Abstract

In this paper, we propose an unsupervised neural model 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. We demonstrate that our embedding is directly compatible with the symbolic, classical planning solvers by performing a “paraphrasing” task. Due to the discrete/logical decision making in classical algorithms with deterministic (non-probabilistic) completeness, and also because it does not require additional training on the paraphrasing dataset, our system can negatively answer a paraphrasing query (inexistence of solutions), and can answer that only some approximate solutions exist — A feature that is missing in the recent, huge, purely neural language models such as GPT-3.

1 Introduction

After the initial success of the distributed word representation in Word2Vec [52], 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 [14][10].

In recent years, significant progress has been made [6, 32, 38, 7, 8, 5, 8] in the field of Automated Planning on resolving the so-called Knowledge Acquisition Bottleneck [18], 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 [23][28] 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 [16], 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 [53] 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 standard method for obtaining such a prediction is 3COSADD [53], which attempts to find the closest word embedding to a vector a* − a + b measured by the cosine distance \( \cos(v_1, v_2) = 1 - \frac{v_1 \cdot v_2}{|v_1||v_2|} \), assuming that the result is close to the target embedding b*. This, along with other analogy calculation methods [45, 59, 20], 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 [76, 32].

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 a and b, \( \cos(a \cdot n + b, a) \rightarrow 0, (n \rightarrow \infty) \) (Fig. 1). Even with a more sophisticated aggregation method for a vector sequence, such as the recurrent neural networks [32], 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 [57]. This issue is also related to the fact that word2vec embedding encodes important information in its magnitude [71, 29]. While Xing et al. [31] 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) [9] 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 [25, 77], 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

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{The shortcoming of adding continuous vectors in a cosine vector space.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Cube-Space AE [8] for a visual time series \((x^i, x^{i'})\) that applies Back-to-Logit to the binary latent representation \((s^i, s^{i'})\) and the action vector \(a^i\).}
\end{figure}
Applying an action \( x \) when \( x \) word 2 word of a ...

To achieve this goal, we combine the existing discrete variational method with CBOW Word2Vec that restricts the state transition graph to be an instance of directed modeling the time series data, cube-space prior \([8]\) is a structural prior for a discrete latent space.

In the context of by the actions, i.e., for any edge \( (s, a, \pi, I, G) \) is defined as a 4-tuple \((P, A, I, G)\) 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 \((\text{PRE}(a), \text{ADD}(a), \text{DEL}(a))\) where \( \text{PRE}(a), \text{ADD}(a), \text{DEL}(a) \subseteq P \) are preconditions, add-effects, and delete-effects, respectively. \( \text{ADD}(a) \cap \text{DEL}(a) = \emptyset \). 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 \( \text{PRE}(a) \), i.e., \( \text{PRE}(a) \subseteq s \).

Applying an action \( a \) to \( s \) yields a new successor state \( s' = a(s) = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a) \). The solution to a classical planning problem is called a plan, which is a sequence of actions \( \pi = (a_1, a_2, \ldots, a_n) \) that leads to a terminal state \( s' = a_1 | \cdots | a_n(s) \) 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 \( s \in \{0, 1\}^{|P|} \) where, for each \( j \)-th proposition \( p_j \in P \), \( s_j = 1 \) when \( p_j \in s \), and \( s_j = 0 \) when \( p_j \not\in s \). A state transition graph of a classical planning problem is a graph \( G = (\mathcal{V}, \mathcal{E}) \) 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 \( a \in A \).

### 3 Preliminary and background

We denote a multi-dimensional array in bold and its subarrays with a subscript (e.g., \( x \in \mathbb{R}^{N \times M} \), \( x_j \in \mathbb{R}^M \)), 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 element-wise.

**Word2Vec Continuous Bag of Word (CBOW) with Negative Sampling.** The CBOW with Negative Sampling \([51, 52]\) 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 \((x^{i-c}, \ldots, x^{i+c})\) \((x^i \in 1..V)\) in a dataset \( X = \{x^i\} \), it computes the continuous-bag-of-words representation \( \mathbf{c}_i = \sum_{-c \leq j \leq c, j \neq 0} W_{x^{i+j}} \). 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 \((x^k, k \in 1..K)\), extracts their embeddings \( W'_{x^k} \), then maximizes the loss: \( \log \sigma(\mathbf{c}_i \cdot W'_{x^i}) + \sum_{k=1}^K \log \sigma(-\mathbf{c}_i \cdot W'_{x_k}) \).

**Variational AutoEncoder with Gumbel Softmax and Binary Concrete distribution.** 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 \( p(x) \). Gumbel-Softmax (GS) VAE \([36]\) and its binary special case Binary Concrete VAE \([50]\) instead use a discrete, uniform categorical distribution 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 \( z = \text{BC}(x) = \text{SIGMOID}(x + \text{LOGISTIC}(0, 1)/\tau) \), where \( \text{LOGISTIC}(0, 1) = \log u - \log(1 - u) \) and \( u \in [0, 1] \) is sampled from Uniform(0, 1). BinConcrete converges to the Heaviside step function at the limit \( \tau \to 0 \): BinConcrete \((x) \to \text{STEP}(x) \) (step function thresholded at 0).

**Classical Planning.** A grounded (propositional) unit-cost STRIPS Planning problem \([23, 28]\) is defined as a 4-tuple \((P, A, I, G)\) 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 \((\text{PRE}(a), \text{ADD}(a), \text{DEL}(a))\) where \( \text{PRE}(a), \text{ADD}(a), \text{DEL}(a) \subseteq P \) are preconditions, add-effects, and delete-effects, respectively.

### Cube-space prior and Back-to-Logit technique for STRIPS action modeling

In the context of modeling the time series data, cube-space prior \([8]\) is a structural prior for a discrete latent space that restricts the state transition graph to be an instance of directed cube-like graphs \([60]\). The state transition graph of any STRIPS planning problem is an instance of a directed cube-like graph \([8]\).
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.

Back-to-Logit (BTL) technique [8] 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) [33], 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 \( \text{EFFECT}(a) \), the current state and the successor state binary latent vector \( s^i \) and \( s'^i \) (\( i \): index in a dataset), the successor state is predicted by:

\[
s'^i \approx \text{APPLY}(a, s^i) = \text{BC}(\text{BN}(s^i) + \text{EFFECT}(a)).
\]

The state representation \( s \) trained with BTL has the following properties:

**Theorem 1** [8]. Under the same action \( a \), the state transitions are monotonic and deterministic:

- (add effect): \( \exists i; (s^i_j, s'^i_j) = (0, 1) \Rightarrow \forall i; (s^i_j, s'^i_j) = (0, 1) \) or \( (1, 1) \),
- (delete effect): \( \exists i; (s^i_j, s'^i_j) = (1, 0) \Rightarrow \forall i; (s^i_j, s'^i_j) = (1, 0) \) or \( (0, 0) \) (for each dimension \( j \)).

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 \( s' = (s \cup \text{DEL}(a)) \cup \text{ADD}(a) \) and a constraint \( \text{DEL}(a) \cap \text{ADD}(a) = \emptyset \). Latplan system [6, 7, 8, 9] 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 [32] or GRU [17], 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 \( W_x \) into the output state \( s^{t+c} = s^0 + \sum_{-c \leq j \leq c, j \neq 0} W_{x^{t+j}}. \) Coincidentally, this output state is merely the sum of the effect vectors if \( s^0 \) is 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 \( \text{add(+), remove(-) or keep(0)} \) 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 et al. [16] (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 (DSAW)

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 \( s^i \) 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 \( \text{ADD}(x) \), \( \text{DEL}(x) \) of a word \( x \) that satisfy \( s^{i+1} = (s^i \& \& \neg \text{DEL}(x)) \| \text{ADD}(x) \), which is a bit-vector implementation of set-based STRIPS action application \( s^{i+1} = (s^i \setminus \text{DEL}(a)) \cap \text{ADD}(a) \).

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 \( \log \sigma(x \cdot y) \) in Word2Vec to the output state because it assumes that the distribution of \( x, y \in \mathbb{R}^E \) 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 \( \{r_1, \ldots, r_K\} \)) is defined as follows, where \( s^i = \text{APPLY}(x^i, s^0) \), \( s^{i+1} = \text{APPLY}(x^{i+1}, s^{i+1}) \), \( s^{i+1} = \text{APPLY}(x^{i+1}, s^{i+1}) \), \( s^{i+j} = \text{APPLY}(x^{i+j}, s^{i+j}) \) (\( j \notin \{-c, 0, 1\} \)):

\[
\log \sigma((s^{i+c} - 0.5) \cdot (s^i - 0.5)) + \sum_{k=1}^K \log \sigma(-(s^{i+c} - 0.5) \cdot (\text{APPLY}(r^k, s^0) - 0.5)).
\]

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

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 \([52]\), 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 \([2, 28]\) (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\([3]\) restores the original state of a STRIPS progression \( s^{i+1} = (s^i \setminus \text{DEL}(a)) \cap \text{ADD}(a) \) by \( s^i = (s^{i+1} \setminus \text{ADD}(a)) \cap \text{DEL}(a) \). For a word \( x \), we denote the corresponding bitwise operations as \( \oplus x \) and \( \ominus x \). We note that our operation is not associative or commutative. That is, the result of “king+woman” may be different from “king+woman-man” etc.

Next, for a sequence of operations \( sR_1x_1 \ldots R_nx_n (R_i \in \{+, \ominus\}) \), we denote its combined effects as \( e = R_1x_1 \ldots R_nx_n \). Its add/delete-effects, \( \text{ADD}(e), \text{DEL}(e) \), are recursively defined as follows:

\[
\begin{align*}
\text{ADD}(e \oplus x) &= \text{ADD}(e) \setminus \text{DEL}(x) \cup \text{ADD}(x), & \text{DEL}(e \oplus x) &= \text{DEL}(e) \setminus \text{ADD}(x) \cup \text{DEL}(x), \\
\text{ADD}(e \ominus x) &= \text{ADD}(e) \setminus \text{ADD}(x) \cup \text{DEL}(x), & \text{DEL}(e \ominus x) &= \text{DEL}(e) \setminus \text{DEL}(x) \cup \text{ADD}(x).
\end{align*}
\]

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.

Note that here we assume that the effect always invoke changes to the state in order to obtain a deterministic outcome from the regression. In the standard setting, the regression is nondeterministic unless warranted by the preconditions, e.g., if \( p_1 \in \text{PRE}(a) \land p_1 \in \text{DEL}(a) \), then \( p_1 \) is guaranteed to be true before applying the action \( a \).
Table 1: An example 2-dimensional embedding.

| word x | DEL(x) (set interpretation) | ADD(x) (set interpretation) |
|--------|-----------------------------|----------------------------|
| King   | [1, 0] = {female}           | [0, 1] = {status}           |
| Man    | [1, 0] = {female}           | [0, 0] = ∅                 |
| Woman  | [0, 0] = ∅                  | [1, 0] = {female}          |
| Queen  | [0, 0] = ∅                  | [1, 1] = {female, status}  |

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.

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. Then the effect of “king+man+woman” applied to a state $s$ is equivalent to those of “queen”:

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

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

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

5 Evaluation

We trained a traditional CBOW model (our implementation) and our discrete word embedding on 1 Billion Word Language Model Benchmark dataset [15]. 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 [54, 64, 1]. 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 [68, 21, 56]. 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 $\text{ADD}(x) - \text{DEL}(x)$, then computed the cosine similarity. We tested our models with the baseline models on 5 different datasets [12, 63, 49, 31, 24].

Next, we evaluated Word Analogy task using the test dataset provided by Mikolov et al. [52]. For CBOW models, we used 3COSADD method (Sec. 3) 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 $e = -a + a^* + b$. 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 [59].

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” [44] and “movie sentiment treebank” [72]. We created binary classification tasks following the existing
work\textsuperscript{[76, 82]}: For 20 Newsgroup, we picked 4 sets of 2 groups to produce 4 sets of classification problems: SCI (science.med vs. science.space), COMP (ibm.pc.hardware vs. mac.hardware), SPORT (baseball vs. hockey), RELI (alt.atheism vs. soc.religion.christian). For movie sentiment (MS), 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 (SCI, ...RELI have \( \approx 2000 \) questions each while MS has \( \approx 9000 \)) and reported the averaged results.

**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. The training/dataset detail and the more in-depth analyses can be found in the appendix.

### 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\textsuperscript{[61]} to visualize the linear projection of the embedding space. Phrase embeddings are obtained by the repeated + (DSAW) or averaging (CBOW) – the latter choice is purely for the visualization (length does not affect the cosine distance.) For CBOW and DSAW, we used the models that performed the best in analogy task.

In Fig. 4 we plotted syntactically and semantically redundant phrases “habit”, “regular habit”, “regular ... regular habit” (repeated 8 times). Continuous 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, 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 discrete 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 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: \( P = P_{\text{add}} \cup P_{\text{del}} = \{ p^a_i \mid i \in 1..E \} \cup \{ p^d_i \mid i \in 1..E \} \), where \( p^a_i, p^d_i \) are propositional symbols with unique names. Actions \( a(x) \in A \) are built from each word \( x \) in the vocabulary: \( \text{PRE}(a(x)) = \emptyset, \text{ADD}(a(x)) = \{ p^a_i \mid \text{ADD}(x)_i = 1 \} \cup \{ p^d_i \mid \text{DEL}(x)_i = 1 \}, \text{DEL}(a(x)) = \{ p^a_i \mid \text{ADD}(x)_i = 1 \} \cup \{ p^d_i \mid \text{DEL}(x)_i = 1 \} \). Finally, \( I = \emptyset \) and \( G(y) = \{ p^a_i \mid \text{ADD}(y)_i = 1 \} \cup \{ p^d_i \mid \text{DEL}(y)_i = 1 \} \). Note that ADD\( (y), \text{DEL}(x) \) etc. are bit-vectors.
Table 3: Paraphrasing of the source words returned by the LAMA planner. See Table 12-13 in the appendix for more examples.

| Word y | word sequence $\pi$ (solution plan) | Word y | word sequence $\pi$ (solution plan) |
|--------|-------------------------------------|--------|-------------------------------------|
| hamburgur | meat lunch chain eat | lake | young sea |
| lamborghini | luxury built car; car recall standard; pond | lamborghini | $\pi$ (solution plan) |
| lamborghini | electric car unlike toyota | river | valley lake nearby delta |
| subaru | motor toyota style ford | valley | mountain tennessee area |
| fiat | italian toyota; italian ford alliance | shout | bail speak |
| sushi | restaurant fish maybe japanese | yell | wish talk |
| onion | add sweet cook | coke | like fat |
| grape | wine tree; wine orange | pepsi | diet apple drink |

While $\text{ADD}(a(x))$ etc. are sets expressed in Planning Domain Description Language (PDDL) [28], finding the optimal solution of this problem is $\text{NP}$-Complete due to $\text{PRE}(a(x)) = 0$ [13]. 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 [38], an extension of classical planning that allows the use of soft-goals. Net-benefit planning task $(P, A, I, G(y), c, u)$ is same as the unit-cost classical planning except the cost function $c : A \rightarrow \mathbb{Z}^+ + 0$ and $u : G(y) \rightarrow \mathbb{Z}^+ + 0$. 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 [38] 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 [65], the winner of International Planning Competition 2011 satisficing track [48]. 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 68 problems from the hand-picked target words $y$. $A$ was generated from the 4000 most-frequent words in the vocabulary ($V \approx 219k$) 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. See Appendix for the more variety of paraphrasing results using the 300 words randomly selected from the vocabulary.

6 Related work

The study on the hybrid systems combining the connectionist and symbolic approaches has a long history [78, 75]. To our knowledge, none of the approaches attempts to generate a set of atomic propositional symbols from the corpus without supervision. Our work is heavily inspired by the philosophical work of Russell [69] and Wittgenstein [80] on Logical Atomism, i.e., the decomposition of a concept into orthogonal logical atomic concepts. For example, the word “king” could be further decomposed to boolean properties like “human”, “male”, “leader”, “in a high social status”, “feudal”, “outdated”, and “from the medieval era”. Lack of these atomic concepts inhibits the in-depth automated logical analysis of natural language inputs using symbolic methods such as theorem provers, boolean satisfaction solvers or classical planners.

Zhao et al. [83] proposed a discrete sentence representation, treating each sentence as an action. Chen et al. [16] 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 [8] 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, Rieser and Lemon [66] 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 [46, 22], but using the opaque human symbols.

7 Discussion

An important aspect of classical algorithms is completeness, which gives language models a new capability fundamentally unachievable in popular probabilistic reasoning algorithms. A complete search algorithm (such as A*, or Greedy Best First Search in LAMA) is guaranteed to find a solution whenever there is one or halt otherwise in a finite amount of time [70, p80]. Probabilistic algorithms have weaker guarantees: Beam Search typically applied to language models is a form of hill climbing (incomplete); MCTS is only probabilistically complete and requires an infinite runtime to guarantee. Complete algorithms are able to return a provably negative answer to a question by ruling out the existence of solutions (cf. Unsolvability International Planning Competition [55]). We demonstrated that planners can prove that there was no precise paraphrasing.

Recently, CogSci and Causal Inference communities are becoming increasingly confident that such a feature, achieved by the counterfactual (what-if) queries operating the symbolic algorithms, is important “post GPT-3 [11]”. Even a gigantic GPT-3 model, despite other impressive results, still fails at negatively answering Out-of-Distribution(OoD) logical queries, e.g., “Who was the U.S. president in 1600?” [43] (The proper answer is “No one”, since the United States did not exist in 1600). Similar straightforward machine learning models, such as text-to-text RNN models, are trained on and are designed to interpolate within the given paraphrasing distribution. In contrast, classical search algorithms work efficiently completely without training, and therefore are designed from the ground up to work on queries that it has never seen.

A related issue in the current mainstream language model is its computational cost. It has been shown that the current progress of recent neural models is limited by the computing power of the hardware accelerator and is practically and environmentally unsustainable [74]. In contrast, classical, symbolic approaches are based on efficient branch-and-bound based pruning rather than brute-force computation, does not require additional training for decision making, and only requires the representation learning. We therefore argue that the neural-symbolic hybrid approach may be able to alleviate the issue.

While the current DSAW model is equivalent to the CBOW model in terms of model complexity and does not capture the complex aspects of human languages, it is a bottom-up approach toward learning Hierarchical Task Network planning (HTN) [27] formalism, the more expressive hierarchical extension of Classical Planning that can encode distant, hierarchical action dependencies. HTN formalism is able to model Context-Sensitive Grammar [26], a formal language class in Chomsky hierarchy that models distant, hierarchical word dependencies in a sentence. While existing work on learning the HTN planning model [58, 24, 84] relies on symbolic plan traces, we aim to learn a HTN model directly from the NLP corpus in the future. Corresponding DSAW model will use attentions in BERT [19] or Transformer [62] to model distant word dependencies, or Tree-LSTMs [73] or probabilistic grammar models [39] to explicitly model the hierarchical structure in HTN and natural language.

Finally, note that planning is merely a single case study. Concise discrete representation opens door to state-of-the-art symbolic methods (theorem provers, boolean satisfiability, constraint optimization) which all share the property. Lastly, note that deterministic algorithms and MCTS share the policy learnability through RL [67].

8 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 extending our model to the deep neural system such as LSTM [32], BERT [19] or GPT-2 [62], while preserving as much logical formalism as possible.

References

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, page 19, 2009.

[2] Vidal Alcázar, Daniel Borrajo, Susana Fernández, and Raquel Fuentetaja. Revisiting Regression in Planning. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2013.

[3] Leonardo Amado, Ramon Fraga Pereira, Joao Aires, Mauricio Magnaguagno, Roger Granada, and Felipe Meneguzzi. Goal Recognition in Latent Space. In Proc. of International Joint Conference on Neural Networks (IJCNN), 2018.

[4] Leonardo Amado, Ramon Fraga Pereira, Joao Aires, Mauricio Magnaguagno, Roger Granada, and Felipe Meneguzzi. LSTM-based Goal Recognition in Latent Space. arXiv preprint arXiv:1808.05249, 2018.

[5] Masataro Asai. Unsupervised Grounding of Plannable First-Order Logic Representation from Images. In Proc. of the International Conference on Automated Planning and Scheduling (ICAPS), 2019.

[6] Masataro Asai and Alex Fukunaga. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In Proc. of AAAI Conference on Artificial Intelligence, 2018.

[7] Masataro Asai and Hiroshi Kajino. Towards Stable Symbol Grounding with Zero-Suppressed State AutoEncoder. In Proc. of the International Conference on Automated Planning and Scheduling (ICAPS), 2019.

[8] Masataro Asai and Christian Muise. Learning Neural-Symbolic Descriptive Planning Models via Cube-Space Priors: The Voyage Home (to STRIPS). In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2020.

[9] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, pages 178–186, 2013.

[10] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In Advances in Neural Information Processing Systems, pages 4349–4357, 2016.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. CoRR, abs/2005.14165, 2020. URL [https://arxiv.org/abs/2005.14165](https://arxiv.org/abs/2005.14165).

[12] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal Distributional Semantics. J. Artif. Intell. Res.(JAIR), 49:1–47, 2014.

[13] Tom Bylander. The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence, 69(1):165–204, 1994.

[14] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334):183–186, 2017.
[15] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In Proc. of Annual Conference of the International Speech Communication Association, 2014.

[16] Ting Chen, Martin Renqiang Min, and Yizhou Sun. Learning K-way D-dimensional Discrete Codes for Compact Embedding Representations. In Proc. of the International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 853–862, 2018.

[17] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proc. of Conference on Empirical Methods in Natural Language Processing, pages 1724–1734, 2014.

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

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4171–4186, 2019.

[20] Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. Word Embeddings, Analogies, and Machine Learning: Beyond King-Man+Woman=Queen. In Proc. of the International Conference on Computational Linguistics, pages 3519–3530, 2016.

[21] Manaal Faruqui and Chris Dyer. Community Evaluation and Exchange of Word Vectors at Wordvectors.org. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 19–24, 2014.

[22] Wenfeng Feng, Hankz Hankui Zhuo, and Subbarao Kambhampati. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2018.

[23] Richard E Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and Executing Generalized Robot Plans. Artificial Intelligence, 3(1-3):251–288, 1972.

[24] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing Search in Context: The Concept Revisited. In Proc. of the International Conference on World Wide Web, pages 406–414, 2001.

[25] Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 1426–1436, 2014.

[26] Christopher W. Geib and Mark Steedman. On Natural Language Processing and Plan Recognition. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), pages 1612–1617, 2007.

[27] Malik Ghallab, Dana Nau, and Paolo Traverso. Automated Planning: Theory and Practice. Elsevier, 2004.

[28] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muiše. An Introduction to the Planning Domain Definition Language. Synthesis Lectures on Artificial Intelligence and Machine Learning, 13(2):1–187, 2019.

[29] Malte Helmert. The Fast Downward Planning System. J. Artif. Intell. Res.(JAIR), 26:191–246, 2006.

[30] Irina Higgins, Loïc Matthey, Arka Pal, et al. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In Proc. of the International Conference on Learning Representations, 2017.

[31] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. Computational Linguistics, 41(4):665–695, 2015.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.
[52] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[53] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

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

[55] Christian Muise and Nir Lipovetzky, editors. *Unsolvability International Planning Competition: Planner Abstracts*, 2016. URL [https://unsolve-ipc.eng.unimelb.edu.au/](https://unsolve-ipc.eng.unimelb.edu.au/).

[56] J.L. Myers, A. Well, and R.F. Lorch. *Research Design and Statistical Analysis*. Routledge, 2010. ISBN 9780805864311. URL [https://books.google.com/books?id=nbs0IJJ_saUAC](https://books.google.com/books?id=nbs0IJJ_saUAC).

[57] Masato Neishi, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda. A Bag of Useful Tricks for Practical Neural Machine Translation: Embedding Layer Initialization and Large Batch Size. In *Proceedings of the 4th Workshop on Asian Translation*, pages 99–109, 2017.

[58] Negin Nejati, Pat Langley, and Tolga Konik. Learning Hierarchical Task Networks by Observation. In *Proc. of the International Conference on Machine Learning*, 2006.

[59] Malvina Nissim, Rik van Noord, and Rob van der Goot. Fair is Better than Sensational: Man is to Doctor as Woman is to Doctor. *Computational Linguistics*, Just Accepted, 2020.

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

[61] Karl Pearson. LIII. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[62] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8):9, 2019.

[63] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A Word at a Time: Computing Word Relatedness using Temporal Semantic Analysis. In *Proc. of the International Conference on World Wide Web*, pages 337–346, 2011.

[64] Philip Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 448–453, 1995.

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

[66] Verena Rieser and Oliver Lemon. 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)*, pages 683–691, 2009.

[67] Or Rivlin, Tamir Hazan, and Erez Karpas. Generalized planning with deep reinforcement learning. *CoRR*, abs/2005.02305, 2020. URL [https://arxiv.org/abs/2005.02305](https://arxiv.org/abs/2005.02305).

[68] Herbert Rubenstein and John B Goodenough. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[69] Bertrand Russell. The Philosophy of Logical Atomism. *The Monist*, pages 345–380, 1919.

[70] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial Intelligence: A Modern Approach*, volume 2, 1995.

[71] Adriaan M. J. Schakel and Benjamin J. Wilson. Measuring Word Significance using Distributed Representations of Words. *CoRR*, abs/1508.02297, 2015.
Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In Proc. of Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, 2013.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 1556–1566. The Association for Computer Linguistics, 2015. doi: 10.3115/v1/p15-1150. URL https://doi.org/10.3115/v1/p15-1150.

Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The Computational Limits of Deep Learning. CoRR, abs/2007.05558, 2020. URL https://arxiv.org/abs/2007.05558.

Geoffrey G Towell and Jude W Shavlik. Knowledge-Based Artificial Neural Networks. Artificial Intelligence, 70(1-2):119–165, 1994.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of Word Vector Representations by Subspace Alignment. In Proc. of Conference on Empirical Methods in Natural Language Processing, pages 2049–2054, 2015.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. A Transition-based Algorithm for AMR Parsing. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 366–375, 2015.

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

Benjamin J. Wilson and Adriaan M. J. Schakel. Controlled Experiments for Word Embeddings. CoRR, abs/1510.02675, 2015.

Ludwig Wittgenstein. Tractatus Logico Philosophicus. 1922.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1006–1011, 2015.

Dani Yogatama and Noah A Smith. Linguistic Structured Sparsity in Text Categorization. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 786–796, 2014.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskénazi. Unsupervised Discrete Sentence Representation Learning for Interpretable Neural Dialog Generation. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 1098–1107, 2018.

Broader Impact

This work tackles the fundamental question of whether it is possible to bring the logical reasoning capability readily available in the symbolic systems to the perceptual agents based on neural networks. While it is difficult to predict the broader impact of having a logical agents from this work alone, we believe such a deliberative agent tends to make a saner, logically plausible and unbiased decision making based on logic, compared to the reactive policy learned from the experiences. This in general would have a positive impact to the society. The main reason of this claim is that generally logical reasoning systems do not completely rely on past experiences and use them only as an advice. However, we should note that the goal of this paper is not to assess the fairness of the resulting representation, which is an interesting avenue for future work. The potential downside of our discrete embedding will be the slight performance degradation due to the early stage of the research, compared to the work in the state-of-the-art continuous language modeling literature. However, we hope to see this improved in the future work that combines our idea with the more complex state-of-the-art architecture. Another downside could be the size of the embedding. Compared to the continuous embeddings, the discrete embeddings tends to require more dimensions in order to perform well on the downstream tasks, which may impact the computational cost.
## Contents

1 Introduction 1

2 Continuous embedding considered harmful 2

3 Preliminary and background 3

4 Model architecture 4

  4.1 Discrete Sequential Application of Words (DSAW) 5

  4.2 Inference in the discrete space 5

5 Evaluation 6

  5.1 Downstream task evaluation 6

  5.2 Redundancy elimination 7

  5.3 Classical Planning in the discrete embedding space 7

6 Related work 8

7 Discussion 9

8 Conclusion 9

A Machine learning experiments 16

  A.1 Source code directory discrete-word-embedding/ 16

  A.2 Training dataset preparation 16

  A.3 Training details 16

  A.4 Additional experiments: Weight initialization with Logistic(0,1) distribution 17

  A.5 Additional model experiments: Discrete implementation of SkipGram 17

  A.6 Additional model experiments: Hybrid discrete-continuous CBOW model 18

  A.7 Detailed, per-category results for the word similarity task 19

  A.8 Detailed, per-category results for the analogy task 19

  A.9 Exploring the best ordering of the discrete additive operations in analogy task 19

  A.10 Detailed, per-category results for the text classification task 21

  A.11 Additional visualizations for the word compositionality 22

B Planning / paraphrasing experiments 25

  B.1 The archive directory paraphrasing/ 25

  B.2 Compilation of a net-benefit planning problem into a classical planning problem 25

  B.3 The list of 68 target words used in the paraphrasing experiment 25

  B.4 The statistics of the discrete effect vectors 25

  B.5 Runtime statistics for the word paraphrasing experiment 26

  B.6 Additional paraphrasing for a set of randomly selected 300 words 26
A Machine learning experiments

A.1 Source code directory discrete-word-embedding/

The directory discrete-word-embedding/ contains the source code for reproducing our experiments, including training, evaluation, plotting and paraphrasing. For details, find the enclosed README.org file in the directory.

A.2 Training dataset preparation

For the model training, we used 1 Billion Word Language Model Benchmark dataset [15] available from https://www.statmt.org/lm-benchmark/. Since the archive contains only the training set and the test set, we split the training set into the training and the validation set by 99:1. The dataset is already tokenized. However, we further downcased each word in order to reduce the size of the vocabulary. Since the vocabulary does not distinguish certain proper nouns, this will equally affect the accuracy across all models trained and evaluated in this paper.

After the split, we pruned the words that appear less than 10 times in the corpus. We further reduced the size of the corpus by removing the frequent words, as suggested in the original Word2Vec paper [52]. However, the formula for computing the probability of dropping a word described in the paper is different from the actual implementation published on their website https://code.google.com/archive/p/word2vec/. We followed the actual implementation for calculating the probability.

In the paper, the probability \( p(x) \) of dropping a word \( x \) in the corpus is given by

\[
p(x) = 1 - \sqrt{\frac{t}{f(x)}}
\]

where \( t \) is a threshold hyperparameter and \( f(x) \) is a frequency of the word in the corpus. For example, if the word appeared 5 times in a corpus consisting of 100 words, \( f(x) = 0.05 \). The paper recommends \( t = 10^{-5} \). However, the actual implementation uses the formula

\[
p(x) = 1 - \left( \sqrt{\frac{f(x)}{t}} + 1 \right) \frac{t}{f(x)}
\]

with \( t = 10^{-4} \) as the default parameter.

A.3 Training details

The training is performed by batched stochastic gradient descent using Rectified Adam optimizer [47] for 8 epochs, batch-size 1000. Each training took maximum of around 32 hours on a single Tesla V100 GPU. For CBOW, the loss function is same as that of the original work:

\[
\log \sigma(e^i \cdot W'_x) + \sum_{k=1}^{K} \log \sigma(-e^i \cdot W'_r_k).
\]

For DSAW, where

\[
s^i = \text{APPLY}(x^i, s^0), \\
s^{i-j} = \text{APPLY}(x^{i-j}, \text{APPLY}(\ldots \text{APPLY}(x^{i-c}, s^0) \ldots)), \\
s^{i+j} = \text{APPLY}(x^{i+j}, \text{APPLY}(\ldots \text{APPLY}(x^{i+1}, \text{APPLY}(x^{i-1}, \ldots \text{APPLY}(x^{i-c}, s^0))))),
\]

for \( 1 \leq j \leq c \), the total loss to maximize is:

\[
\log \sigma((s^{i+c} - \frac{1}{2}) \cdot (s^i - \frac{1}{2})) - \sum_{k=1}^{K} \log \sigma(-(s^{i+c} - \frac{1}{2}) \cdot (\text{APPLY}(s^k, s^0) - \frac{1}{2})) - \sum_{-c \leq j < c, j \notin \{0,1\}} \beta D_{KL}(q(s^{i+j} | x^{i+j}, s^{i+j-1}) || p(s^{i+j})) \\
- \beta D_{KL}(q(s^{i+1} | x^{i+1}, s^{i-1}) || p(s^{i+1})) \\
- \beta D_{KL}(q(s^i | x^i, s^0) || p(s^i))
\]
where \( p(s_i^t) = \text{Bernoulli}(0.5) \) for all \( i, j \), \( s^0 \sim \text{Bernoulli}(0.5) \), \( D_{KL}(q(\cdot)||p(\cdot)) \) is the KL divergence for each discrete variational layer and \( \beta \) is the scale factor as in \( \beta \)-VAE [40].

The temperature parameter \( \tau \) for the Binary Concrete at the epoch \( t \) \((0 \leq t \leq 8\), where \( t \) could be a fractional number, proportionally spread across the mini-batches) follows a stepped schedule below:

\[
\tau(t) = \begin{cases} 
5 & (0 \leq t < T) \\
5 \cdot \exp(\log \frac{0.7}{0.0} \cdot | \frac{t-T}{0.2} | \cdot 0.2) & (T < t \leq 8)
\end{cases}
\]

where \( T \) is a hyperparameter that determines when to start the annealing. \( \tau \) approaches 0.7 at the end of the training.

We performed a grid search in the following hyperparameter space: Embedding size \( V \in \{200, 500, 1000\} \), learning rate \( lr \in \{0.001, 0.003, 0.0001\} \), scaling factor \( \beta \in \{0.0, 0.1, 1.0\} \), annealing start epoch \( T \in \{1, 7\} \), and a boolean flag \( A \in \{\top, \bot\} \) that controls whether the Batch Normalization layers in BTL use the Affine transformation. We kept \( c = 2 \) words context window before and after the target words and the number of negative-samples \( K = 5 \) for all experiments. We initialize the weight matrix with Gaussian noise for CBOW, and with Logistic noise for DSAW, as we discuss in Sec. A.4.

### A.4 Additional experiments: Weight initialization with Logistic(0,1) distribution

In the original Word2Vec CBOW, the embedding weights are initialized by Uniform noise. Gaussian noise \( \mathcal{N}(0, 1) \) is also used in some studies [41, 57], and they show comparable results. The row \( W_x \) selected by the word index \( x \) is directly used as the continuous effects in each recurrent step. In contrast, DSAW applies BinConcrete in each step, which contains a squashing function (sigmoid) and a noise that follows Logistic distribution \( \text{LOGISTIC}(0, 1) \), which has a shape similar to Gaussian noise \( \mathcal{N}(0, 1) \) but has a fatter tail.

We hypothesized that the word effect \( W_x \) may fail to sufficiently affect the output values if its absolute value \(|W_x|\) is relatively small compared to the Logistic noise and is squashed by the activation. To address this issue, we initialized the embedding weights by \( \text{LOGISTIC}(0, 1) \). While the in-depth theoretical analysis is left for future work, this initialization helped the training of DSAW models in empirical evaluation. All results reported for DSAW in other places use this Logistic initialization.

Results in Table 4 shows that the DSAW models trained with Logistic weight initialization tend to outperform the DSAW trained with Gaussian weight initialization, which is the default initialization scheme for the embedding layers in PyTorch library.

| Embedding size \( E \) | 200 | 500 | 1000 |
|------------------------|-----|-----|------|
| Word Similarity        | 0.504 | 0.531 | 0.538 |
| Analogy Top1 acc.      | 0.222 | 0.332 | 0.373 |
| Analogy Top10 acc.     | 0.526 | 0.662 | 0.683 |
| Text Classification Test | 0.680 | 0.707 | 0.908 |

Table 4: Downstream task performance of DSAW models using Logistic vs. Gaussian weight initialization. The better initialization under the same hyperparameter set is highlighted in bold.

### A.5 Additional model experiments: Discrete implementation of SkipGram

In addition to the main model architecture studied in the main paper, we also explored two additional potential architectures: Skipgram and Skipgram-BTL. Word2Vec Skipgram (SG), is the other model architecture originally proposed by Mikolov et al. along with CBOW. Instead of using a set of context words to predict a target word like CBOW, Skipgram reverses the task: it attempts to predict the set of context words from the target word. To modify Skipgram model to include our discrete property, we pass the target word through Back-To-Logit [8] on one side, and pass each context word on the other side (individually), and calculate the loss on both sides. Effectively, context size is now reduced to one word and the model loses the recurrent nature of the DSAW architecture. Empirically, we find SG-BTL to perform worse than DSAW, possibly due to the lack of the recurrence. Table 5 shows the summary of the SG model and the SG-BTL model on the tasks we evaluated on.
### Table 5: Downstream task performance of SG and SG-BTL.

| Embedding size $E$ | Model | 200 | 500 | 1000 |
|-------------------|-------|-----|-----|------|
| Word Similarity   | SG    | 0.450 | 0.446 | 0.430 | 0.466 | 0.388 | 0.466 |
| Analogy Top1 acc. | SG-BTL | 0.312 | 0.121 | 0.266 | 0.203 | 0.198 | 0.233 |
| Analogy Top10 acc.| SG    | 0.545 | 0.355 | 0.507 | 0.463 | 0.421 | 0.520 |
| Text Classification Test | SG-BTL | 0.814 | 0.636 | 0.823 | 0.651 | 0.822 | 0.694 |

### Table 6: Schematic diagram of SG-BTL.

A.6 Additional model experiments: Hybrid discrete-continuous CBOW model

Hybrid discrete-continuous models are the second group of additional models we implemented. Instead of purely training a discrete or a continuous model, this architecture merges the two and trains both embeddings jointly. We experimented with this model because we hypothesized that there are ambiguous, continuous concepts that are hard to capture logically (e.g., temperature, emotion) as well as discrete, logical concepts (e.g., apple, mathematics) within the semantic space.

A hybrid model of embedding size $E$ contains a discrete embedding of size $\frac{E}{2}$ and a continuous embedding of size $\frac{E}{2}$. Two embeddings are concatenated together before the subsequent operations. For example, during the training, the loss is calculated by concatenating the continuous-bag-of-word representation $e$ and the shifted discrete output state $s^{i+c} = 0.5$, then applying the standard Word2Vec loss $\log \sigma(x \cdot y)$ [Mikolov et al.] between the target and predict embedding $x, y$. Similarly, for the vector addition/subtraction operations in the analogy task or the word aggregation in text classification, the two embeddings are treated with respective methods separately and concatenated in the end.

The evaluation results in Table 7 shows that the hybrid model performs somewhere in between CBOW and DSAW, except for analogy top 10 category, which outperforms the best performance of DSAW. This could have resulted from our crude way of aggregating the hybrid embedding by splitting the vector into two, performing the aggregation separately, then concatenating the results together. Future experiments call for more strategic fusing of two types of models which allow meaningful and effective word embedding aggregation.

| Embedding size $E$ | Model | 200 | 500 | 1000 |
|-------------------|-------|-----|-----|------|
| Word Similarity   | Hybrid | 0.444 | 0.498 | 0.492 |
| Analogy Top1 acc. | Hybrid | 0.136 | 0.283 | 0.370 |
| Analogy Top10 acc.| Hybrid | 0.377 | 0.596 | 0.689 |
| Text Classification Test | Hybrid | 0.836 | 0.858 | 0.849 |

Table 7: Overall performance of the Hybrid models.
A.7 Detailed, per-category results for the word similarity task

Word similarity task is the standard benchmark for measuring the attributional similarity [54, 64, 1]. 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 [68, 21, 56]. The scores for CBOW are obtained by the cosine similarity between two word vectors. For the DSAW embedding, the standard cosine distance is not directly applicable as each embedding consists of two binary vectors. We, therefore, turn the effect of a word \( x \) into an integer vector of tertiary values \( \{1, 0, -1\} \) by ADD(\( x \)) − DEL(\( x \)), then compute the cosine similarity.

We tested our models with the baseline models on 5 different datasets Bruni (MEN), Radinsky (MT), Luong rare-word (RW), Hill Sim999 (SM), and WS353 (WS) [12, 63, 49, 31, 24]. WS353 dataset is further separated into relatedness (WSR) and similarity (WSS) [1]. To illustrate the difference between relatedness and similarity, we use the example of “ice cream” and “spoon”. The two words are not similar but they are related in the sense that “spoon” is often used to consume “ice cream”. DSAW model outperforms CBOW model in all datasets except MT. The detailed, per-category results for this task can be found in Table 8.

| Data     | Number of word pairs | \( E = 200 \) | \( E = 500 \) | \( E = 1000 \) |
|----------|----------------------|---------------|---------------|----------------|
| WS       | 353                  | .540          | .506          | .478           | .568           |
| WSR      | 252                  | .474          | .472          | .503           | .475           | .518           |
| WSS      | 203                  | .641          | .580          | .652           | .594           | .680           |
| MT       | 287                  | .617          | .609          | .599           | .589           | .561           |
| MEN      | 3000                 | .692          | .691          | .696           | .667           | .710           |
| RW       | 2034                 | .359          | .341          | .394           | .302           | .377           |
| SM       | 999                  | .340          | .344          | .356           | .304           | .359           |
| Total    | 7128                 | .528          | .518          | .538           | .488           | .545           |

Table 8: Word similarity results compared by the datasets (CBOW and DSAW).

A.8 Detailed, per-category results for the analogy task

In addition to the overall accuracy in the analogy task, we break down the performance of different models into the sub-categories in the dataset provided by Mikolov et al. [52]. In the ADD column of Table 9 we show the per-category accuracy of the CBOW and DSAW models that achieved the best overall accuracy as a result of hyperparameter tuning. CBOW uses the vector addition \( a^* - a + b \) for the nearest neighbor, and DSAW uses the STRIPS progression \( \hat{a} + a^{*} + b \). DSAW performs similarly to, if not better than CBOW, in different analogy categories. Specifically, DSAW performs significantly better than CBOW in “capital-world”, “gram2-opposite”, “gram5-present-participle”, “gram7-past-tense”, and “gram9-plural-verbs”.

We also show the results of Ignore-A and Only-B aggregation scheme [45, 59, 20] compared to ADD scheme. Compared to the ADD column in Table 9 which uses all three input words for the analogy (e.g., \( a^* - a + b \)), Ignore-A does not use \( a \) (e.g., \( a^* + b \)), and Only-B uses \( b \) unmodified, and searches for the nearest neighbor. The intention behind testing these variants is to see if the analogy performance is truly coming from the differential vector \( (a^* - a) \), or just from the neighborhood structure of the target word \( b \) and \( a^* \). A good representation with nice vector-space property is deemed to have the performance ordering ADD > Ignore-A > Only-B. Our model indeed tends to have this property, as can be seen in the plot (Fig. 5), confirming the validity to our approach.

A.9 Exploring the best ordering of the discrete additive operations in analogy task

Because the proposed bit-wise operations \( \hat{+}, \hat{-} \) are not associative or commutable, we evaluated the effect of the orders of operations used while performing the analogy task. As seen in Table 10 the different order of operations significantly affects the results. The performance of different ordering was consistent across the different hyperparameters. In the main paper, we reported the best-performing ordering, \( \hat{-} a + a^* + b \).
Figure 5: Scatter plot of the best analogy accuracies with CBOW (x-axis) and DSAW (y-axis) for each dataset category. Each path indicates the performance change caused by the different analogy method ADD, Ignore-A, Only-B. The path tends to move toward top-right, indicating that both embeddings are utilizing the differential information in the vectors $a$ and $a^*$ for analogy, not just the neighborhood structure of $b$ and $a^*$. 
Table 9: Word Analogy accuracies compared by each category. Data from the best performing CBOW and DSAW models with the embedding size 1000.

Table 10: Word Analogy accuracies compared by each category, comparing the effect of the different ordering of $\hat{+}, \hat{-}$ operations in DSAW. Data from the best performing DSAW model with the embedding size 1000.

A.10 Detailed, per-category results for the text classification task

We used our embeddings for the semantic text classification, in which the model must capture the semantic information to perform well. We evaluated our model on two datasets: "20 Newsgroup" [44] and "movie sentiment treebank" [72]. We created binary classification tasks following the existing work [76, 82]. For 20 Newsgroup, we picked 4 sets of 2 groups to produce 4 sets of classification problems: SCI (science.med vs. science.space), COMP (ibm.pc.hardware vs. mac.hardware), SPORT (baseball vs. hockey), RELI (alt.atheism vs. soc.religion.christian). For movie sentiment (MS), we ignored the neutral comments and set a threshold for the sentiment values: $\leq 0.4$ as 0, and $> 0.6$ as 1. In all 20-newsgroup datasets, we split the corpus into train, validation, test set by proportion 0.48, 0.12, and 0.40 [82]. The movie sentiment dataset, after removing all neutral reviews (about 20% of the original data), is then split into train, validation, and test sets by proportion 0.72, 0.09, and 0.19 [82].

In both the CBOW and the DSAW models, 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 normalize the imbalance in the number of questions between
subtasks (SCI, ...RELI have \( \approx 2000 \) questions each while MS has \( \approx 9000 \)), that is, the total accuracy is unweighted average of the accuracies over the 5 datasets. This is in order to account for the imbalance in the number of classification inputs in each dataset.

As seen in Table 11 our method performs better than the traditional CBOW on three 20 Newsgroup datasets, comparably on RELI, and less ideally on Movie sentiments. We interpreted this result as follows: This is caused by the ability of DSAW embedding to preserve the embedded value of the rare, key terms in the document during the aggregation. Imagine if a continuous embedding of a rare word \( x \) has a dimension whose absolute value is significantly large. However, all other words in the sentence have a varying degree of noisy values in the same dimension, which accumulates during the aggregation and cause the value to deviate from the original value in the rare word, essentially "blurring" the significance of that word. In contrast, discrete representations obtained by DSAW have the three, clear-cut modes of operations – add, delete, or no-op on specific bits. Therefore, the effects from unrelated words, which are frequently no-op (see Sec. B.4), tend not to affect the value of the important bit in the rare word. This characteristics would be less prominent if the length of the sequence is short (MS), or if the important key words used for classifying the sentence are used frequently enough in the document that it is not obscured by other common words, which we subjectively observed in the RELI dataset. This interpretation also matches the better performance of DSAW on the RW (rare word) dataset in the word similarity task.

| Data (Num. documents) | Sentence len. (Num. documents) | 200 | 500 | 1000 |
|-----------------------|-------------------------------|-----|-----|------|
| SCI (1994)            | 276 229                       | .976| .952| .983 |
| COMP (1981)           | 341 255                       | .809| .938| .892 |
| SPORT (1987)          | 383 269                       | .902| .941| .952 |
| RELI (1995)           | 447 324                       | .996| .976| \textbf{.999} |
| MS (9142)             | 17 17                         | .770| .629| .773 |
| Total (17099)         | 17 17                         | .890| .867| .920 |

Table 11: Per-category accuracies for the text classification task. We observed that the continuous embeddings performs well in MS, which has shorter sentences, and relatively worse in longer sentences, except RELI.

A.11 Additional visualizations for the word compositionality

In this section, we show more examples of word compositionality visualization. An example of such a visualization would contain “Long thin solid cylindrical pasta = spaghetti”, where the left-hand-side is a compositional phrase and the right-hand-side is a target word. Our aim is to showcase that the given phrase should aggregate to the respective target word in the embedding space. Additionally, as more adjectives are added to the phrase, the resulting phrase embedding should approach the target word (i.e. “cylindrical pasta” – “spaghetti” > “solid cylindrical pasta” – “spaghetti”). For each phrase, we added adjectives incrementally to obtain each partial phrase embedding through aggregating the word embeddings. We then plotted these embeddings, along with a few of their neighboring words and target words, using Principle Component Analysis (PCA) to show relationships in the embedding space. For all examples, we come up with the compositional phrase of each target word inspired by the opening sentences of the corresponding Wikipedia article, which often contains the definition of the target word.

In Fig. 6, we plot compositional phrase: Long thin solid cylindrical pasta = spaghetti. In both CBOW and DSAW plot, we can clearly see two clusters of words. In CBOW, the top cluster (and the right cluster in DSAW) includes words related to food and cuisine, and the other cluster in respective plots includes words associated with the adjectives “long”, “thin”, “solid”, “cylindrical”. We can see that under bit-operation, discrete embedding kept the phrase embedding close to the food cluster while continuous embedding caused the phrase embedding to wonder around different embedding space.

In Fig. 7, we plot compositional phrase: Adult male cattle = ox. In the continuous embedding, the phrase is dragged by the “human” aspect of “adult” and “male”, resulting in the bottom cluster containing the words related to humans, e.g. to “white” (presumably race), “babies”, “inmates”, “girls”. In the discrete embedding, both vectors resides in the spread-out cluster containing farm (pasture, poultry), animals (chimpanzee, elephants, pig), and foods (beef, steak, patties).

In Fig. 8, we plot Quadrupedal ruminant mammal = sheep. The phrase vector in both embeddings appears to roughly share the direction with “sheep”.

22
In Fig. 9 we plot compositional phrase: Italian luxury sports car manufacturer = Ferrari. In this case, discrete embedding fails to share the direction with the intended word “ferrari” while continuous embedding roughly succeeds.
Figure 6: Plotting the compositional phrases with CBOW (left) and DSAW (right): Long thin solid cylindrical pasta = spaghetti, from https://en.wikipedia.org/wiki/Spaghetti.

Figure 7: Plotting the compositional phrases with CBOW (left) and DSAW (right): Adult male cattle = ox, from https://en.wikipedia.org/wiki/Ox.

Figure 8: Plotting the compositional phrases with CBOW (left) and DSAW (right): Quadrupedal ruminant mammal = sheep, from https://en.wikipedia.org/wiki/Sheep.

Figure 9: Plotting the compositional phrases with CBOW (left) and DSAW (right): Italian luxury sports car manufacturer = Ferrari, from https://en.wikipedia.org/wiki/Ferrari.
B Planning / paraphrasing experiments

B.1 The archive directory paraphrasing/

The accompanied data dump in paraphrasing/ contains the sample domain PDDL file (paraphrasing/domain_soft_0_4000.pddl) for embedding size 200, and the problem files, log files and the plan files found in each experiment in paraphrasing/target_words_examples-1-100/ directory.

This data dump includes the results from other planning configurations and the embedding size. The secondary planning configuration uses FF-Eager-Iterative, an iterative variant of FF planner [33] (reimplementation in Fast Downward [29]) that first performs Greedy Best First Search, then continues refining the solution with Weighted $A^*$ with decreasing weights \{10, 5, 3, 2, 1\}.

B.2 Compilation of a net-benefit planning problem into a classical planning problem

A net-benefit planning task $\langle P, A, I, G, c, u \rangle$ can be compiled into a classical planning problem with action cost $\langle P', A', I', G', c' \rangle$ as follows [38]. We use the slightly different notation from [38] by assuming a negative precondition extension [28] and by assuming all goals are soft:

$$P' = P \cup \{\text{end-mode}\} \cup \{\text{marked}(p) \mid p \in G\}$$
$$A' = A'' \cup \{\text{end}\} \cup \{\text{collect}(p), \text{forgo}(p) \mid p \in G\}$$
$$A'' = \{(\text{PRE}(a) \land \neg\text{end-mode}, \text{ADD}(a), \text{DEL}(a)) \mid a \in A\}$$

$$\forall a'' \in A''; c'(a'') = c(a)$$
$$\text{collect}(p) = (\text{end-mode} \land p \land \neg\text{marked}(p), \text{marked}(p), \emptyset)$$
$$c'(\text{collect}(p)) = 0$$
$$\text{forgo}(p) = (\text{end-mode} \land \neg p \land \neg\text{marked}(p), \text{marked}(p), \emptyset)$$
$$c'(\text{forgo}(p)) = u(p)$$
$$\text{end} = (\neg\text{end-mode}, \text{end-mode}, \emptyset)$$

$$c'(\text{end}) = 0$$
$$G' = \{\text{marked}(p) \mid p \in G\}.$$ 

Also, as mentioned in [38], we add additional preconditions to collect, forgo that linearize the ordering between the actions, i.e., for $i > 0$, $\text{PRE} (\text{collect}(p_i)) = \text{end-mode} \land p_i \land \neg\text{marked}(p_i) \land \text{marked}(p_i - 1)$. (Same for forgo.)

As a paraphrasing-specific enhancement, we further add the constraint that forces to avoid using the same word twice. That is, $\forall a'' \in A''; \text{PRE}(a'') \ni \neg \text{used}(a'')$; $\text{ADD}(a'') \ni \text{used}(a'')$, where $\text{used}(a'')$ is added to $P'$ for all $a''$.

B.3 The list of 68 target words used in the paraphrasing experiment

In the paraphrasing experiments, we hand-picked the words listed in Fig. 10 and generate the paraphrasing planning problem.

| lamborghini | ferrari | maserati | fiat | renault | bmw | mercedes | audi | toyota |
|-------------|---------|----------|------|---------|-----|----------|------|--------|
| honda | mazda | nissan | subaru | ford | chevrolet | suzuki | kawasaki | ducati |
| yamaha | king | queen | prince | princess | sea | lake | river | pond | island | mountain |
| hill | valley | forest | woods | apple | grape | orange | muscat | potato | carrot | onion |
| garlic | pepper | cumin | oregano | wine | sake | coke | pepsi | water | meat | steak |
| hamburger | salad | sushi | grill | spaghetti | noodle | ramen | run | flee | escape | jump |
| dance | wave | speak | yell | murmur | shout |

Figure 10: The list of words used for the paraphrasing experiments.

B.4 The statistics of the discrete effect vectors

Schakel and Wilson [71, 29] discussed the relationship between the word frequency and the magnitude (length) of the continuous word embedding vectors. In contrast, DSAW embedding consists of two vec-
tors, \(\text{ADD}(x)\) and \(\text{DEL}(x)\), and each dimension in the embedding is restricted to the binary values \(\{0, 1\}\). In order to understand the behavior of our discrete embedding, we visualized the density of the effect presence, i.e., \(\frac{1}{E} \sum_{j=1}^{E} \text{ADD}(x)_j\) and \(\frac{1}{E} \sum_{j=1}^{E} \text{DEL}(x)_j\). We plotted these statistics for each word \(x\) in the order of frequency.

In Fig. 11, we observe that rare words tend to have more effects. This matches our intuitive understanding of the meaning of the rare, complex words: Complex words tend to be explained by or constructed from the simpler, more basic words. This may also be suggesting why the paraphrasing task works well: The planner is able to compose simpler words to explain the more complex word because of this characteristics.

### B.5 Runtime statistics for the word paraphrasing experiment

We visualized the runtime statistics of the paraphrasing task. Fig. 12 (left) shows the cumulative plots of the number of solutions found at a certain point of time, over all target words / problem instances used in the experiment. \(x\)-axis plots the runtime, and \(y\)-axis plots the number of solutions found. We plotted the results obtained by the embedding size \(E = 200\), soft-goal cost \(U = 100\) and the LAMA planner. The plot shows that the first solutions are obtained relatively quickly and more solutions are found later due to the iterative, anytime planning behavior of LAMA.

Moreover, in Fig. 12 (right), we show the “actual search time” which excludes the time for parsing, preprocessing and datastructure setup for the heuristic search. This shows that the majority of the time was spent on just reading the large PDDL file that was produced from the embedding vector of 4000 words. On a practical, long-running system with an appropriate caching mechanism, this bottleneck can be largely amortized.

In Fig. 13, we also show the cumulative plots restricted to the solutions with the length larger than 2, because a solution with the length 1 in the net-benefit planning problem is equivalent to merely finding a nearest neighbor word in L1 distance, rather than finding a phrase.

### B.6 Additional paraphrasing for a set of randomly selected 300 words

Finally, we performed further experiments with an additional set of 300 words randomly selected from the 4000th to the 8000th most frequent words in the vocabulary. This additional set includes more proper nouns, whose paraphrasing tends to be meaningless. However, we discover even more new examples that are interesting. The paraphrasing examples can be found in Table 12-13. The additional data dump can be found in the "paraphrasing/target_words_4000_8000-1-100" directory.
Figure 11: The density of add/delete effects (left, right) in the best DSAW model trained with $E = 200$, where $x$-axis is the word index sorted according to the frequency (frequent words are assigned the smaller indices), cut off at 32000-th word. We observe that rare words tend to have more effects.

Figure 12: (left) Cumulative plot of the number of solutions found at the total time $t$, and (right) the same statistics based on the actual search time, i.e., the runtime excluding the time for the input parsing and initialization.

Figure 13: The same plot as Table [12] but the length is restricted to be larger than 2.
| Word $y$                                      | word sequence $\pi$ (solution plan)                        |
|----------------------------------------------|----------------------------------------------------------|
| adventure                                    | classic trip; drama movie                                |
| amateur                                      | professional maybe                                       |
| anxiety                                      | uncertainty stress                                       |
| appreciate                                   | listen understand                                        |
| ballet                                       | theatre dance                                            |
| bipartisan                                   | support proposal                                         |
| bold                                         | fresh simple move                                        |
| cake                                         | birthday eat                                             |
| cancel                                       | continue delay                                           |
| cholesterol                                 | blood bad                                                |
| cigarette                                    | smoke alcohol ; tax smoke                                 |
| compliance                                  | risk ensure                                              |
| concent                                      | approval written ; written approval prior; formal knowlege|
| corrupt                                      | regime good sick act                                     |
| deck                                         | floor roof ; roof floor                                  |
| deserve                                      | want ensure ; accept know                                 |
| disappear                                    | presense soon                                            |
| disciplinary                                 | action legal                                             |
| distress                                     | emotional shock                                          |
| dominant                                     | position china                                           |
| explore                                      | continue enjoy                                           |
| fantasy                                      | dream novel                                              |
| grip                                         | tight presence                                           |
| hint                                         | evidence listen                                          |
| identification                               | photo formal identity ; identity card                    |
| immune                                       | system response                                          |
| innocent                                     | ordinary woman                                           |
| interference                                 | penalty conduct                                          |
| interrogation                               | cia torture                                              |
| intervention                                 | necessary plan                                           |
| isolation                                    | cuba situation                                           |
| jazz                                         | music band; music song band                              |
| kremlin                                      | moscow claim ; pro putin                                  |
| laptop                                       | personal mac device                                      |
| learnt                                       | learn yesterday                                          |
| lesson                                       | history addition                                         |
| liquidity                                    | boost cash ; guarantee cash                              |
| lobby                                        | pro group ; group pro                                    |
| louisville                                   | kentucky pittsburgh; kentucky                             |
| nutrition                                    | medicine food                                            |
| offence                                      | criminal cause ; criminal sign                           |
| passport                                     | card identity ; account identity                         |

Table 12: (Part 1) Paraphrasing of the source words returned by the LAMA planner. 300 source words are randomly selected from the 4000th to 8000th most frequent words. Note¹: Louisville, Cleveland and Pittsburgh are the central city of Kentucky, Ohio and Pensilvania, respectively.
| Word $y$ | word sequence $\pi$ (solution plan) |
|----------|-----------------------------------|
| passage  | secure route; easy congress route; final passage advance; win safe |
| phrase   | word theory; theory word |
| plain    | english simple; english nice combination; english look pretty nice; english typical just stuff |
| plea     | guilty deal |
| pleasure | enjoy brought; ride great |
| poet     | artist author; author born artist |
| prosperity | stability peace; yield stability; wealth stability; |
| puerto   | taiwan argentina |
| pump     | oil blood put |
| railway  | rail train; line train |
| reactor  | nuclear plant; nuclear uranium plant |
| reconstruction | infrastructure recovery effort |
| referendum | vote hold; independence hold |
| restoration | restore project |
| resume   | continue begin; begin continue |
| robust   | weak strong |
| rough    | ride tough; wild difficult |
| rubbish  | collection waste; bin waste |
| sample   | survey evidence blood dna |
| sectarian | ethnic violence |
| showdown | controversy ahead; final battle |
| slight   | steady slow substantial |
| slot     | wish machine |
| spacecraft | nasa flew |
| subsidiary | unit corporation |
| successor | departure replace |
| surgon   | surgery resident plastic doctor; surgery specialist; plastic doctor |
| sustain  | maintain continue |
| swap     | debt exchange listen deal agree; debt exchange; currency buy |
| teach    | children learn |
| throat   | breast mouth; neck mouth |
| transit  | transportation system; mass transportation |
| trash    | waste bin |
| uncertain | unclear future; unclear confident |
| unfair   | advantage competition |
| unity    | sort coalition; national democracy |
| yuan     | yen china dollar |

Table 13: (Part 2) Paraphrasing of the source words returned by the LAMA planner. 300 source words are randomly selected from the 4000th to 8000th most frequent words. Note$^2$: Puerto Rico and Taiwan are both island countries; Puerto Rico and Argentina both speak Spanish.
References

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, page 19, 2009.

[2] Vidal Alcázar, Daniel Borrajo, Susana Fernández, and Raquel Fuentetaja. Revisiting Regression in Planning. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2013.

[3] Leonardo Amado, Ramon Fraga Pereira, Joao Aires, Mauricio Magnaguagno, Roger Granada, and Felipe Meneguzzi. Goal Recognition in Latent Space. In Proc. of International Joint Conference on Neural Networks (IJCNN), 2018.

[4] Leonardo Amado, Ramon Fraga Pereira, Joao Aires, Mauricio Magnaguagno, Roger Granada, and Felipe Meneguzzi. LSTM-based Goal Recognition in Latent Space. arXiv preprint arXiv:1808.05249, 2018.

[5] Masataro Asai. Unsupervised Grounding of Plannable First-Order Logic Representation from Images. In Proc. of the International Conference on Automated Planning and Scheduling (ICAPS), 2019.

[6] Masataro Asai and Alex Fukunaga. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In Proc. of AAAI Conference on Artificial Intelligence, 2018.

[7] Masataro Asai and Hiroshi Kajino. Towards Stable Symbol Grounding with Zero-Suppressed State AutoEncoder. In Proc. of the International Conference on Automated Planning and Scheduling (ICAPS), 2019.

[8] Masataro Asai and Christian Muise. Learning Neural-Symbolic Descriptive Planning Models via Cube-Space Priors: The Voyage Home (to STRIPS). In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2020.

[9] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, pages 178–186, 2013.

[10] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligram, and Adam Tauman Kalai. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In Advances in Neural Information Processing Systems, pages 4349–4357, 2016.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Arie Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. CoRR, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

[12] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal Distributional Semantics. J. Artif. Intell. Res.(JAIR), 49:1–47, 2014.

[13] Tom Bylander. The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence, 69(1):165–204, 1994.

[14] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334):183–186, 2017.

[15] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Philipp Koehn, and Tony Robinson. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In Proc. of Annual Conference of the International Speech Communication Association, 2014.

[16] Ting Chen, Martin Renqiang Min, and Yizhou Sun. Learning K-way D-dimensional Discrete Codes for Compact Embedding Representations. In Proc. of the International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 853–862, 2018.
[17] Kyunghyun Cho, Bart van Merrienboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proc. of Conference on Empirical Methods in Natural Language Processing, pages 1724–1734, 2014.

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

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4171–4186, 2019.

[20] Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. Word Embeddings, Analogies, and Machine Learning: Beyond King-Man+Woman=Queen. In Proc. of the International Conference on Computational Linguistics, pages 3519–3530, 2016.

[21] Manaal Faruqui and Chris Dyer. Community Evaluation and Exchange of Word Vectors at Word-vectors.org. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 19–24, 2014.

[22] Wenfeng Feng, Hankz Hankui Zhuo, and Subbarao Kambhampati. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2018.

[23] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and Executing Generalized Robot Plans. Artificial Intelligence, 3(1-3):251–288, 1972.

[24] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing Search in Context: The Concept Revisited. In Proc. of the International Conference on World Wide Web, pages 406–414, 2001.

[25] Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 1426–1436, 2014.

[26] Christopher W. Geib and Mark Steedman. On Natural Language Processing and Plan Recognition. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), pages 1612–1617, 2007.

[27] Malik Ghallab, Dana Nau, and Paolo Traverso. Automated Planning: Theory and Practice. Elsevier, 2004.

[28] Patrik Haslum, Nir Lifshitz, Daniele Magazzeni, and Christian Muise. An Introduction to the Planning Domain Definition Language. Synthesis Lectures on Artificial Intelligence and Machine Learning, 13(2):1–187, 2019.

[29] Malte Helmert. The Fast Downward Planning System. J. Artif. Intell. Res.(JAIR), 26:191–246, 2006.

[30] Irina Higgins, Loïc Matthey, Arka Pal, et al. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In Proc. of the International Conference on Learning Representations, 2017.

[31] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. Computational Linguistics, 41(4):665–695, 2015.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.

[33] Jörg Hoffmann and Bernhard Nebel. The FF Planning System: Fast Plan Generation through Heuristic Search. J. Artif. Intell. Res.(JAIR), 14:253–302, 2001.

[34] Chad Hogg, Héctor Muñoz-Avila, and Ugur Kuter. Learning Hierarchical Task Models from Input Traces. Comput. Intell., 32(1):3–48, 2016. doi: 10.1111/coin.12044. URL https://doi.org/10.1111/coin.12044.
[35] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proc. of the International Conference on Machine Learning, pages 448–456, 2015.

[36] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In Proc. of the International Conference on Learning Representations, 2017.

[37] Robin Jia and Percy Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. In Proc. of Conference on Empirical Methods in Natural Language Processing, pages 2021–2031, 2017.

[38] Emil Keyder and Hector Geffner. Soft goals can be compiled away. J. Artif. Intell. Res.(JAIR), 36: 547–556, 2009. doi: 10.1613/jair.2857. URL https://doi.org/10.1613/jair.2857.

[39] Yoon Kim, Chris Dyer, and Alexander M Rush. Compound Probabilistic Context-Free Grammars for Grammar Induction. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 2369–2385, 2019.

[40] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In Proc. of the International Conference on Learning Representations, 2013.

[41] Tom Kocmi and Ondrej Bojar. An Exploration of Word Embedding Initialization in Deep-Learning Tasks. In Proceedings of the 14th International Conference on Natural Language Processing, ICON, pages 56–64, 2017.

[42] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart Russell, and Pieter Abbeel. Learning Plannable Representations with Causal InfoGAN. In Advances in Neural Information Processing Systems, 2018.

[43] Kevin Lacker. Giving GPT-3 a Turing Test, 2020. URL https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html.

[44] Ken Lang. Newsweeder: Learning to Filter Netnews. In Proc. of the International Conference on Machine Learning, pages 331–339, 1995.

[45] Omer Levy and Yoav Goldberg. Linguistic Regularities in Sparse and Explicit Word Representations. In Proc. of Conference on Computational Natural Language Learning, pages 171–180, 2014.

[46] Alan Lindsay, Jonathon Read, Joao F Ferreira, Thomas Hayton, Julie Porteous, and Peter J Gregory. Framer: Planning Models from Natural Language Action Descriptions. In Proc. of the International Conference on Automated Planning and Scheduling(ICAPS), 2017.

[47] Liyuan Liu, Haoming Jiang, Pengcheng He, et al. On the Variance of the Adaptive Learning Rate and Beyond. arXiv:1908.03265, 2019.

[48] Carlos Linares López, Sergio Jiménez Celorrio, and Ángel García Olaya. The Deterministic Part of the Seventh International Planning Competition. Artificial Intelligence, 223:82–119, 2015.

[49] Minh-Thang Luong, Richard Socher, and Christopher D. Manning. Better Word Representations with Recursive Neural Networks for Morphology. In Proc. of Conference on Computational Natural Language Learning, 2013.

[50] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In Proc. of the International Conference on Learning Representations, 2017.

[51] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proc. of the International Conference on Learning Representations, 2013.

[52] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionalality. In Advances in Neural Information Processing Systems, pages 3111–3119, 2013.

[53] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 746–751, 2013.
[74] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The Computational Limits of Deep Learning. *CoRR*, abs/2007.05558, 2020. URL https://arxiv.org/abs/2007.05558.

[75] Geoffrey G Towell and Jude W Shavlik. Knowledge-Based Artificial Neural Networks. *Artificial Intelligence*, 70(1-2):119–165, 1994.

[76] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of Word Vector Representations by Subspace Alignment. In *Proc. of Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, 2015.

[77] Chuan Wang, Nianwen Xue, and Sameer Pradhan. A Transition-based Algorithm for AMR Parsing. In *Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, 2015.

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

[79] Benjamin J. Wilson and Adriaan M. J. Schakel. Controlled Experiments for Word Embeddings. *CoRR*, abs/1510.02675, 2015.

[80] Ludwig Wittgenstein. *Tractatus Logico Philosophicus*. 1922.

[81] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *Proc. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, 2015.

[82] Dani Yogatama and Noah A Smith. Linguistic Structured Sparsity in Text Categorization. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 786–796, 2014.

[83] Tiancheng Zhao, Kyusong Lee, and Maxine Eskénazi. Unsupervised Discrete Sentence Representation Learning for Interpretable Neural Dialog Generation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 1098–1107, 2018.

[84] Hankz Hankui Zhuo, Héctor Muñoz-Avila, and Qiang Yang. Learning Hierarchical Task Network Domains from Partially Observed Plan Traces. *Artif. Intell.*, 212:134–157, 2014. doi: 10.1016/j.artint.2014.04.003. URL https://doi.org/10.1016/j.artint.2014.04.003.