ji} v_j, \text{ where } \sum_j \alpha_{ji} = 1, \alpha_{ji} \geq 0 \quad (1)$$

The coefficients α_{ji}'s denote the importance of each respective word v_j in relation to a plain word d_i in the context of v_j.

Let α_{ji} be the probability of the disambiguation for d_i being word $v_j \in s(d_i)$ given the context d_{-i} in the definition sentence[1]: $\alpha_{ji} \triangleq \Pr(v_j|d_{-i})$. We apply Bayes' rule to further expand this probability:

$$\alpha_{ji} = \frac{\Pr(v_j)\Pr(d_{-i}|v_j)}{\sum_{v_k \in s(d_i)} \Pr(v_k)\Pr(d_{-i}|v_k)} \quad (2)$$

[1]We have used the shorthand notation d_{-i} to mean the set $\{d_1, \cdots, d_{i-1}, d_{i+1}, \cdots, d_k\}$.

The prior $\Pr(v_j)$ allows the designer to incorporate pre-existing knowledge about the language of interest. For example, one might wish to reflect the fact that only the first few senses of a certain plain word are active majority of the time. This is achievable by assigning $\Pr(v_j) \propto \exp(-j)$ so later senses receive exponentially smaller probabilities. In case such an assumption does not hold, a uniform prior $\Pr(v_j) = |s(d_i)|^{-1}$ will suffice.

To compute the probability $\Pr(d_{-i}|v_j)$, we make the assumption that the probability of "observing" the plain words in d_{-i}'s in the context is conditionally independent given the disambiguated word v_j in question. That is,

$$\Pr(d_{-i}|v_j) = \prod_{d_m \in d_{-i}} \Pr(d_m|v_j). \quad (3)$$

Next, we specify the form of $\Pr(d_m|v_j)$ to complete Equation 3.

$$\Pr(d_m|v_j) \propto \exp\left(\frac{\tau}{|s(d_m)|} \sum_{v_k \in s(d_m)} v_k^T L v_j\right).$$

That is, the probability is defined as the average of similarities between v_j and the senses of d_m, parameterized by $L \in \mathbb{R}^{D \times D}$. To ensure we are computing a valid inner-product, we restrict L to be positive-definite. τ is a hyper-parameter that controls the "temperature" of the softmax distribution.

This way of modeling sense-disambiguation is motivated by how humans look up new words in the dictionary: When we encounter an ambiguous plain word w in a sentence S, we must select among w's various senses. The method we use to choose the sense is a simple comparison between the definitions of w's senses and the plain words in S. Our approach can be thought of as a simplification of this process where we use the sense embedding instead of directly using the definitions.

To train the parameters, we need a supervised signal. To this end, it is tempting to add a a classification layer at the

end of our RNN to predict the disambiguated word of the current definition sentence. However, training a classifier to predict the word corresponding to the current definition is ill-founded, due to insufficient data: For each class (*i.e.*, word), there is only a single data point (*i.e.*, definition embedding) to train. Instead, we opt to obtain the supervised signal via regression. For word $v_i \in V$ and its definition embedding e_i, let u_i be v_i's current embedding Then our objective is to learn a (possibly non-linear) function $f_\theta : \mathbb{R}^D \to \mathbb{R}^D$ that will take e_i to produce u_i. For our experiments, we take f_θ to be a two-layered neural network parameterized by layer weights $\Theta_0 = \{W_1, W_2\}$. Thus, our cost function becomes

$$J = \frac{1}{N} \sum_{i=1}^{N} \|f_{\Theta_0}(e_i) - u_i\|_2^2 + \lambda_0 \|\Theta_0\|_1, \qquad (4)$$

where we add an L1-regularization term to prevent overfitting.

Although we have dropped the idea of using a classification layer, we suggest that the classification layer might become useful in the presence of additional data. For example, if we were to use Wordnet resources freely, then we could reformulate the class structure that better reflects the semantic hierarchy of the words. Then we will have sufficient training data for each class for our classification layer.

Of course, regular sentences we normally encounter will not be definitions of any words in general. But we expect this algorithm to learn the correct semantics associated with the sentence we process by training on word-definition pairs.

3.2 MESSAGE-PASSING-BASED RNN

In theory, directly assigning the definition embedding e_i to v_i appears to be a sound way to represent v_i. This intuition naturally yields an iterative fixed-point iteration procedure $v_i \leftarrow e_i$ until convergence.

However, this process converges to bad embeddings empirically. This is because of the premature usage of v_i's by the RNN training procedure. That is, the RNN takes a convex combination of each of the senses per plain word of the sentence, but the embeddings of such senses have not yet converged. Furthermore, the random initialization of the senses seems to exacerbate the matter.

Here, we propose to use a simple form of a definition embedding that not only addresses this problem, but also reduces the number of parameters. Furthermore, it can potentially leverage good pre-existing embeddings to quickly stabilize the learning of disambiguated word embeddings. We will explain towards the end of this section, how this approach is a simplification of the typical RNN-based definition encoding.

The main idea is to approach the embedding task as simultaneously finding the clustering of the words and their coordinates. A typical approach to this problem is to use manifold learning algorithms such as multi-dimensional scaling (MDS: Kruskal [1964]) or locally-linear embedding (LLE: Roweis and Saul [2000]). However, such methods require pairwise distances, which we lack.

Instead, we construct an embedding model where the embedding of a word is influenced by the words in its definition sentence. In particular, we assume a linear influence model where the plain words in the definition are linearly combined to update the corresponding disambiguated word embedding. Our model takes doubly-convex combination of the words, where the first level combines the plain words according to some pre-computed weights, and the second level combines each disambiguated word of each plain word. The second level is achieved by the α_{ji}'s we train via Equation 4. This process is illustrated in Figure 3.2 for the word `paper.n.4`.

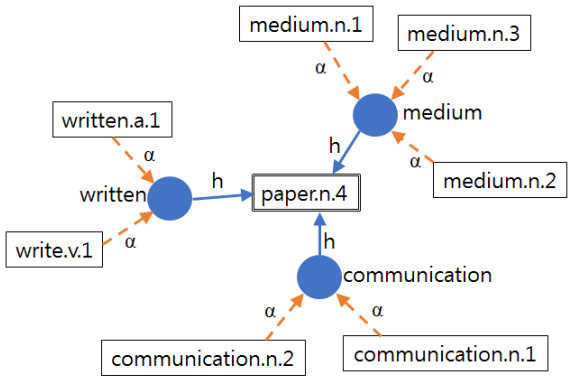

Figure 2: Illustration of passing influences.

Thus, the update formula takes the following recursive form for each word v_i:

$$v_i \leftarrow \sum_{d \in d(v_i)} h_d \sum_{v_j \in s(d)} \alpha_{ji} v_j. \qquad (5)$$

The plain word weights h_d are normalized for each disambiguated word v_i, and can be pre-computed as constants. In our work, we initialize them as the inverse document frequency (IDF) of the corresponding plain words, but any weighting scheme is allowed. Intuitively, words that are defined using similar words (*i.e.*, words with similar meaning) will have higher chance of receiving and passing influences among themselves, and hence result in closer embedding.

In the context of graphical model, our procedure is reminiscent of the belief propagation algorithm on the graph formed by all the words in the dictionary. The plain and disambiguated words consist the vertex set, where a plain word receives directed edges from its disambiguated words, and a disambiguated word receives directed edges from the plain words in its definition. The edges are weighted with either h_d or α_{ji} accordingly as in Figure 3.2. Under this interpretation, the updates to v_i are similar to the log-messages from the variable nodes, that are summed up.

Unfortunately, the graph we work with is not a tree. Therefore the convergence of the update is not guaranteed. We work around this issue by the following measure.

We introduce another parameter $\beta_t \in (0, 1)$ that changes over each iteration epoch t, to balance between stabilization and update:

$$v_i \leftarrow (1 - \beta_t)v_i + \beta_t \left(\sum_{d \in d(v_i)} h_d \sum_{v_j \in s(d)} \alpha_{ji} v_j \right). \quad (6)$$

High value of β_t will aggressively reflect the update coming from the message-passing encoder, while lower β_t will be more conservative and retain previous value of v_i. When training, we start with a high value of β_t and gradually decrease β_t over t. By decreasing β_t, we can now ensure convergence of our update procedure.

This update rule can be loosely thought of as an unnormalized version of random walk, where each word receives influences from the plain words that constitute its definition. We call it unnormalized because the outgoing influences do not add up to 1. Under this intuition, the random walker sends unnormalized influence to its neighbors. Then, words sharing similar words will receive roughly similar influences, thus leading an approximate clustering of the semantics.

Simplified RNN Finally, we point out that the message-passing-based update can be thought of as a simplified RNN computation in its own right. This RNN has an identity activation function, and operates by simply accumulating the inputs ignoring the order. Such a view has also been adopted by (Hill et al. [2016]) under the name of *Bag-of-words (BOW) neural language model*. Although it may appear to be a crude way to encode sentences, this method becomes effective when encoding short phrases. Since most definition "sentences" we deal in this work are actually phrase-level expressions, the use of BOW is justified.

3.3 OVERALL TRAINING

Our approach to word embedding involves two phases: a regressor training phase and a message-passing-based update phase. These two phases are intertwined in that the variables updated in one phase is used as constants in the other. Therefore, we take an alternating training scheme where we fix one set of the variables while training for the others.

When training the regressor (and other related parameters), we freeze the disambiguated word vectors v_i's and update the parameters $\Theta = \{L, \Theta_0\}$. We would ideally have to train the RNN until convergence, but we only run K rounds of stochastic gradient descent on L_1 in practice.

On the other hand, we spend more time on the message-passing-based updates, since we found that such fixed-point iteration converge relatively quickly. The iteration is stopped when the maximum difference between the previous value and the update is smaller than some small threshold ϵ. Notice that the updates are *in-place*, meaning that the new values are updated immediately rather than postponed for a batch update. This allows the recently-updated values to be used as soon as possible for quicker convergence. Our full training procedure is outlined in Algorithm 1.

Algorithm 1 Overall algorithm

Input: K, $\beta_0 \in (0, 1)$, learning rate γ, ϵ
For training epoch $t = 0, 1, \cdots$
repeat
  // Repeat the regressor training for K iterations
  for $i = 1, \cdots, K$ **do**
    $\Theta \leftarrow \Theta - \gamma \frac{\partial J}{\partial \Theta}$ (See Eqn 4 for J)
    Update α_{ji}'s using Θ (Eqn 2)
  end for
  // Run iteration for message-passing RNN
  repeat
    $\Delta \leftarrow -\infty$
    for Each word $v_i \in V$ **do**
      $v_{new} \leftarrow v_i + \beta_t(\sum h_d \sum \alpha_{ji} v_j - v_i)$ (Eqn 6)
      $\Delta \leftarrow \max\{|v_{new} - v_i|, \Delta\}$
      $v_i \leftarrow v_{new}$
    end for
  until $\Delta < \epsilon$
  $\beta_{t+1} \leftarrow \beta_t \beta_0$
until Converged

3.4 EXTENSION WITH BASE VECTORS

In the current presentation, we model the updates to vector v_i to be a function of purely its neighbors. In this section, we provide a generalized extension to our scheme, where each v_i can also take a pre-trained vector b_i into account when updating. This becomes possible by a simple modification to Equation 5:

$$v_i \leftarrow b_i + \sum_{d \in d(v_i)} h_d \sum_{v_j \in s(d)} \alpha_{ji} v_j. \quad (7)$$

The vectors b_i are called *base vectors*, since they provide base representations. The b_i's are supposed to be given as pre-trained vectors but we briefly discuss how to lift this assumption. We begin by treating each definition as a short document and perform topic modeling on them to compute topic distributions of each word. For example, latent Dirichlet allocation (LDA: (Blei et al. [2003])) with k topics yields a k-dimensional probability vector for each word. Using these topic vectors, we are able to compute the distance between any given two words using various distance metrics such as the Bhattacharyya distance, or the Jensen-Shannon distance. Such distances are gathered into an $N \times N$ matrix that are input to MDS. Performing MDS on this matrix results in the vector representations of each word that reflects the given distances. While these vectors

may not be the best representations, we expect them to be good-enough initial values for our approach. This approach works better when the definitions we have are of moderate length. If not, one should consider using topic models that are specialized to short texts (*e.g.*, Yan et al. [2013]).
In Section 3.5, we give a scenario where this initialization scheme becomes more useful. For the experiments, we do not use the base vectors when computing the embeddings.

3.5 APPLICATIONS

Next we discuss about potential applications of our approach. There exist several well-known solutions for some of the fields we address, but we stress that our structure is particularly well-suited for such problems.

WORD SENSE DISAMBIGUATION

Word sense disambiguation (WSD) is a task where one has to identify the correct sense of a plain word. An instance of this problem is usually given in a sentence where the plain word can be interpreted in a variety of ways for each of its senses. For example, the word `paper` in the sentence "He submitted a *paper* to the professor" would refer to the sense `paper.n.5`, whose meaning is "a scholarly article".
As such, disambiguating plain words involves comparing each of the senses to the surrounding words in the context. This is exactly how our model operates to assign weights on the senses. To perform WSD, we simply input the given sentence that contains the plain word w to disambiguate into our RNN model, and read off the computed α_{wj}'s.

OUT-OF-VOCABULARY EMBEDDING

One important shortcoming of all word embedding is the processing of out-of-vocabulary (OOV) words. Most word embedding algorithms heavily relies on co-occurrence patterns in order to compute embeddings. However, rarely occurring words or newly created words will likely have few or no context data to use to compute the embeddings.
Our model, on the other hand, is able to compute the embeddings of any new word as long as its definition is provided as well. There is one other restriction that states the definition should consist of known words only, but this natural to expect. In fact, a closed set of related definitions is the minimally required information to learn the meaning of a new word.
The process of embedding an unknown word is simple: Run the definition through the RNN to compute the α_{ji}'s and use them to perform a single step update (Equation 5) on the unknown word.

ENCYCLOPEDIA EMBEDDING

So far, we have worked with a type of dictionary with limited information. We have implicitly assumed that each definition is a sentence. However, there are much richer resources nowadays that we can use in place of a simple dictionary.
One prominent example is Wikipedia, which contains multi-sentence and multi-modal definitions of concepts.
Furthermore, the entries are hyperlinked, so our random walk framework naturally yields a local iterative algorithm. That is, in hyperlinked environments, Equation 7 can be thought as the Bellman equation (Bellman [2003]). The b_i would represent the contents of page i, while the h_d's and the α_{ji}'s merge to become the random walk probability of jumping from i to its outgoing neighbor j.

One factor to consider is the embedding of the entry document, which is significantly different from embedding a single sentence. One should employ multi-modal embedding techniques to form a summary vector of the document. Note that representing a document generally requires an object more complex than a single vector. However, we conjecture that sufficiently high-dimensional vector should work well for documents that are highly focused on a single concept.

4 EXPERIMENTS

To show the effect of sense embeddings, we perform both qualitative and quantitative experiments. The qualitative results will mainly show the relationship among the embedded words, and the quantitative results will aim on solving the word-sense disambiguation (WSD) task.
For the setting of our experiments, the only training corpus we use is the list of words and their definitions extracted from WordNet (Miller [1995]). We first select a small group of "seed words" from the English Wikipedia corpora, by choosing the 10,000 most commonly occuring plain words. Based on this set of seed words, we prune out words that are not in the WordNet, and traverse the WordNet definition tree over the current pool of plain words used in the definitions. Words (senses) that form the plain words newly-encountered along the traversal are also added into the vocabulary list. As the final refinement, we removed words that occur less than three times in other definitions. This last step filters out words like minor proper nouns and scholar jargons. The final dictionary collected this way consists of $|V| = 82,831$ words whose definitions are formed by $|O| = 38,730$ plain words.
One disadvantage of using WordNet is that the definitions are limited to words having one of four parts-of-speeches (POSs): noun, verb, adjective, and adverb. This means all stop-words including prepositions, conjunctions, and articles are left out of the main vocabulary. While it is crucial to hold on to the stop-words to correctly learn the semantics of a sentence, we ignore them in this work. The reason for doing so is twofold: 1) we would like to assess how well our algorithm can learn the embeddings in the face of limited vocabulary, and 2) full integration of stop-words into

the vocabulary might require sophisticated pre- and post-processing. We use no other linguistic/textual resources in this work.

As for the parameters of the model, we use the following setting.

- The trainable inner-product matrix L is factored into $L = UU$, where U is a diagonal matrix with non-zero elements. The optimization of Equation 4 is performed with respect to U instead of L.
- The decay schedule for β_t is computed as $\beta_t = 0.8^{t+1}$. Values of $\beta_0 = 0.7$ yielded similar results. $\Pr(d_m|v_j)$ being roughly uniform, due to exponentiating small values.
- The parameters Θ_0 and the word vectors $\{v_j\}$ are randomly initialized from the interval $(-0.1, 0.1)$. Word vectors are $D = 300$ dimensional.
- The per-plain-word coefficients $\{h_d\}$ for all $d \in O$ are computed as the IDF values over the dictionary.
- The priors $\Pr(v_i)$ are initialized as Dirichlet-distributed random variables, whose concentration parameter is set as $(4, 2, 1, \cdots, 1)$
- We perform $K = 5$ steps of stochastic gradient descent on Equation 4 with learning rate $\gamma = 0.05$, followed by iterative update of the word vectors. The iteration is stopped when the update size becomes smaller than $\epsilon = 10^{-3}$. The size of the mini-batches is set to 64.

4.1 QUALITATIVE RESULTS

The first experiment we perform is examining the nearest neighbors of some often-confused words when in their plain form. Table 1 shows the seven words we examine for their nearest neighbors.

Table 1: Definitions of words used in Table 2.

Word	Definition
hood.n.1	A violent young criminal.
hood.n.2	A protective covering part of a plant.
paper.n.1	A material made of cellulose pulp.
paper.n.5	A scholarly article.
bank.n.1	Sloping land.
bank.n.5	A stock held in reserve for future use.
apple.n.1	Fruit with red or yellow or green skin.

The measure used to retrieve the nearest neighbor is the cosine similarity: $sim(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}/\|\mathbf{x}\|\|\mathbf{y}\|$. Notice that the definitions are terse, and often seem to have insufficient information to learn the precise meaning. The list of nearest neighbors for seven selected words is given in Table 2.

Most of the retrieved words generally fall within the same category as the target word. There are two exceptions, however, that are underlined in the table.

The first one is `chiwere.n.1`, whose definition is "Language spoken by the Sioux people". We conjecture that this phenomenon is because of the flow of influence starting with `paper.n.1` → `card.n.1` → `michigan.n.3`[2] → `sioux.n.1` → `chiwere.n.1`. The reason of why the intermediate words did not appear in the nearest neighbors seems to be their definition lengths: longer definition tends to dampen the influence of the composing words in our BOW model. On the other hand, `chiwere.n.1` is relatively short and contains the plain word 'Sioux' which has an extremely high IDF. This, combined with the high IDF values of the intermediate words, boosted up the relatedness to `paper.n.1`.

The second exception is `hereafter.r.2`, whose definition is "in a future life or state". The influence from `bank.n.5` seems to have been passed via the term 'future'. Although its meaning is not entirely different from that of `bank.n.5`, it does seem a bit odd to be included in the neighbor list.

On a passing note, the nearest neighbors of `apple.n.1` are all valid, despite their deceiving forms. `cortland.n.1`, `delicious.n.1`, and `baldwin.n.3` are all types of apples.

The next set of experiments involves visualizing the embeddings in 2-dimension using t-SNE (van der Maaten and Hinton [2008]). First, we examine how analogical transformation of some words are reflected in their embeddings. The analogies we examine are liquid-solid and female-male relationships. For the former experiment, we take six words `drink.n.3`, `drink.v.1`, `magma.n.1`, `food.n.1`, `eat.v.1`, and `rock.n.1` and project their embeddings onto the 2-D plane. The first three words are related to liquid objects, and the last three are their solid counterparts. For example, `magma.n.1` is a liquid form of `rock.n.1`.

We can see from Figure 4.1 that the liquid versions are all generally located upward relative to the solid ones.

The second analogy is that of female-male relationship. The words used in this set are the gender-swapped version of the same concepts. Similarly to the previous embedding, we observe a somewhat consistent relative positioning in Figure 4.1.

Finally, we demonstrate that using an unprejudiced dictionary eliminates gender-biased embeddings such as those pointed out by Bolukbasi et al. [2016]. For this plotting, we compare the relative positions of four occupations `fireman.n.1`, `policeman.n.1`, `housewife.n.1`, and `seamstress.n.1`, compared to two anchor words

[2]A type of card game which has been mistaken as `michigan.n.1` when encoding `sioux.n.1`

Table 2: Nearest neighbors of select senses, with obvious "false neighbors" underlined.

Word	5 Nearest Neighbors
hood.n.1	bully.n.2, thugee.n.1, thuggery.n.1, muscleman.n.1, criminalism.n.1
hood.n.2	sheath.n.1, roof.n.1, skin.n.1, mask.n.4, roof.n.2
paper.n.1	cellulose.n.1, cellulosic.n.1, cellulose_xanthate.n.1, pulp.n.3, chiwere.n.1
paper.n.5	scholarly.a.1, unscholarly.a.1, tome.n.1, orientalism.n.1, memoir.n.2
bank.n.1	sloping.a.1, acclivitous.a.1, slop.v.3, hipped.a.1, highland.n.1
bank.n.5	store.n.2, forehanded.a.2, hereafter.r.2, stock.v.5, future.a.2
apple.n.1	cortland.n.1, winesap.n.1, delicious.n.1, corer.n.1, baldwin.n.3

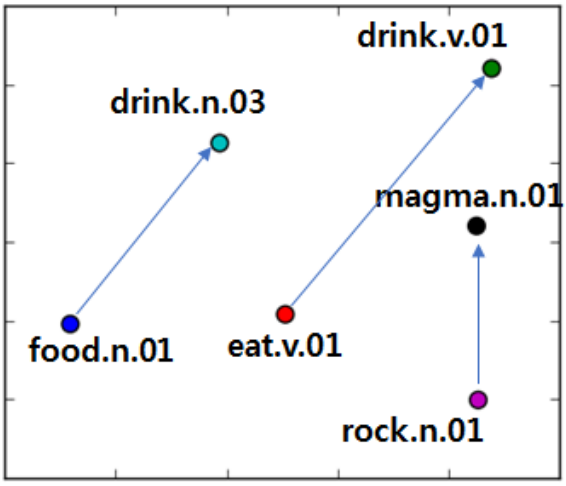

Figure 3: Liquid-solid relationship.

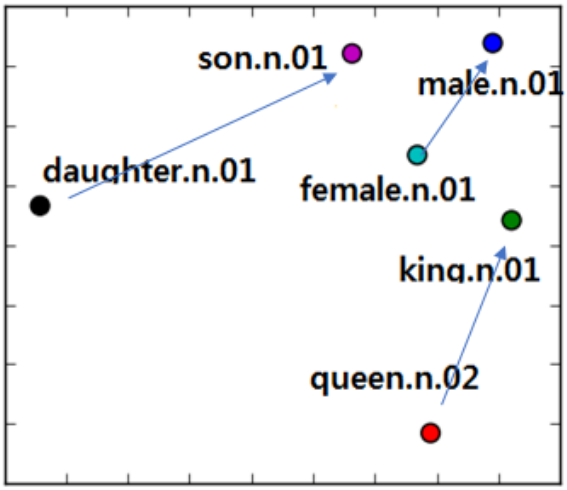

Figure 4: Gender relationship.

man.n.1 and woman.n.1. The first two occupations have been historically regarded as masculine jobs, while the last two were thought as feminine jobs (as shown etymologically). However, Figure 4.1 shows that our dictionary-based embeddings ignore such stereotypes; the positions of the occupations do not exihibit any patterns relative to the embeddings of man.n.1 and woman.n.1. This is not surprising, since the definitions for the four occupations used contain almost no gender-specific words, with the exception of housewife.n.1, whose definition is "A wife who manages a household while her husband earns the family income".

4.2 WORD SENSE DISAMBIGUATION RESULTS

Finally, we report result on the word sense disambiguation (WSD) task (Jurgens and Klapaftis [2013]). The evaluation set for this task had been provided by the SemEval2013 workshop, and we compare the performance of our approach to those of four other participants of the then-held competition. In addition to the participants, we also ran five baseline algorithms that rely on traditional WSD algorithms. These include: 1) the Lesk algorithm (Lesk [1986]), the adaptive Lesk (Banerjee and Pedersen [2002]), cosine Lesk (which uses cosines to compute overlaps), path similarity-based disambiguation (Leacock and Chodorow [1998]), and information-theoretic approach (Lin [1998]). Of these, we only report the result for the adaptive Lesk since it outperformed the other four[3].

The WSD task we address asks us to identify the correct sense of a particular word in context. The input is a sentence with a designated plain word in the sentence to disambiguate. The output should either be a single identified sense or multiple senses each labeled with applicability weights. SemEval provides five different evaluation metrics, but we only show three; two from the F1 measure family and one from the clustering measure family. The first is the well-known Jaccard index and the second is the weighted NDCG metric that computes a weighted F1 score. The third is the fuzzy NMI metric which views the weighted sense identification as a fuzzy clustering. We direct the readers to the cited paper for more information.

When identifying the sense, our algorithm simply computes the α_{ji} coefficients via Equation 2 and the trained sense embeddings. Then we remove the senses receiving

[3]The baseline algorithms are collectively implemented by the PyWSD package (https://github.com/alvations/pywsd)

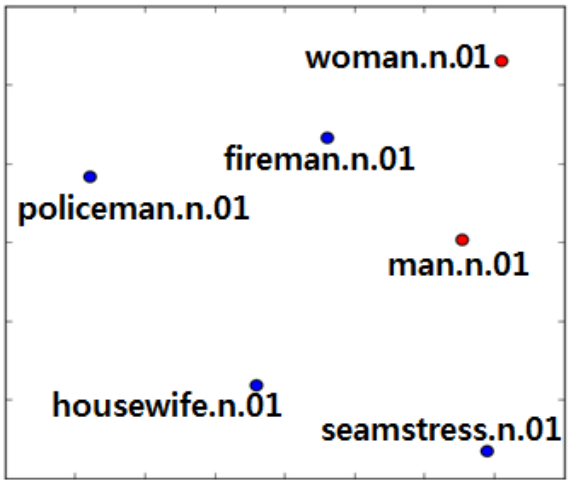

Figure 5: Gender-neutral embedding.

α_{ji} weights below a threshold of 0.02. For the Jaccard measure, we select the sense with the highest weight, since the Jaccard index does not account for the weights. Other than a straightforward thresholding, we perform no extra operations to fine-tune the final outcome. The result of our system is given in Table 3.

Despite the high Fuzzy NMI score, the Jaccard index and the WNDCG score do not stand out. But keep in mind that we only use a relatively small-sized corpus for an unsupervised training, and perform no post-processing on the WSD output. The competing algorithms, on the other hand, make heavy use of large corpora such as ukWaC (Ferraresi et al. [2008]) and Wikipedia, to train systems specifically targeted for the WSD task. Table 4 contrasts the amount of resources used by the competitors against ours. WN-sim is a WordNet-based similarity measure that reflects the entire ontological tree of WordNet. While there is room for improvement, we claim that our algorithm provides a good starting-ground for NLP applications.

Table 3: Results of the SemEval2013 WSD task. Duplicate entries indicate multiple submissions.

Algorithm	Jaccard	WNDCG	FuzzyNMI
AI-KU1	0.197	**0.387**	0.065
AI-KU2	0.197	0.215	0.035
AI-KU3	0.244	0.332	0.039
Unimelb1	0.218	0.365	0.056
Unimelb2	0.213	0.371	0.060
UoS1	0.192	0.315	0.047
UoS2	0.232	0.374	0.045
La Sapienza1	0.149	0.311	-
La Sapienza2	0.149	0.383	-
ALesk (baseline)	**0.254**	0.293	0.041
DiBoots (ours)	0.209	0.370	**0.074**

Table 4: Comparison of resource requirements. WN-sim and WN-dict refer to the WordNet similarity functions and dictionary, respectively. The corpora ukWaC and Wikipedia each consist of 2 billion and 800 million English words.

System	Corpora	External Algorithms
AI-KU	ukWaC	Fastsub, S-CODE, K-means
Unimelb	Wikipedia	Hierarichical Dir. Proc.
UoS	ukWaC	Depend. parser, MaxMax
La Sapienza	ukWaC	PageRank, WN-sim.
DiBoots	WN-dict	-

5 DISCUSSION

Our algorithm does provide promising aspects in the field of word embedding. While we do not achieve state-of-the art results in the problems we address, the performance is more than reasonable given the 6.9 MB dictionary size. Furthermore, we do not process the text in any way, other than simple case-lowering.

One interesting direction of improvement lies with language pre-processing: We have ignored the POS tag, tense, numbers, and compound nouns in this work. But such are the important factors for properly encoding the semantics. For example, the term “rocky mountain” should be treated as a single compound noun instead of two as we do. Also, as mentioned above, the restriction of WordNet definitions to four major POS categories has excluded the processing of many important conjunctions and prepositions such as “and” and “for”. In order to better-reflect the meaning of the definition sentence, we must devise a method to bring in these words.

Another interesting direction is with combining existing embeddings, motivated by Table 2. The nearest neighbors of the word `apple.n.1` are all valid apples, but are unfamiliar to the general public. Other embeddings such as GloVe will retrieve more user-friendly words, since they are trained to recognize naturally co-occurring word pairs such as apple-banana or apple-fruit. While the nearest neighbors DiBoots retrieves might be “more correct” than words such as `banana.n.1`, they might be inadequate for non-professionals. After all, acquiring precise semantics from the dictionary is often not what people expect. In that respect, using our algorithm to fine-tune existing co-occurrence-based embeddings seems like a desirable combination.

Acknowledgement B.Kang and K.-A. Sohn are supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education [NRF-2016R1A6A3A11932796 and NRF-2016R1D1A1B03933875, respectively.]

References

S. Banerjee and T. Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of CICLing*, 2002.

R. Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003. ISBN 0486428095.

D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

T. Bolukbasi, K. Chang, J. Zou, V. Saligrama, and A. Kalai. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Proceedings of NIPS*, 2016.

A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini. Introducing and evaluating ukWaC, a very large web-derived corpus of english. In *Proceedings of the 4-th Web as Corpus (WAC) Workshop*, 2008.

F. Hill, K. Cho, A. Korhonen, and Y. Bengio. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9, 1997.

I. Iacobacci, M. T. Pilehvar, and R. Navigli. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of IJCNLP*, 2015.

D. Jurgens and I. Klapaftis. SemEval-2013 Task 13: Word sense induction for graded- and non-graded senses. In *Proceedings of International Workshop on Semantic Evaluation*, 2013.

R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. In *Proceedings of NIPS*, 2015.

J. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29, 1964.

C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. *Computational Linguistics*, 24, 1998.

M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC*, 1986.

D. Lin. An information-theoretic definition of similarity. In *Proceedings of ICML*, 1998.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

G. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38, 1995.

J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.

L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9, 2008.

L. Vilnis and A. McCallum. Word representation via gaussian embedding. In *Proceedings of ICLR*, 2015.

X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *Proceedings of WWW*, 2013.

X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Proceedings of NIPS*, 2015.