# Discrete Word Embedding for Logical Natural Language Understanding: Extended Abstract

**Submission 9**

**Abstract**

In this paper, we propose an unsupervised method for learning a discrete embedding of words. While being discrete, our embedding supports vector arithmetic operations similar to continuous embeddings by interpreting each word as a set of propositional statements describing a rule. The formulation of our vector arithmetic closely reflects the logical structure originating from the symbolic sequential decision making formalism (classical/STRIPS planning). Contrary to the conventional wisdom that discrete representation cannot perform well due to the lack of ability to capture the uncertainty, our representation is competitive against the continuous representations in several downstream tasks. Finally, we demonstrate that our embedding is directly compatible with the symbolic, classical planning solvers by performing a "paraphrasing" task.

## 1 Introduction

After the initial success of the distributed word representation in Word2Vec (Mikolov et al. 2013b), natural language processing techniques have achieved tremendous progress in the last decade, propelled primarily by the advancement in data-driven machine learning approaches based on neural networks. Meanwhile, these data-driven approach could suffer from biased decision making and the lack of interpretability/explainability (Caliskan, Bryson, and Narayanan 2017; Bolukbasi et al. 2016).

In recent years, significant progress has been made (Asai and Fukunaga 2018; Kurutach et al. 2018; Amado et al. 2018; Asai and Muise 2020) in the field of Automated Planning on resolving the so-called *Knowledge Acquisition Bottleneck* (Cullen and Bryman 1988), the common cost of human involvement in converting real-world problems into the inputs for symbolic AI systems. Given a set of noisy visual transitions in fully observable puzzle environments, the above-mentioned systems can extract a set of anonymous propositional, predicate, or action symbols entirely without human supervision. Each action symbol maps to a description of the propositional transition rule in STRIPS classical planning (Fikes, Hart, and Nilsson 1972; Haslum et al. 2019) formalism that can be directly fed to the optimized implementations of the off-the-shelf state-of-the-art classical planning solvers. Due to the logical correctness of the graph-theoretic analysis performed by the symbolic systems, results are guaranteed to be correct, and sometimes also guaranteed to be optimal, depending on the setting.



In this paper, we point out some weaknesses of continuous distributed word embeddings and address them by proposing a discrete word embedding operated by set-based bit-vector arithmetic. Unlike existing work on discrete embeddings (Chen, Min, and Sun 2018), our embedding can perform semantic tasks directly in the discrete representation. We evaluate our approach in several downstream natural language tasks, including word similarity, analogy, and text classification. Furthermore, our discrete embedding has a unique feature that it is directly compatible with state-of-the-art symbolic methods. We demonstrate its ability to perform a "paraphrasing" task using classical planners, where the task is to logically compose the words based on cause-effect structure in the discrete embedding.

## 2 Continuous embedding considered harmful

Common downstream tasks in modern natural language processing with word embedding involve arithmetic vector operations that aggregate the embedding vectors. Analogy task (Mikolov, Yih, and Zweig 2013) is one such task that requires a sequence of arithmetic manipulations over the embeddings. Given two pairs of words *"a is to a* as b is to b*"*, the famous example being *"man is to king as woman is to queen"*, the model predicts $b^*$ by manipulating the embedded vectors of the first three words. The stan-

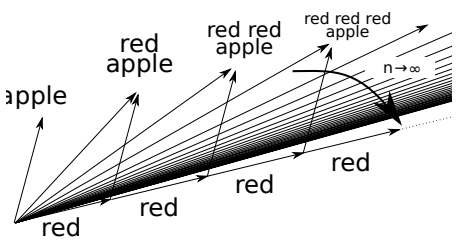


Figure 1: The shortcoming of adding continuous vectors in a cosine vector space.

dard method for obtaining such a prediction is 3COSADD (Mikolov, Yih, and Zweig 2013), which attempts to find the closest word embedding to a vector $\boldsymbol{a}^* - \boldsymbol{a} + \boldsymbol{b}$ measured by the cosine distance $\cos(\boldsymbol{v}_1, \boldsymbol{v}_2) = 1 - \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{|\boldsymbol{v}_1||\boldsymbol{v}_2|}$, assuming that the result is close to the target embedding $\boldsymbol{b}^*$. This, along with other analogy calculation methods (e.g., (Levy and Goldberg 2014)), uses simple vector arithmetic to obtain the result embedding used to predict the target word. In addition, text classification evaluation methods often build classifiers based on the mean or the sum of the word vectors in a sentence or a document (Tsvetkov et al. 2015; Yogatama and Smith 2014).

One shortcoming of these vector operations is that the resulting embedding is easily affected by the syntactic and semantic redundancy. These redundancies should ideally carry no effect on logical understanding, and at most with diminishing effect when repetition is used for subjective emphasis. Consider the phrase "red red apple". While the first "red" has the effect of specifying the color of the apple, the second "red" is logically redundant in the syntactic level. Phrases may also contain semantic redundancy, such as "free gift" and "regular habit". However, in a continuous word embedding, simple summation or averaging would push the result vector toward the repeated words or meanings. That is, for any non-zero vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\cos(\boldsymbol{a} \cdot n + \boldsymbol{b}, \boldsymbol{a}) \to 0, (n \to \infty)$ (Fig. 1). Even with a more sophisticated aggregation method for a vector sequence, such as the recurrent neural networks (Hochreiter and Schmidhuber 1997), the problem still remains as long as it is based on a continuous representation.

This behavior is problematic in critical applications which require logical soundness. For example, one may attempt to fool the automated topic extraction or auditing system by repeatedly adding a certain phrase to a document in an invisible font (e.g., transparent) as a form of adversarial attack (Jia and Liang 2017). This issue is also related to the fact that word2vec embedding encodes important information in its magnitude (Schakel and Wilson 2015; Wilson and Schakel 2015). While Xing et al. (2015) proposed a method to train a vector embedding constrained to a unit sphere, the issue caused by the continuous operations still remains.

On the other hand, symbolic natural language methods rely on logical structures to extract and process information. For example, Abstract Meaning Representation (AMR) (Banarescu et al. 2013) encodes a natural language sentence into a tree-structured representation with which a logical query can be performed. However, while there are systems that try to extract AMR from natural language corpora (Flanigan et al. 2014; Wang, Xue, and Pradhan 2015), these approaches rely on annotated data and hand-crafted symbols such as `want-01` or `c / city`. In addition to the annotation cost, these symbols are opaque and lack the internal structure which allows semantic information to be queried and logically analyzed. For example, a node `city` does not by itself carry information that it is inhabited by the local people and is a larger version of a `town`. In contrast, a Word2Vec embedding may encode such information in its own continuous vector.

To address the issues in both paradigms, we propose a binary discrete embedding scheme that stands upon propositional logic (like AMR) while supporting vector arithmetic (like continuous embedding). To achieve this goal, we combine the existing discrete variational method with CBOW Word2Vec and obtain an atomic propositional representation of the words.

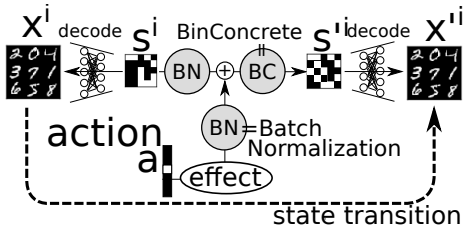


Figure 2: Cube-Space AE (Asai and Muise 2020) for a visual time series $(\boldsymbol{x}^i, \boldsymbol{x}'^i)$ that applies Back-to-Logit to the binary representation $(\boldsymbol{s}^i, \boldsymbol{s}'^i)$ and the action vector $\boldsymbol{a}^i$.

## 3 Preliminary and background

We denote a multi-dimensional array in bold and its subarrays with a subscript (e.g., $\boldsymbol{x} \in \mathbb{R}^{N \times M}$, $\boldsymbol{x}_2 \in \mathbb{R}^M$), an integer range $n < i < m$ by $n..m$, and the $i$-th data point of a dataset by a superscript $^i$ which we may omit for clarity. Functions (e.g., $\log, \exp$) are applied to the arrays elementwise.

The Word2Vec **Continuous Bag of Word** (CBOW) with Negative Sampling (Mikolov et al. 2013a; 2013b) language model is a shallow neural network that predicts a specific center word of a $2c+1$-gram from the rest of the words (context words). The model consists of two embedding matrices $W, W' \in \mathbb{R}^{V \times E}$ where $V$ is the size of the vocabulary and $E$ is the size of the embedding. For a $2c+1$-gram $\left\langle x^{i-c}, \ldots, x^{i+c} \right\rangle$ $(x^i \in 1..V)$ in a dataset $\mathcal{X} = \left\{x^i\right\}$, it computes the continuous-bag-of-words representation $\boldsymbol{e}^i = \sum_{-c \le j \le c, j \neq 0} W_{x^{i+j}}$. While it is possible to map this vector to the probabilities over $V$ vocabulary words with a linear layer, it is computationally expensive due to the large constant $V$. To avoid this problem, Negative Sampling maps the target word $x^i$ to an embedding $W'_{x^i}$, performs a random sampling over $V$ to select $K$ words $(r^k, k \in 1..K)$, extracts their embeddings $W'_{r^k}$, then maximizes the loss: $\log \sigma(\boldsymbol{e}^i \cdot W'_{x^i}) + \sum_{k=1}^{K} \log \sigma(-\boldsymbol{e}^i \cdot W'_{r^k})$.

**Variational AutoEncoder** (VAE) is a framework for reconstructing the observation $x$ from a compact latent representation $z$ that follows a certain prior distribution, which is often a Normal distribution $\mathcal{N}(0, 1)$ for a continuous $z$. Training is performed by maximizing the sum of the reconstruction loss and the KL divergence between the latent random distribution $q(z|x)$ and the target distribution $p(z) = \mathcal{N}(0, 1)$, which gives a lower bound for the likelihood $p(x)$ (Kingma and Welling 2013). **Gumbel-Softmax** (GS) VAE (Jang, Gu, and Poole 2017) and its binary special case **Binary Concrete** (BC) VAE (Maddison, Mnih, and Teh 2017) instead use a discrete, uniform categorical dis-

tribution as the target distribution, and further approximate it with a continuous relaxation by annealing the controlling parameter (temperature $\tau$) down to 0. The latent value $z$ of Binary Concrete VAE is activated from an input logit $x$ by $z = \mathrm{BC}(x) = \textsc{Sigmoid}((x + \textsc{Logistic}(0,1))/\tau)$, where $\textsc{Logistic}(0,1) = \log u - \log(1-u)$ and $u \in [0,1]$ is sampled from $\textsc{Uniform}(0,1)$. BC converges to the Heaviside step function at the limit $\tau \to 0$: $\mathrm{BC}(x) \to \textsc{Step}(x)$ (step function thresholded at 0).

A grounded (propositional) unit-cost **STRIPS Planning problem** (Fikes, Hart, and Nilsson 1972; Haslum et al. 2019) is defined as a 4-tuple $\langle P, A, I, G\rangle$ where $P$ is a finite set of propositions, $A$ is a finite set of actions, $I \subseteq P$ is an initial state, and $G \subseteq P$ is a goal condition. Each action $a \in A$ is a 3-tuple $\langle \textsc{Pre}(a), \textsc{Add}(a), \textsc{Del}(a)\rangle$ where $\textsc{Pre}(a), \textsc{Add}(a), \textsc{Del}(a) \subseteq P$ are preconditions, add-effects, and delete-effects, respectively. $\textsc{Add}(a) \cap \textsc{Del}(a) = \emptyset$. A *state* $s \subseteq P$ is a set of propositions which are considered to hold true in $s$. An action $a$ is *applicable* when $s$ *satisfies* $\textsc{Pre}(a)$, i.e., $\textsc{Pre}(a) \subseteq s$. *Applying* an action $a$ to $s$ yields a new successor state $s' = a(s) = (s \setminus \textsc{Del}(a)) \cup \textsc{Add}(a)$. The solution to a classical planning problem is called a *plan*, which is a sequence of actions $\pi = \langle a_1, a_2, \ldots a_{|\pi|}\rangle$ that leads to a terminal state $s^* = a_{|\pi|} \circ \ldots \circ a_1(s)$ that satisfies the goal condition, i.e., $G \subseteq s^*$. *Optimal* plans are those whose lengths are the smallest among possible plans. Each state $s \subseteq P$ can be encoded as a bit vector $\boldsymbol{s} \in \{0,1\}^{|P|}$ where, for each $j$-th proposition $p_j \in P$, $\boldsymbol{s}_j = 1$ when $p_j \in s$, and $\boldsymbol{s}_j = 0$ when $p_j \notin s$. A state transition graph of a classical planning problem is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ generated by $I \in \mathcal{V}$ and $A$. The nodes $\mathcal{V}$ are the states and the edges $\mathcal{E}$ are labeled by the actions, i.e., for any edge $(s, s') \in \mathcal{E}$, $s' = a(s)$ for some $a \in A$.

In the context of modeling the time series data, **cube-space prior** (Asai and Muise 2020) is a structural prior for a discrete latent space that restricts the state transition graph to be an instance of directed *cube-like graphs* (Payan 1992). The state transition graph of any STRIPS planning problem is an instance of a directed cube-like graph. Therefore, combining Binary Concrete variational method with this prior, a neural network is able to encode raw inputs (time-series data) into a state and an action representation compatible with STRIPS planning systems.

Finally, **Back-to-Logit** (BTL) technique (Asai and Muise 2020) implements this prior in the continuous relaxation of the binary latent space during the training (Fig. 2). To mitigate the issue of applying a prior to a discrete representation that is known to be difficult to train by itself, BTL avoids directly operating on the discrete vectors. Instead, it converts them to continuous vectors using Batch Normalization (BN) (Ioffe and Szegedy 2015), takes the continuous sum with the *effect* vector of an action coming from an additional network, and puts the resulting logit back to the discrete space using Binary Concrete. Formally, given an action representation $a$, an effect prediction network $\textsc{Effect}(a)$, the current state and the successor state binary latent vector $\boldsymbol{s}^i$ and $\boldsymbol{s}'^i$

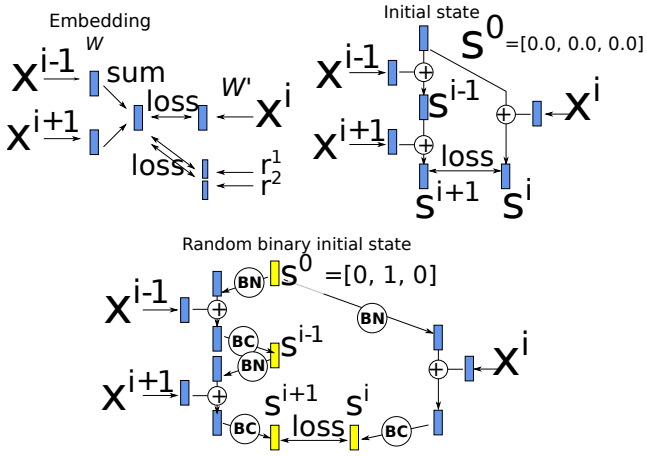


Figure 3: (Top Left) Traditional 3-gram CBOW with negative sampling. (Top Right) 3-gram CBOW seen as a sequence of continuous state manipulations. (Negative sampling is not shown) (Bottom) 3-gram Discrete Sequential Application of Words model. BN=Batch Normalization, BC=Binary Concrete.

($i$: index in a dataset), the successor state is predicted by:

$$\boldsymbol{s}'^i \approx \textsc{Apply}(a, \boldsymbol{s}^i) = \mathrm{BC}(\mathrm{BN}(\boldsymbol{s}^i) + \textsc{Effect}(a)).$$

The state representation $\boldsymbol{s}$ trained with BTL has the following properties:

**Theorem 1** ((Asai and Muise 2020)). *Under the same action $a$, the state transitions are monotonic and deterministic:*

$$\begin{aligned}
(\textit{add effect:})\ \exists i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) &= (0,1) \\
\Rightarrow \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) &= (0,1)\ \textit{or}\ (1,1), \\
(\textit{delete effect:})\ \exists i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) &= (1,0) \\
\Rightarrow \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) &= (1,0)\ \textit{or}\ (0,0)\ (\textit{for each } j).
\end{aligned}$$

This theorem guarantees that each action deterministically turns a certain bit on and off in the binary latent space, thus the resulting action theory and the bit-vector representation satisfies the STRIPS state transition rule $s' = (s \setminus \textsc{Del}(a)) \cup \textsc{Add}(a)$ and a constraint $\textsc{Del}(a) \cap \textsc{Add}(a) = \emptyset$. Latplan system (Asai and Fukunaga 2018; Asai and Muise 2020) uses this framework to implement Cube-Space AE, an autoencoder that learns to encode a noisy visual time-series data into a STRIPS planning model with unsupervised learning.

## 4 Model architecture

To introduce the model, we modify the CBOW Word2Vec (Fig. 3, left) in two steps. We first identify that CBOW can be seen as a simple constant recurrent model (Fig. 3, middle). This trivial "recurrent" model merely adds the input embedding to the current state. Unlike the more complex, practical RNNs, such as LSTM (Hochreiter and Schmidhuber 1997) or GRU (Cho et al. 2014), this model lacks any form of weights or nonlinearity that transforms the current state to the next state.

This interpretation of CBOW yields several insights: First, there is a concept of "initial states" $s^0$, like any other recurrent model, that are inherited by the surrounding context outside the ngram and manipulated by the effects $W_{x^i}$ into the output state $\boldsymbol{s}^{i+c} = \boldsymbol{s}^0 + \sum_{-c \le j \le c, j \neq 0} W_{x^{i+j}}$. Coincidentally, this output state is merely the sum of the effect vectors if $s^0$ is a zero vector, resulting in the equivalent formulation as the original CBOW. This also helps us understand the optimization objective behind CBOW: The *effect* of the target word closely resembles the accumulated *effect* of the context.

Second, upon discretizing some of the elements in this model in the next step, we should preserve the fundamental ability of CBOW to *add(+), remove(-) or keep(0)* the value of each dimension of the state vector. It is important to realize that a simple binary or categorical word embedding, such as the work done by Chen, Min, and Sun (2018) (for a significantly different purpose), is incompatible with the concept of *adding*, *removing* or *keeping*. Notice that unlike continuous values, categorical values lack the inherent ordering (total or partial). Therefore, categorical values are not able to define *adding* and *removing* as the inverse operations, as well as *keeping* as an identity.

### 4.1 Discrete Sequential Application of Words

Based on the observations above, we propose Discrete Sequential Application of Words (DSAW, Fig. 3, right), which addresses the issues in continuous embeddings, naive discrete models, or hand-crafted symbolic models by using *two* binary vectors to represent each word.

DSAW sequentially applies the Back-to-Logit technique to a random initial state vector sampled from the Bernoulli(0.5) distribution. Each recurrent state $\boldsymbol{s}^t$ is a continuous relaxation of a binary vector modeled by Binary Concrete. The embedding matrix $W$ itself is not discrete. However, due to Theorem 1, we can extract *two* binary vectors $\textsc{add}(x)$, $\textsc{del}(x)$ of a word $x$ that satisfy $\boldsymbol{s}' = (\boldsymbol{s}\ \&\&\ !\textsc{del}(x)) \mid\mid \textsc{add}(x)$, which is a bit-vector implementation of set-based STRIPS action application $s' = (s \setminus \textsc{del}(a)) \cap \textsc{add}(a)$.

Since state vectors are activated by Binary Concrete, which behaves like a Sigmoid function in high temperature and as a step function in low temperature, all state vectors reside in the unit hypercube $[0,1]^E$. This means that we cannot directly apply the traditional objective function $\log \sigma(\boldsymbol{x} \cdot \boldsymbol{y})$ in Word2Vec to the output state vector because it assumes that the distribution of $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^E$ is centered around the origin, while our discrete output states are heavily biased toward the positive orthant. To address this issue, we shift the mean by subtracting 0.5 from the output vector before computing the loss. Formally, our maximization objective (including negative sampling with $\{r^1, \dots r^K\}$) is defined as follows, where $\boldsymbol{s}^i = \textsc{apply}(x^i, \boldsymbol{s}^0)$, $\boldsymbol{s}^{i-c} = \textsc{apply}(x^{i-c}, \boldsymbol{s}^0)$, $\boldsymbol{s}^{i+1} = \textsc{apply}(x^{i+1}, \boldsymbol{s}^{i-1})$, $\boldsymbol{s}^{i+j} = \textsc{apply}(x^{i+j}, \boldsymbol{s}^{i+j-1})(j \notin \{-c, 0, 1\})$:

$$\log \sigma((\boldsymbol{s}^{i+c} - 0.5) \cdot (\boldsymbol{s}^i - 0.5)) + \sum_{k=1}^{K} \log \sigma(-(\boldsymbol{s}^{i+c} - 0.5) \cdot (\textsc{apply}(r^k, \boldsymbol{s}^0) - 0.5)). \quad (1)$$

Once the training has been completed, we compute one forward recurrent step for each word $x$ with two initial state vectors $\mathbf{0}, \mathbf{1}$ each consisting of all 0s and all 1s. We can then determine the effect in each dimension $j$: $\textsc{add}(x)_j = 1$ if $\textsc{apply}(x, \mathbf{0})_j = 1$, and $\textsc{del}(x)_j = 1$ if $\textsc{apply}(x, \mathbf{1})_j = 0$.

### 4.2 Inference in the discrete space

An important question for any discrete models is how to perform arithmetic operations with the discrete representation. Specifically, to perform the word analogy task (Mikolov et al. 2013b), the representation must support both *addition and subtraction* of words, which is non-trivial for discrete vectors.

We propose to use the STRIPS *progression* and *regression* (Alcázar et al. 2013; Haslum et al. 2019) (also called *forward reasoning* and *backward reasoning*) as the vector addition and subtraction operation for our binary word embedding. Recall that, in the continuous effect model, vector subtraction is equivalent to *undoing* the effect of the action that has been performed. Similarly, a STRIPS regression [1] restores the original state of a STRIPS progression $s' = (s \setminus \textsc{del}(a)) \cap \textsc{add}(a)$ by $s = (s' \setminus \textsc{add}(a)) \cap \textsc{del}(a)$. We denote these operations as $s \mathbin{\hat{+}} x$ and $s \mathbin{\hat{-}} x$. We note that our operation is not associative or commutative. That is, the result of "king-man+woman" may be different from "king+woman-man" etc.

Next, for a sequence of operations $sR_1x_1 \dots R_nx_n (R_i \in \{\hat{+}, \hat{-}\})$, we denote its combined effects as $e = R_1x_1 \dots R_nx_n$. Its add/delete-effects, $\textsc{add}(e), \textsc{del}(e)$, are recursively defined as follows:

$$\begin{aligned}
\textsc{add}(e \mathbin{\hat{+}} x) &= \textsc{add}(e) \setminus \textsc{del}(x) \cup \textsc{add}(x) \\
\textsc{del}(e \mathbin{\hat{+}} x) &= \textsc{del}(e) \setminus \textsc{add}(x) \cup \textsc{del}(x) \\
\textsc{add}(e \mathbin{\hat{-}} x) &= \textsc{add}(e) \setminus \textsc{add}(x) \cup \textsc{del}(x) \\
\textsc{del}(e \mathbin{\hat{-}} x) &= \textsc{del}(e) \setminus \textsc{del}(x) \cup \textsc{add}(x)
\end{aligned}$$

In the following artificial examples, we illustrate that (1) our set-based arithmetic is able to replicate the behavior of the classic word analogy *"man is to king as woman is to queen"*, and (2) our set-based operation is robust against semantic redundancy.

**Example 1.** *Assume a 2-dimensional word embedding, where each dimension is assigned a meaning [female, status]. Assume each word has the effects as shown in Table 1.*

[1] Here we assume that the effect always invoke changes to the state in order to obtain a deterministic outcome from the regression. Normally, the regression is nondeterministic unless warranted by the preconditions, e.g., if $p_1 \in \textsc{pre}(a) \wedge p_1 \in \textsc{del}(a)$, then $p_1$ is guaranteed to be true before applying the action $a$.

| word $x$ | DEL$(x)$ | ADD$(x)$ |
|---|---|---|
| King | $[1,0] = \{\text{female}\}$ | $[0,1] = \{\text{status}\}$ |
| Man | $[1,0] = \{\text{female}\}$ | $[0,0] = \emptyset$ |
| Woman | $[0,0] = \emptyset$ | $[1,0] = \{\text{female}\}$ |
| Queen | $[0,0] = \emptyset$ | $[1,1] = \{\text{female}, \text{status}\}$ |

Table 1: An example 2-dimensional embedding.

*Then the effect of "king-man+woman" applied to a state $s$ is equivalent to those of "queen":*

$$s \hat{+} \textit{king} \hat{-} \textit{man} \hat{+} \textit{woman} = s \hat{+} \textit{queen}$$
$$= s \setminus \{\textit{female}\} \cup \{\textit{status}\} \setminus \emptyset \cup \{\textit{female}\} \setminus \emptyset \cup \{\textit{female}\}.$$

**Example 2.** *The effect of "king+man" is equivalent to "king" itself as the semantic redundancy about "female" disappears in the set operation.*

$$s \hat{+} \textit{king} \hat{+} \textit{man} = s \setminus \{\textit{female}\} \cup \{\textit{status}\} \setminus \{\textit{female}\} \cup \emptyset$$
$$= s \hat{+} \textit{king}.$$

## 5 Evaluation

We trained a traditional CBOW model (our implementation) and our discrete word embedding on 1 Billion Word Language Model Benchmark dataset (Chelba et al. 2014). Training details are available in the appendix. We first compared the performance of the resulting embedding on several downstream tasks.

### 5.1 Downstream task evaluation

Word similarity task is the standard benchmark for measuring attributional similarity (Miller and Charles 1991; Resnik 1995; Agirre et al. 2009). Given a set of word pairs, each embedding is evaluated by computing the Spearman correlation between the similarity scores assigned by the embedding and those assigned by human (Rubenstein and Goodenough 1965; Faruqui and Dyer 2014; Myers, Well, and Lorch 2010). The scores for CBOW are obtained by the cosine similarity. For the DSAW embedding, the standard cosine distance is not directly applicable as each embedding consists of two binary vectors. We, therefore, turned the effect of a word $x$ into an integer vector of tertiary values $\{1, 0, -1\}$ by ADD$(x)$ − DEL$(x)$, then computed the cosine similarity. We tested our models with the baseline models on 5 different datasets (Bruni, Tran, and Baroni 2014; Radinsky et al. 2011; Luong, Socher, and Manning 2013; Hill, Reichart, and Korhonen 2015; Finkelstein et al. 2001).

Next, we evaluated Word Analogy task using the test dataset provided by Mikolov et al. (2013b). For CBOW models, we used 3COSADD method (Sec. 2) to approximate the target word. For the proposed models, we perform a similar analogy, SEQADD, which computes the combined effects $e$, turns it into the tertiary representation, then finds the most similar word using the cosine distance. Since our set-based arithmetic is not associative or commutative, we permuted the order of operations and report the best results obtained from $e = \hat{-}\ \boldsymbol{a} \hat{+} \boldsymbol{a}^* \hat{+} \boldsymbol{b}$. We counted the number of correct predictions in the top-1 and top-10 nearest neighbors. We excluded the original words ($a$, $a^*$ and $b$) from the candidates, following the later analysis of the Word2Vec implementations (Nissim, van Noord, and van der Goot 2020).

Finally, we used our embeddings for semantic text classification, in which the model must capture the semantic information to perform well. We evaluated our model in two datasets: "*20 Newsgroup*" (Lang 1995) and "*movie sentiment treebank*" (Socher et al. 2013). We created binary classification tasks following the existing work (Tsvetkov et al. 2015; Yogatama and Smith 2014): For *20 Newsgroup*, we picked 4 sets of 2 groups to produce 4 sets of classification problems (science.med vs. science.space, ibm.pc.hardware vs. mac.hardware, baseball vs. hockey, alt.atheism vs. soc.religion.christian.) For movie sentiment, we ignored the neutral comments and set a threshold for the sentiment values: $\leq 0.4$ as 0, and $> 0.6$ as 1. In both the CBOW and the DSAW model, we aggregated the word embeddings (by $+$ or $\hat{+}$) in a sentence or a document to obtain the sentence / document-level embedding. We then classified the results with a default L2-regularized logistic regression model in Scikit-learn. We recorded the accuracy in the test split and compared it across the models. We normalized the imbalance in the number of questions between subtasks ("*20 Newsgroup*" have $\approx 2000$ each, while "*movie sentiment*" has $\approx 9000$) and reported the averaged results.

Table 2 shows that the performance of our discrete embedding is comparable to the continuous CBOW embedding in these three tasks. This is a surprising result given that discrete embeddings are believed to carry less information in each dimension compared to the continuous counterpart and are believed to fail because they cannot model uncertainty.

### 5.2 Redundancy elimination

We next focus on the ability of DSAW model to eliminate the syntactic and the semantic redundancy. We used Principal Component Analysis to visualize the linear projection of the embedding space. Phrase embeddings are obtained by the repeated $\hat{+}$ (DSAW) or averaging (CBOW) – the latter choice is purely for the visualization (length does not affect the cosine distance.) For both models, we used the best performing hyperparameters in analogy.

In Fig. 4, we plotted syntactically and semantically redundant phrases "habit", "regular habit", "regular ... regular habit" (repeated 8 times). In continuous embeddings (left), phrase embeddings approach closer and closer to the embedding of "regular" as more "regular"s are added. On the course of additions, the vector tends to share the direction with irrelevant words such as "experiment" or "stunt". In contrast, in discrete embedding (right), semantically redundant addition of "regular" does not seem to drastically change the direction, nor share the direction with irrelevant words. Also, repetitive additions do not affect the discrete embedding.

### 5.3 Classical Planning in the embedding space

Finally, with a logically plausible representation of words, we show how it can be used by a symbolic AI system. We find "paraphrasing" an ideal task, where we provide an input

| Embedding size $E$ | 200 | | 500 | | 1000 | |
|---|---|---|---|---|---|---|
| Model | CBOW | DSAW | CBOW | DSAW | CBOW | DSAW |
| Word Similarity | 0.528 | 0.509 | 0.518 | 0.538 | 0.488 | **0.545** |
| Analogy Top1 acc. | **0.438** | 0.273 | 0.413 | 0.373 | 0.333 | 0.373 |
| Analogy Top10 acc. | 0.682 | 0.564 | 0.671 | **0.683** | 0.587 | 0.673 |
| Text Classification Test | 0.890 | 0.867 | 0.920 | 0.902 | 0.920 | **0.930** |

Table 2: Downstream task performance comparison between CBOW and DSAW with the best tuned hyperparameters. In all tasks, higher scores are better. Best results in bold.

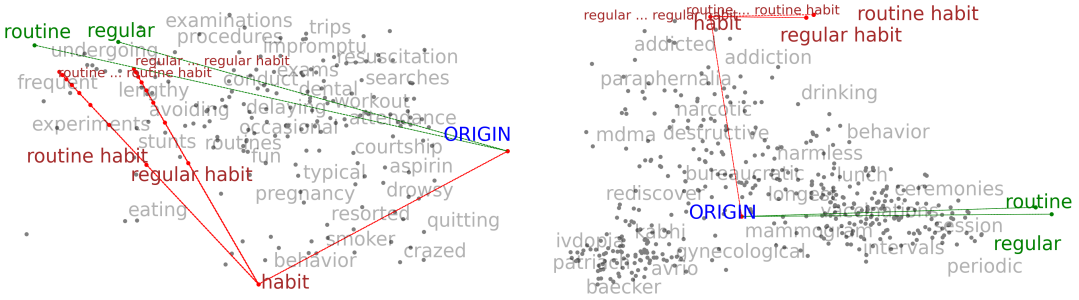


Figure 4: PCA plots of words/phrases in continous (Left)/discrete (Right) embeddings (best on computer screen). In all plots, we additionally included the union of 50 nearest neighbor words of each dot.

word $y$ and ask the system to discover the phrase that shares the same concept. Given a word $y$, we generate a classical planning problem whose task is to combine several words to achieve the effects included in $y$ in the correct order.

Formally, the instance $\langle P, A, I, G(y)\rangle$ is defined as follows: $P = P_{\text{add}} \cup P_{\text{del}} = \{p_i^a \mid i \in 1..E\} \cup \{p_i^d \mid i \in 1..E\}$, where $p_i^a, p_i^d$ are propositional symbols with unique names. Actions $a(x) \in A$ are built from each word $x$ in the vocabulary: $\text{PRE}(a(x)) = \emptyset$, $\text{ADD}(a(x)) = \{p_i^a \mid \text{ADD}(x)_i = 1\} \cup \{p_i^d \mid \text{DEL}(x)_i = 1\}$, $\text{DEL}(a(x)) = \{p_i^d \mid \text{ADD}(x)_i = 1\} \cup \{p_i^a \mid \text{DEL}(x)_i = 1\}$. Finally, $I = \emptyset$ and $G(y) = \{p_i^a \mid \text{ADD}(y)_i = 1\} \cup \{p_i^d \mid \text{DEL}(y)_i = 1\}$. Note that $\text{ADD}(y), \text{DEL}(x)$ etc. are bit-vectors, while $\text{ADD}(a(x))$ etc. are sets expressed in Planning Domain Description Language (PDDL) (Haslum et al. 2019). Finding the optimal solution of this problem is **NP**-Complete due to $\text{PRE}(a(x)) = \emptyset$ (Bylander 1994). Due to its worst-case hardness, we do not try to find the optimal solutions.

Notice that the goal condition of this planning problem is overly specific because the neighbors of an embedding vector often also carry a similar meaning. In fact, these instances tend to become unsolvable (i.e., no solution exists). To address this issue, we adapt the *net-benefit* planning formalism (Keyder and Geffner 2009), an extension of classical planning that allows the use of *soft-goals*. Net-benefit planning task $\langle P, A, I, G(y), c, u\rangle$ is same as the unit-cost classical planning except the cost function $c : A \to \mathbb{Z}^{+0}$ and $u : G(y) \to \mathbb{Z}^{+0}$. The task is to find an action sequence $\pi$ minimizing the cost $\sum_{a\in\pi} c(a) + \sum_{p\in G(y)\setminus s^*} u(p)$, i.e., the planner tries to find a cheaper path while also satisfying as many goals as possible at the terminal state $s^*$. We used a simple compilation approach (Keyder and Geffner 2009) to convert this net-benefit planning problem into a normal classical planning problem.

We specified both costs a constant: $c(a) = E$ for all actions and $u(p) = U$ for all goals, where we heuristically chose $U = 100$. We solved the problem with LAMA planner (Richter and Westphal 2010), the winner of International Planning Competition 2011 satisficing track. This configuration searches for suboptimal plans, iteratively refining the solution by setting the upper-bound based on the cost of the last solution. We generated 300 problems from the hand-picked target words $y$. $A$ was generated from the 4000 most-frequent words in the vocabulary ($V \approx 219$k) excluding $y$, function words (e.g., "the", "make"), and compound words (e.g., plurals). For each problem, we allowed the maximum of 4 hours runtime and 16GB memory. Typically the planner found the first solution early, and continued running until the time limit finding multiple better solutions. We show its example outputs in Table 3.

## 6 Related work

The study on the hybrid systems combining the connectionist and symbolic approaches has a long history (Wermter and Lehnert 1989). To our knowledge, none of the approaches attempts to generate a set of atomic propositional symbols from the corpus without supervision. Zhao, Lee, and Eskénazi (2018) proposed a discrete sentence representation, treating each sentence as an action. Chen, Min, and Sun (2018) improved the training efficiency with an intermediate discrete code between the vocabulary $V$ and the continuous embedding. These discrete representations lack the cube-space prior and thus the state transitions cannot be expressed as a consistent deterministic rule, precluding the symbolic logical analysis. In the intersection of planning and natural language processing, Geib and Steedman (2007)

| Word $y$ | paraphrasing $\pi$ | Word $y$ | paraphrasing $\pi$ | Word $y$ | paraphrasing $\pi$ |
|---|---|---|---|---|---|
| lamborghini | luxury built car; electric car unlike toyota | lake | young sea | intervention | necessary plan |
| subaru | motor toyota style ford | river | valley lake nearby delta | louisville | kentucky pittsburgh |
| fiat | italian toyota; italian ford alliance | valley | mountain tenessee area | laptop | personal mac device |
| sushi | restaurant fish maybe japanese | shout | bail speak | reactor | nuclear plant |
| grape | wine tree; wine orange | pepsi | diet apple drink | lesson | history addition |
| bold | fresh simple move | deck | floor roof; roof floor | slot | wish machine |
| fantasy | dream novel | grip | tight presence | learnt | learn yesterday |
| interrogatoin | cia torture | isolation | cuba situation | reconstruction | infrastructure recovery effort |

Table 3: Paraphrasing of the source words returned by the LAMA planner.

bridged hierarchical planning (HTN) (Ghallab, Nau, and Traverso 2004) formalisms and Context-Sensitive Grammar. Rieser and Lemon (2009) introduced a system which models conversations as probabilistic planning and learns a reactive policy from interactions. Recent approaches extract a classical planning model from a natural language corpus (Lindsay et al. 2017; Feng, Zhuo, and Kambhampati 2018), but using the opaque human symbols. An interesting avenue of future work is to learn a hierarchical planning model (Hogg, Kuter, and Munoz-Avila 2010) to model the structured aspect of the natural language.

## 7 Conclusion

We proposed an unsupervised learning method for discrete binary word embeddings that preserve the vector arithmetic similar to the continuous embeddings. Our approach combines three distant areas: Unsupervised representation learning method for natural language, discrete generative modeling, and STRIPS classical planning formalism which is deeply rooted in the symbolic AIs and the propositional logic. Inspired by the recurrent view of the Continuous Bag of Words model, our model represents each word as a symbolic action that modifies the binary (i.e., propositional) recurrent states through effects. Unlike the black-box recurrent methods, our framework can extract the effects applied by each word as explicit logical formulae. Our representation has several notable features: Logical robustness against redundancy and the compatibility to the highly-optimized off-the-shelf implementations of classical planners. We demonstrated that our discrete embedding fairs on par with the continuous embedding in several downstream tasks, contrary to the conventional wisdom that discreteness would lose too much expressivity compared to the continuous representations. Future work includes incorporating ideas in state-of-the-art systems such as LSTM (Hochreiter and Schmidhuber 1997) or BERT (Devlin et al. 2019) to our model, while staying within planning formalism.

## References


Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Pasca, M.; and Soroa, A. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *NAACL*, 19.

Alcázar, V.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting Regression in Planning. In *IJCAI*.

Amado, L.; Pereira, R. F.; Aires, J.; Magnaguagno, M.; Granada, R.; and Meneguzzi, F. 2018. Goal Recognition in Latent Space. In *IJCNN*.

Asai, M., and Fukunaga, A. 2018. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In *AAAI*.

Asai, M., and Muise, C. 2020. Learning Neural-Symbolic Descriptive Planning Models via Cube-Space Priors: The Voyage Home (to STRIPS). In *IJCAI*.

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract Meaning Representation for Sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186.

Bolukbasi, T.; Chang, K.; Zou, J. Y.; Saligrama, V.; and Kalai, A. T. 2016. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In *NIPS*, 4349–4357.

Bruni, E.; Tran, N.-K.; and Baroni, M. 2014. Multimodal Distributional Semantics. *J. Artif. Intell. Res.(JAIR)* 49:1–47.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence* 69(1):165–204.

Caliskan, A.; Bryson, J. J.; and Narayanan, A. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science* 356(6334):183–186.

Chelba, C.; Mikolov, T.; Schuster, M.; Ge, Q.; Brants, T.; Koehn, P.; and Robinson, T. 2014. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *INTERSPEECH*.

Chen, T.; Min, M. R.; and Sun, Y. 2018. Learning K-way D-dimensional Discrete Codes for Compact Embedding Representations. In *ICML*, volume 80, 853–862.

Cho, K.; van Merrienboer, B.; Gülçehre, c.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 1724–1734.

Cullen, J., and Bryman, A. 1988. The Knowledge Acquisition Bottleneck: Time for Reassessment? *Expert Systems* 5(3).

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 4171–4186.

Faruqui, M., and Dyer, C. 2014. Community Evaluation and Exchange of Word Vectors at Wordvectors.org. In *ACL*, 19–24.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In *IJCAI*.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and Executing Generalized Robot Plans. *Artificial Intelligence* 3(1-3):251–288.

Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; and Ruppin, E. 2001. Placing Search in Context: The Concept Revisited. In *WWW*, 406–414.

Flanigan, J.; Thomson, S.; Carbonell, J. G.; Dyer, C.; and Smith, N. A. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *ACL*, 1426–1436.

Geib, C. W., and Steedman, M. 2007. On Natural Language Processing and Plan Recognition. In *IJCAI*, 1612–1617.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Elsevier.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. An Introduction to the Planning Domain Definition Language. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13(2):1–187.

Hill, F.; Reichart, R.; and Korhonen, A. 2015. Simlex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *Computational Linguistics* 41(4):665–695.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Hogg, C.; Kuter, U.; and Munoz-Avila, H. 2010. Learning Methods to Generate Good Plans: Integrating HTN Learning and Reinforcement Learning. In *AAAI*.

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 448–456.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.

Jia, R., and Liang, P. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *EMNLP*, 2021–2031.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *J. Artif. Intell. Res.(JAIR)* 36:547–556.

Kingma, D. P., and Welling, M. 2013. Auto-Encoding Variational Bayes. In *ICLR*.

Kurutach, T.; Tamar, A.; Yang, G.; Russell, S.; and Abbeel, P. 2018. Learning Plannable Representations with Causal InfoGAN. In *NIPS*.

Lang, K. 1995. Newsweeder: Learning to Filter Netnews. In *ICML*, 331–339.

Levy, O., and Goldberg, Y. 2014. Linguistic Regularities in Sparse and Explicit Word Representations. In *CoNLL*, 171–180.

Lindsay, A.; Read, J.; Ferreira, J. F.; Hayton, T.; Porteous, J.; and Gregory, P. J. 2017. Framer: Planning Models from Natural Language Action Descriptions. In *ICAPS*.

Luong, M.-T.; Socher, R.; and Manning, C. D. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *CoNLL*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*, 3111–3119.

Mikolov, T.; Yih, W.-t.; and Zweig, G. 2013. Linguistic Regularities in Continuous Space Word Representations. In *NAACL*, 746–751.

Miller, G. A., and Charles, W. G. 1991. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes* 6(1):1–28.

Myers, J.; Well, A.; and Lorch, R. 2010. *Research Design and Statistical Analysis*. Routledge.

Nissim, M.; van Noord, R.; and van der Goot, R. 2020. Fair is Better than Sensational: Man is to Doctor as Woman is to Doctor. *Computational Linguistics* Just Accepted.

Payan, C. 1992. On the Chromatic Number of Cube-Like Graphs. *Discrete mathematics* 103(3).

Radinsky, K.; Agichtein, E.; Gabrilovich, E.; and Markovitch, S. 2011. A Word at a Time: Computing Word Relatedness using Temporal Semantic Analysis. In *WWW*, 337–346.

Resnik, P. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *IJCAI*, 448–453.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.(JAIR)* 39(1):127–177.

Rieser, V., and Lemon, O. 2009. Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, 683–691.

Rubenstein, H., and Goodenough, J. B. 1965. Contextual Correlates of Synonymy. *Communications of the ACM* 8(10):627–633.

Schakel, A. M. J., and Wilson, B. J. 2015. Measuring Word Significance using Distributed Representations of Words. *CoRR* abs/1508.02297.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *EMNLP*, 1631–1642.

Tsvetkov, Y.; Faruqui, M.; Ling, W.; Lample, G.; and Dyer, C. 2015. Evaluation of Word Vector Representations by Subspace Alignment. In *EMNLP*, 2049–2054.

Wang, C.; Xue, N.; and Pradhan, S. 2015. A Transition-based Algorithm for AMR Parsing. In *NAACL*, 366–375.

Wermter, S., and Lehnert, W. G. 1989. A Hybrid Symbolic/Connectionist Model for Noun Phrase Understanding. *Connection Science* 1(3):255–272.

Wilson, B. J., and Schakel, A. M. J. 2015. Controlled Experiments for Word Embeddings. *CoRR* abs/1510.02675.

Xing, C.; Wang, D.; Liu, C.; and Lin, Y. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *NAACL*, 1006–1011.

Yogatama, D., and Smith, N. A. 2014. Linguistic Structured Sparsity in Text Categorization. In *ACL*, 786–796.

Zhao, T.; Lee, K.; and Eskénazi, M. 2018. Unsupervised Discrete Sentence Representation Learning for Interpretable Neural Dialog Generation. In *ACL*, 1098–1107.