DISCRETE WORD EMBEDDING FOR LOGICAL NATURAL LANGUAGE UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose an unsupervised neural model for learning a discrete embedding of words. Unlike existing discrete embeddings, our binary embedding supports vector arithmetic operations similar to continuous embeddings. Our embedding represents each word as a set of propositional statements describing a transition rule in classical/STRIPS planning formalism. This makes the embedding directly compatible with symbolic, state of the art classical planning solvers.

1 INTRODUCTION

When we researchers write a manuscript for a conference submission, we do not merely follow the probability distribution crystalized in our brain cells. Instead, we modify, erase, rewrite sentences over and over, while only occasionally let the fingers produce a long stream of thought. This writing process tends to be a zero-shot attempt to materialize and optimize a novel work that has never been written. Except for casual writing (e.g. online messages), *intelligent writing* inevitably contains an aspect of *backtracking* and *heuristic search* behavior, and thus is often like *planning for information delivery* while optimizing various metrics, such as the impact, ease of reading, or conciseness.

After the initial success of the distributed word representation in Word2Vec (Mikolov et al., 2013b), natural language processing techniques have achieved tremendous progress in the last decade, propelled primarily by the advancement in data-driven machine learning approaches based on neural networks. However, these purely data-driven approaches that blindly follow the highest probability interpolated from data at each time step could suffer from biased decision making (Caliskan et al., 2017; Bolukbasi et al., 2016) and is heavily criticized recently.

Meanwhile, in recent years, significant progress has been made (Asai & Fukunaga, 2018; Kurutach et al., 2018; Amado et al., 2018a;b; Asai & Muise, 2020) in the field of Automated Planning on resolving the so-called *Knowledge Acquisition Bottleneck* (Cullen & Bryman, 1988), the common cost of human involvement in converting real-world problems into the inputs for symbolic AI systems. Given a set of noisy visual transitions in fully observable puzzle environments, they can extract a set of latent *propositional* symbols and latent *action* symbols entirely without human supervision. Each action symbol maps to a description of the propositional transition rule in STRIPS classical planning (Fikes et al., 1972; Haslum et al., 2019) formalism that can be directly fed to the optimized implementations of the off-the-shelf state-of-the-art classical planning solvers.

To answer the high-level question of whether a zero-shot sentence generation is a planning-like symbolic processing, we focus on the most basic form of language models, i.e., word embedding. Building on the work on word embedding and STRIPS action model learning, we propose a discrete, propositional word embedding directly compatible with symbolic, classical planning solvers. We demonstrate its zero-shot unsupervised phrase generation using classical planners, where the task is to compose a phrase that has the similar meaning as the target word.

2 PRELIMINARY AND BACKGROUND

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

We assume background knowledge of discrete VAEs with continuous relaxations (See appendix Sec. A.1), such as Gumbel-Softmax (GS) and Binary-Concrete (BC) (Jang et al., 2017; Maddison et al., 2017). Their activations are denoted as GS and BC, respectively.

Word2Vec Continuous Bag of Word (CBOW) with Negative Sampling. The CBOW with Negative Sampling (Mikolov et al., 2013a;b) 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 $\langle x^{i-c}, \ldots, x^{i+c} \rangle$ ($x^i \in 1..V$) in a dataset $\mathcal{X} = \{x^i\}$, it computes the continuous-bag-of-words representation $\boldsymbol{e}^i = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} W_{x^{i+j}}$. 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}, sample K words $\{r^k\}$ ($k \in 1..K$) over V, extracts their embeddings W'_{r^k}, then maximizes the loss: $\log \sigma(\boldsymbol{e}^i \cdot W'_{x^i}) + \sum_{k=1}^{K} \log \sigma(-\boldsymbol{e}^i \cdot W'_{r^k})$.

Classical Planning. Classical Planning is a formalism for deterministic, fully-observable high-level sequential decision making problems. High-level decision making deals with a logical chunk of actions (e.g. opening a door) rather than low-level motor actuations, thus is considered fundamental to intelligence and has been actively studied since the early history of AI. Its input is encoded in a modeling language called Planning Domain Description Language (PDDL), which contains extensions from its most basic variant STRIPS.

A grounded (propositional) unit-cost STRIPS Planning problem (Fikes et al., 1972; Haslum et al., 2019) is defined as a 4-tuple $\langle P, A, I, G \rangle$ where P is a finite set of propositions, A is a finite set of actions, $I \subseteq P$ is an initial state, and $G \subseteq P$ is a goal condition. Here, a *state* is represented by a set of propositions $s \subseteq P$, where each $p \in s$ corresponds to the proposition whose truth value is $\top$, thus can be interpreted conjunctively, i.e, a set $\{p_1, p_2\}$ represents $p_1 \wedge p_2$. Each state $s \subseteq P$ can also be encoded as a bit vector $\boldsymbol{s} \in \{0,1\}^{|P|}$ where, for each j-th proposition $p_j \in P$, $\boldsymbol{s}_j = 1$ when $p_j \in s$, and $\boldsymbol{s}_j = 0$ when $p_j \notin s$. The entire set of states expressible in P is a power set 2^P.

While the propositional representation provides a syntax for denoting the environment, actions provides the rules for the time evolution, which plays a role similar to those of semantic and grammatical rules. Each action $a \in A$ is a 3-tuple $\langle \text{PRE}(a), \text{ADD}(a), \text{DEL}(a) \rangle$ where $\text{PRE}(a), \text{ADD}(a), \text{DEL}(a) \subseteq P$ are preconditions, add-effects, and delete-effects, respectively. Without loss of generality $\text{ADD}(a) \cap \text{DEL}(a) = \emptyset$. 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* $s' = a(s) = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a)$. A solution to a classical planning problem is called a *plan*, which is a sequence of actions $\pi = \langle a_1, a_2, \ldots a_{|\pi|} \rangle$ that leads to a terminal state $s^* = a_{|\pi|} \circ \ldots \circ a_1(s)$ that satisfies the goal condition, i.e., $G \subseteq s^*$. *Optimal* plans are those whose lengths are the smallest among possible plans.

STRIPS action modeling with neural networks. Cube-Space AutoEncoder (Asai & Muise, 2020) proposed a method for learning a binary latent representation s of visual time-series data while guaranteeing that every state transition in the latent representation can be expressed in STRIPS action rule $s_{t+1} = a(s_t) = (s_t \setminus \text{DEL}(a)) \cup \text{ADD}(a)$ for some action a. Therefore, it is able to encode raw inputs (time-series data) into a state and an action representation compatible with STRIPS planners.

Cube-Space AE does so by using a unique architecture called Back-to-Logit (BTL) that regularizes the state transitions. Since directly regularizing the discrete dynamics proved to be difficult, BTL performs all latent dynamics operations in the continuous space and discretizes the results as follows: It converts a given discrete state vector into a continuous vector using Batch Normalization (BN) (Ioffe & Szegedy, 2015), takes the continuous sum with an *effect* embedding of an action, and discretizes the sum (logit) using Binary Concrete. Formally, given an action a^i, its embedding $\text{EFFECT}(a^i)$ and a binary vector $\boldsymbol{s}^i$, the next state $\boldsymbol{s'}^i$ is predicted by:

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

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

Theorem 1 (Asai & Muise (2020)). *Under the same action a, state transitions are bitwise monotonic, deterministic, and restricted to three mutually exclusive modes. For each bit j:*

$$(\textit{add:})\ \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) \in \{(0,1),(1,1)\}\ \textit{i.e.}\ \boldsymbol{s}^i_j \leq \boldsymbol{s}'^i_j$$
$$(\textit{del:})\ \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) \in \{(1,0),(0,0)\}\ \textit{i.e.}\ \boldsymbol{s}^i_j \geq \boldsymbol{s}'^i_j$$
$$(\textit{nop:})\ \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) \in \{(0,0),(1,1)\}\ \textit{i.e.}\ \boldsymbol{s}^i_j = \boldsymbol{s}'^i_j.$$

It guarantees that each action deterministically turns a certain bit on and off in the binary latent space, thus the resulting action theory and the bit-vector representation satisfies the STRIPS state transition rule $s' = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a)$ and a constraint $\text{DEL}(a) \cap \text{ADD}(a) = \emptyset$. (Proof is straightforward from the monotonicity of BC and BN – See Appendix Sec. A.2.)

3 Zero-Shot Sequence Generation as Planning

To establish the connection between planning and zero-shot sequence generation, we first show the equivalence of classical planning and the basic right-regular formal grammar by mutual compilation. A formal grammar is a 4-tuple $\langle N, \Sigma, R, S\rangle$ which consists of non-terminal and terminal symbols N, Σ, production rules R, and a start symbol $S \in N$. A zero-shot sequence generation can be seen as a problem of producing a string of terminal symbols by iteratively applying one production rule $r \in R$ at a time to expand the current sequence, starting from the start symbol S. Right-regular grammar is a class of grammar whose rules are limited to $X \to \alpha Y$, $X \to \alpha$, and $X \to \epsilon$ where $X, Y \in N, \alpha \in \Sigma$ and ϵ is an empty string. With actions as production rules and plans as sentences, a classical planning problem forms a right-regular grammar. Moreover, any right-regular grammar can be modeled as a classical planning problem.

Theorem 2. *A classical planning problem $\langle P, A, I, G\rangle$ maps to the following grammar: (1) Σ consist of actions, i.e., $\Sigma = A$. (2) N contains the entire states, i.e., $N = 2^P$. (3) S is equivalent to the initial state I. (4) For each action $a = \langle \text{PRE}(a), \text{ADD}(a), \text{DEL}(a)\rangle \in A$ and each state s where a is applicable ($\text{PRE}(a) \subseteq s$), we add a production rule $s \to sas'$ where s' is a successor state $s' = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a)$. Note that s, s' are both non-terminal. (6) Finally, for every goal state s^* that satisfies the goal condition G ($G \subseteq s^*$), we add a production rule $s^* \to \epsilon$.*

Theorem 3. *A right-regular grammar maps to a classical planning problem $\langle P, A, I, G\rangle$ as follows: (1)P consists of non-terminal symbols N and a special proposition g, i.e. $P = N \cup \{g\}$. (2)I is a set $\{S\}$ where S is a start symbol. (3)For a rule $X \to \alpha Y$ where $X, Y \in N$, $\alpha \in \Sigma$, we add an action $\langle\{X\}, \{Y\}, \{X\}\rangle$. (4)For a rule $X \to \alpha$ and $X \to \epsilon$, we add an action $\langle\{X\}, \{g\}, \{X\}\rangle$. (5)The goal condition G consists of a single proposition g, i.e. $G = \{g\}$.*

Under this framework, the task of zero-shot sentence generation under a right regular grammar can be formalized as a classical planning problem. Notice that preconditions of each action plays a role similar to semantic and grammatical rules. While the simplicity of regular grammar may give a wrong impression that planning is easy, it is in fact PSPACE-hard and the search space explodes easily due to the exponential number of non-terminals ($N = 2^P$). A similar result between a more expressive planning formalism (Hierarchical Task Network planning (Ghallab et al., 2004)) and a more expressive Context Free Grammar is reported by Geib & Steedman (2007). In general, this connection between planning formalisms and formal grammars is often overlooked.

4 Discrete Sequential Application of Words (DSAW)

Common downstream tasks and embedding evaluation tasks in modern natural language processing with word embedding involve arithmetic vector operations that aggregate the embedding vectors. Analogy task (Mikolov et al., 2013c) is one such embedding evaluation 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 (Mikolov et al., 2013c), which attempts to find the closest word embedding to a vector $\boldsymbol{a}^* - \boldsymbol{a} + \boldsymbol{b}$ measured by the cosine distance $\cos(\boldsymbol{v}_1, \boldsymbol{v}_2) = 1 - \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{|\boldsymbol{v}_1||\boldsymbol{v}_2|}$, assuming that the result is close to the target embedding $\boldsymbol{b}^*$. This, along with other analogy calculation methods

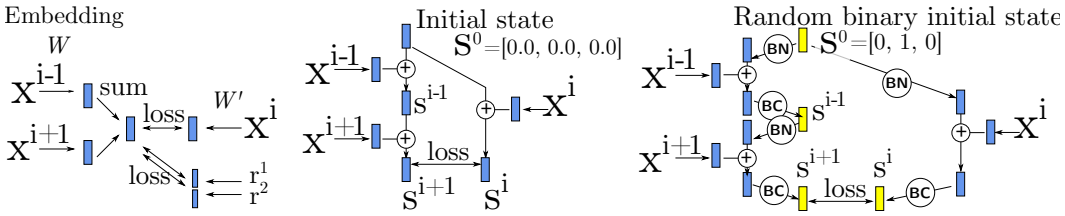

Figure 1: (Left) Traditional 3-gram CBOW with negative sampling. (Middle) 3-gram CBOW seen as a sequence of continuous state manipulations. (Negative sampling is not shown) (Right) 3-gram Discrete Sequential Application of Words model. BN=Batch Normalization, BC=Binary Concrete.

(Levy & Goldberg, 2014; Nissim et al., 2020; Drozd et al., 2016), uses simple vector arithmetic to obtain the result embedding used to predict the target word. In addition, text classification evaluation methods sometimes build classifiers based on the mean or the sum of the word vectors in a sentence or a document (Tsvetkov et al., 2015; Yogatama & Smith, 2014).

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

Provided that the zero-shot sentence generation under regular grammar can be seen as a classical planning problem, we aim to generate a classical planning model from a natural language corpus. This approach addresses the weaknesses above of existing symbolic NLP approaches — dependency to human symbols and opaqueness — by *generating* a set of propositional symbols by itself. Our embedding scheme thus stands upon propositional logic (like AMR) while supporting vector arithmetic (like continuous embedding). To achieve this goal, we combine the existing discrete variational method with CBOW Word2Vec and obtain atomic propositional representations of words.

To introduce the model, we modify the CBOW Word2Vec (Fig. 1, left) in two steps. We first identify that CBOW can be seen as a simple constant recurrent model (Fig. 1, middle). This trivial "recurrent" model merely adds the input embedding to the current state. Unlike the more complex, practical RNNs, such as LSTM (Hochreiter & Schmidhuber, 1997) or GRU (Cho et al., 2014), this model lacks any form of weights or nonlinearity that transforms the current state to the next state.

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

Second, upon discretizing some of the elements in this model in the next step, we should preserve the fundamental ability of CBOW to *add(+), remove(-) or keep(0)* the value of each dimension of the state vector. It is important to realize that a simple binary or categorical word embedding, such as the work done by Chen et al. (2018) (for a significantly different purpose), is incompatible with the concept of *adding*, *removing* or *keeping*. Notice that unlike continuous values, categorical values lack the inherent ordering (total or partial). Therefore, categorical values are not able to define *adding* and *removing* as the inverse operations, as well as *keeping* as an identity. Also notice that this *adding* and *removing* directly corresponds to the add/delete effects in classical planning formalism. An arbitrary binary representation that is not regularized to have these elements cannot be compactly represented in the STRIPS semantics, precluding efficient planning.

Based on the observations above, we propose Discrete Sequential Application of Words (DSAW, Fig. 1, right), which addresses the issues in continuous embeddings, naive discrete models, or handcrafted symbolic models (AMR) by using *two* binary vectors to represent each word.

DSAW sequentially applies the BTL technique to an initial state vector $\boldsymbol{s}^0$. It applies a Bernoulli(0.5) prior to every state, therefore $\boldsymbol{s}^0$ is sampled from Bernoulli(0.5) and each recurrent latent state $\boldsymbol{s}^{i+j}$ ($-c \leq j \leq c$) is sampled from Binary Concrete, a continuous relaxation. The embedding matrix W itself is not discrete. However, due to Theorem 1, we can extract *two* binary vectors $\textsc{add}(x)$, $\textsc{del}(x)$ of a word x that satisfy $\boldsymbol{s}^{i+1} = (\boldsymbol{s}^i \;\&\&\; !\textsc{del}(x)) \,||\, \textsc{add}(x)$, which is a bit-vector implementation of set-based STRIPS action application $s^{i+1} = (s^i \setminus \textsc{del}(a)) \cap \textsc{add}(a)$.

Since state vectors are activated by Binary Concrete, which behaves like a Sigmoid function in high temperature and as a step function in low temperature, all state vectors reside in the unit hypercube $[0, 1]^E$. This means that we cannot directly apply the traditional objective function $\log \sigma(\boldsymbol{x} \cdot \boldsymbol{y})$ in Word2Vec to the output state vector because it assumes that the distribution of $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^E$ is centered around the origin, while our discrete output states are heavily biased toward the positive orthant. To address this issue, we shift the mean by subtracting 0.5 from the output vector before computing the loss. Formally, our maximization objective (including negative sampling with $\{r^1, \dots r^K\}$) is defined as shown below, where $\boldsymbol{s}^i = \textsc{apply}(x^i, \boldsymbol{s}^0)$, $\boldsymbol{s}^{i-c} = \textsc{apply}(x^{i-c}, \boldsymbol{s}^0)$, $\boldsymbol{s}^{i+1} = \textsc{apply}(x^{i+1}, \boldsymbol{s}^{i-1})$, $\boldsymbol{s}^{i+j} = \textsc{apply}(x^{i+j}, \boldsymbol{s}^{i+j-1})(j \notin \{-c, 0, 1\})$. (Note that the formula below omits the variational loss. See Appendix Sec. B.3 for the full form.)

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

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

4.1 Inference in the discrete space

An important question about our model is how to perform arithmetic operations with the discrete representation. Specifically, to perform the word analogy task (Mikolov et al., 2013b), the representation must support both *addition and subtraction* of words, which is non-trivial for discrete vectors. We propose to use the STRIPS *progression* (applying an action) and *regression* (reversing an action) (Alcázar et al., 2013; Haslum et al., 2019) 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 (= vector addition). Similarly, for a state s^{i+1} generated by applying an action a to s^i ($s^{i+1} = (s^i \setminus \textsc{del}(a)) \cap \textsc{add}(a)$), a STRIPS regression [1] restores the previous state by $s^i = (s^{i+1} \setminus \textsc{add}(a)) \cap \textsc{del}(a)$. For a word x, we denote the corresponding bitwise operations as $s \,\hat{+}\, x$ and $s \,\hat{-}\, x$. We note that our operation is not associative or commutative. That is, the result of "king-man+woman" may be different from "king+woman-man" etc.

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

$$\textsc{add}\big(e \,\hat{+}\, x\big) = \textsc{add}(e) \setminus \textsc{del}(x) \cup \textsc{add}(x), \quad \textsc{del}\big(e \,\hat{+}\, x\big) = \textsc{del}(e) \setminus \textsc{add}(x) \cup \textsc{del}(x),$$
$$\textsc{add}\big(e \,\hat{-}\, x\big) = \textsc{add}(e) \setminus \textsc{add}(x) \cup \textsc{del}(x), \quad \textsc{del}\big(e \,\hat{-}\, x\big) = \textsc{del}(e) \setminus \textsc{del}(x) \cup \textsc{add}(x).$$

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

Example 1. *Assume a 2-dimensional word embedding, where each dimension is assigned a meaning [female, status]. Assume each word has the effects as shown in Table 1. Then the effect of "king-man+woman" applied to a state s is equivalent to those of "queen":*

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

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

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

Table 1: An example 2-dimensional embedding.

Embedding size E	200		500		1000	
Model	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
Word Similarity	0.531	0.509	0.533	0.538	0.523	**0.545**
Analogy Top1 acc.	0.521	0.273	**0.527**	0.373	0.477	0.373
Analogy Top10 acc.	0.736	0.564	**0.758**	0.683	0.720	0.673
Text Classification Test	0.845	0.867	0.8667	0.908	0.875	**0.930**

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

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

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

5 EVALUATION

We trained a traditional CBOW (our implementation) and our DSAW on 1 Billion Word Language Model Benchmark dataset (Chelba et al., 2014). Training details are available in the appendix Sec. B.3. We first compared the quality of embeddings on several downstream tasks.

5.1 EMBEDDING EVALUATION TASKS

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

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

Finally, we used our embeddings for semantic text classification, in which the model must capture the semantic information to perform well. We evaluated our model in two datasets: "*20 Newsgroup*" (Lang, 1995) and "*movie sentiment treebank*" (Socher et al., 2013). We created binary classification

tasks following the existing work (Tsvetkov et al., 2015; Yogatama & Smith, 2014): 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: ≤ 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 Sec. B.

5.2 Zero-Shot Paraphrasing with Classical Planning

Next, 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 zero-shot discover the phrase that shares the same concept. Given a word y, we generate a classical planning problem whose task is to sequence several words in the correct order to achieve the same effects that y has.

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 while excluding the target word y: $\textsc{pre}(a(x)) = \emptyset$, $\textsc{add}(a(x)) = \{p^a_i \mid \textsc{add}(x)_i = 1\} \cup \{p^d_i \mid \textsc{del}(x)_i = 1\}$, $\textsc{del}(a(x)) = \{p^d_i \mid \textsc{add}(x)_i = 1\} \cup \{p^a_i \mid \textsc{del}(x)_i = 1\}$. Finally, $I = \emptyset$ and $G(y) = \{p^a_i \mid \textsc{add}(y)_i = 1\} \cup \{p^d_i \mid \textsc{del}(y)_i = 1\}$. Note that $\textsc{add}(y), \textsc{del}(x)$ etc. are bit-vectors, while $\textsc{add}(a(x))$ etc. are sets expressed in PDDL. Finding the optimal solution of this problem is **NP**-Complete due to $\textsc{pre}(a(x)) = \emptyset$ (Bylander, 1994). Due to its worst-case hardness, we do not try to find the optimal solutions. We solved the problems with LAMA planner (Richter & Westphal, 2010) in Fast Downward planning system (Helmert, 2006), the winner of International Planning Competition 2011 satisficing track (López et al., 2015).

Notice that the goal condition of this planning problem is overly specific because it requires to perfectly match the target effect, while the neighbors of an embedding vector often also carry a similar meaning. In fact, LAMA classical planner were able to prove that there are no precise paraphrasing to all queries we provided. This ability to answer the inexistence of solutions is offered by the deterministic completeness of the algorithms (Greedy Best First Search and Weighted A^*) in these planners, which guarantees that the algorithm returns a solution in finite time whenever there is a solution, and returns "unsolvable" when there are no solutions. Such finite deterministic completeness is typically missing in probabilistic search algorithm such as Monte-Carlo Tree Search (Kocsis & Szepesvári, 2006), or greedy approach such as Beam Search commonly used in NLP literature. Also, recent state-of-the-art language models such as GPT-3 (Brown et al., 2020) are known to generate a bogus answer to a bogus question with high confidence (Lacker, 2020).

We can still address this issue in a non-probability-driven manner by *net-benefit* planning formalism (Keyder & Geffner, 2009), an extension of classical planning that allows the use of *soft-goals*. Net-benefit planning task $\langle P, A, I, G(y), c, u\rangle$ is same as the unit-cost classical planning except the cost function $c : A \to \mathbb{Z}^{+0}$ and $u : G(y) \to \mathbb{Z}^{+0}$. The task is to find an action sequence π minimizing the cost $\sum_{a\in\pi} c(a) + \sum_{p\in G(y)\setminus s^*} u(p)$, i.e., the planner tries to find a cheaper path while also satisfying as many goals as possible at the terminal state s^*. We used a simple compilation approach (Keyder & Geffner, 2009) to convert this net-benefit planning problem into a normal classical planning problem. The compilation details can be found in the Appendix Sec. C.2.

We specified both costs a constant: $c(a) = E$ for all actions and $u(p) = U$ for all goals, where we heuristically chose $U = 100$. The LAMA planner searches for suboptimal plans, iteratively refining

Word y	word sequence π (solution plan)	Word y	word sequence π (solution plan)
hamburgur	meat lunch chain eat	lake	young sea
lamborghini	luxury built car; car recall standard;	pond	wildlife nearby
lamborghini	electric car unlike toyota	river	valley lake nearby delta
subaru	motor toyota style ford	valley	mountain tenessee 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

Table 3: Paraphrasing of the source words returned by the LAMA planner. See Table 18-19 in the appendix Sec. C.6 for more examples.

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 ($V \approx 219$k) excluding y, function words (e.g., "the", "make"), and compound words (e.g., plurals). For each problem, we allowed the maximum of 4 hours runtime and 16GB memory. Typically the planner found the first solution early, and continued running until the time limit finding multiple better solutions. We show its example outputs in Table 3. See Appendix Sec. C.6 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 (Wermter & Lehnert, 1989; Towell & Shavlik, 1994). Zhao et al. (2018) proposed a discrete sentence representation, treating each sentence as an action. Chen et al. (2018) improved the training efficiency with an intermediate discrete code between the vocabulary V and the continuous embedding. These representations lack the STRIPS compatibility since the discrete dynamics is not regularized. In the intersection of planning and natural language processing, Rieser & Lemon (2009) introduced a system which models conversations as probabilistic planning and learns a reactive policy from interactions. Recent approaches extract a classical planning model from a natural language corpus (Lindsay et al., 2017; Feng et al., 2018), but using the opaque human symbols.

7 Conclusion

We proposed an unsupervised learning method for discrete binary word embeddings that preserve the vector arithmetic similar to the continuous embeddings. Our approach combines three distant areas: Unsupervised representation learning method for natural language, discrete generative modeling, and STRIPS classical planning formalism which is deeply rooted in the symbolic AIs and the propositional logic. Inspired by the recurrent view of the Continuous Bag of Words model, our model represents each word as a symbolic action that modifies the binary (i.e., propositional) recurrent states through effects.

We answered an important connection between zero-shot sequence generation, formal language grammar and planning as heuristic search. This is done by first establishing the theoretical connection between right-regular grammar and classical planning, then by proposing a system that learns a propositional embedding compatible with planning, then demonstrating that the planner can meaningfully compose words within the regular grammar. Future directions include learning hierarchical plannable actions and goals by taking advantage of Hierarchical Task Network planning formalism (Ghallab et al., 2004) which corresponds to Context Free Grammar, and probabilistic CFG grammar induction methods (Kim et al., 2019). Additionally, our goal-oriented sentence generation approach can be further expanded to the task of machine translation (same goal, different set of actions), or code generation where the grammar is stricter than in natural language.

REFERENCES

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*, pp. 19, 2009.

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

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)*, 2018a.

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*, 2018b.

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.

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.

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*, pp. 178–186, 2013.

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*, pp. 4349–4357, 2016.

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`.

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

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

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

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.

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*, pp. 853–862, 2018.

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*, pp. 1724–1734, 2014.

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

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*, pp. 3519–3530, 2016.

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*, pp. 19–24, 2014.

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.

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

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*, pp. 406–414, 2001.

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*, pp. 1426–1436, 2014.

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

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

Patrik Haslum, Nir Lipovetzky, 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.

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

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.

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

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

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

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*, pp. 448–456, 2015.

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

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

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`.

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*, pp. 2369–2385, 2019.

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

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*, pp. 56–64, 2017.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. pp. 282–293, 2006.

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.

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

Ken Lang. Newsweeder: Learning to Filter Netnews. In *Proc. of the International Conference on Machine Learning*, pp. 331–339, 1995.

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

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.

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

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.

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.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

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.

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*, 2013a.

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*, pp. 3111–3119, 2013b.

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*, pp. 746–751, 2013c.

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

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=nbsOIJ_saUAC`.

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*, pp. 99–109, 2017.

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.

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

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.

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*, pp. 337–346, 2011.

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

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.

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)*, pp. 683–691, 2009.

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

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*, pp. 1631–1642, 2013.

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*, pp. 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*, pp. 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.

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*, pp. 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*, pp. 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*, pp. 1098–1107, 2018.

CONTENTS

1 Introduction **1**

2 Preliminary and background **1**

3 Zero-Shot Sequence Generation as Planning **3**

4 Discrete Sequential Application of Words (DSAW) **3**

4.1 Inference in the discrete space 5

5 Evaluation **6**

5.1 Embedding Evaluation Tasks 6

5.2 Zero-Shot Paraphrasing with Classical Planning 7

6 Related work **8**

7 Conclusion **8**

A Extended Backgrounds **14**

A.1 Variational AutoEncoder with Gumbel Softmax and Binary Concrete distribution 14

A.2 Learning Discrete Latent Dynamics using Back-To-Logit 14

B Machine learning experiments **16**

B.1 Source code directory `discrete-word-embedding/` 16

B.2 Training dataset preparation 16

B.3 Training details 17

B.4 Rebuttal Experiments: CBOW Average Embedding vs. Sum Embedding, with a Fixed Text Classification Test Split 18

B.5 Rebuttal Experiments: Detailed Results on Text Classification with Average-Based CBOW and Fixed Test Split 19

B.6 Rebuttal Experiments: Detailed Results on Word Similarity with Average-Based CBOW 20

B.7 Additional experiments: Weight initialization with Logistic(0,1) distribution 21

B.8 Additional model experiments: Discrete implementation of SkipGram 22

B.9 Additional model experiments: Hybrid discrete-continuous CBOW model 23

B.10 Detailed, per-category results for the word similarity task 24

B.11 Detailed, per-category results for the analogy task 25

B.12 Exploring the best ordering of the discrete additive operations in analogy task 27

B.13 Detailed, per-category results for the text classification task 28

B.14 Additional experiments: Word compositionality 29

C Planning / paraphrasing experiments **32**

C.1 The archive directory `paraphrasing/` 32

C.2 Compilation of a net-benefit planning problem into a classical planning problem 32

C.3 The list of 68 target words used in the paraphrasing experiment 32
C.4 The statistics of the discrete effect vectors 32
C.5 Runtime statistics for the word paraphrasing experiment 33
C.6 Additional paraphrasing for a set of randomly selected 300 words 33
C.7 Rebuttal Experiments: Ad-hoc Paraphrasing using CBOW 38
C.7.1 Bag-of-word 38
C.7.2 Greedy approach 38
C.7.3 MILP 38
C.7.4 Qualitative Comparison 39
C.8 Details of the Vocabulary Pruning during the Paraphrasing Experiments 41

A Extended Backgrounds

A.1 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 $p(x)$ Kingma & Welling (2013). Gumbel-Softmax (GS) VAE Jang et al. (2017) and its binary special case Binary Concrete (BC) VAE Maddison et al. (2016) 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 τ) 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 $\text{UNIFORM}(0, 1)$. BinConcrete converges to the Heaviside step function at the limit $\tau \to 0$: $\text{BINCONCRETE}(x) \to \text{STEP}(x)$ (step function thresholded at 0).

A.2 Learning Discrete Latent Dynamics using Back-To-Logit

Cube-Space AutoEncoder (Asai & Muise, 2020) proposed a method for learning a binary latent representation s of visual time-series data while guaranteeing that every state transition / dynamics in the latent representation can be expressed in STRIPS action rule $s_{t+1} = a(s_t) = (s_t \setminus \text{DEL}(a)) \cup \text{ADD}(a)$ for some action a. It does so by using a unique architecture called Back-to-Logit (BTL) that regularizes the state transitions. BTL places a so-called *cube-like graph prior* on the binary latent space / transitions. To understand the prior, the background of *cube-like graph* is necessary.

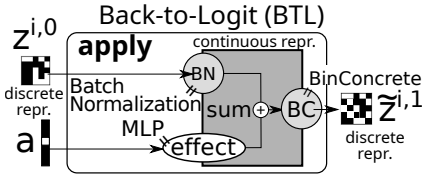

Figure 2: Back-To-Logit architecture

cube-like graph (Payan, 1992) is a graph class originating from graph theory. Asai & Muise (2020) identified that state transition graphs of STRIPS planning problems is equivalent to directed cube-like graph. A cube-like graph $G(S, D) = (V, E)$ is a simple[2] undirected graph defined by the sets S and D. Each node $v \in V$ is a finite subset of S, i.e., $v \subseteq S$. The set D is a family of subsets of S, and for every edge $e = (v, w) \in E$, the symmetric difference $d = v \oplus w = (v \setminus w) \cup (w \setminus v)$ must belong to D. For example, a unit cube is a cube-like graph because $S = \{x, y, z\}, V = \{\emptyset, \{x\}, \ldots \{x, y, z\}\}, E =$

[2]No duplicated edges between the same pair of nodes

$\{(\emptyset, \{x\}), \ldots (\{y, z\}, \{x, y, z\})\}, D = \{\{x\}, \{y\}, \{z\}\}$. The set-based representation can be alternatively represented as a bit-vector, e.g., $V' = \{(0, 0, 0), (0, 0, 1), \ldots (1, 1, 1)\}$.

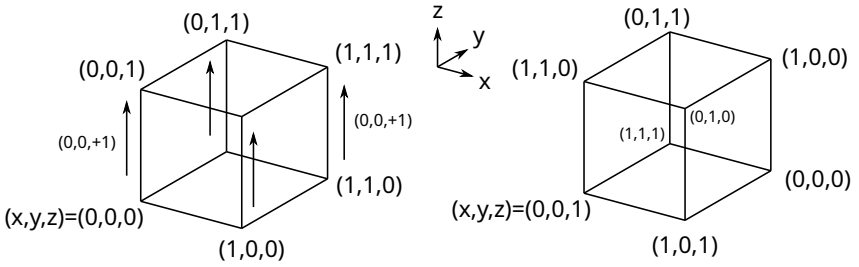

Figure 3: (Left) A graph representing a 3-dimensional cube which is a cube-like graph. (Right) A graph whose shape is identical to the left, but whose unique node embeddings are randomly shuffled.

Consider coloring a graph which forms a unit cube (Fig. 3) and has binary node embeddings. A cube-like graph on the left can be efficiently (i.e., by fewer colors) colored by the difference between the neighboring embeddings. Edges can be categorized into 3 labels (6 labels if directed), where each label is assigned to 4 edges which share the node embedding differences, as depicted by the upward arrows with the common node difference $(0, 0, +1)$ in the figure. This node embedding differences correspond to the set D, and each element of D represents an action. In contrast, the graph on the right has the node embeddings that are randomly shuffled. Despite having the same topology and the same embedding size, this graph lacks the common patterns in the embedding differences like we saw on the left, thus cannot be efficiently colored by the node differences.

In STRIPS modeling, Asai & Muise (2020) used a directed version of this graph class. For every edge $e = (v, w) \in E$, there is a pair of sets $d = (d^+, d^-) = (w \setminus v, v \setminus w) \in D$ which satisfies the asymmetric difference $w = (v \backslash d^-) \cup d^+$. It is immediately obvious that this graph class corresponds to the relationship between binary states and action effects in STRIPS, $s' = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a)$.

Cube-Space AE restricts the binary latent encoding and the transitions to directed cube-like graph, thereby guaranteeing the direct translation of latent space into STRIPS action model. However, since discrete representation learning is already known to be a challenge, adding a prior to it makes the training particularly difficult. Back-to-Logit (Fig. 2) was proposed in order to avoid directly operating on the discrete vectors. Instead, it converts a discrete current state $\boldsymbol{s}^i$ back to a continuous logit using Batch Normalization (Ioffe & Szegedy, 2015, BN), takes a sum with a continuous *effect* vector produced by an additional MLP $\text{EFFECT}\left(a^i\right)$, and re-discretize the resulting logit using Binary Concrete. Formally,

$$\tilde{\boldsymbol{z}}^{i,1} = \text{APPLY}(a, \boldsymbol{s}^i) = \text{BC}(\text{BN}(\boldsymbol{s}^i) + \text{EFFECT}\left(a^i\right)).$$

States learned by BTL has the following property:

Theorem 4. ***(Asai & Muise, 2020)*** *(same as Theorem 1) Under the same action a, state transitions are bitwise monotonic, deterministic, and restricted to three mutually exclusive modes, i.e., for each bit j:*

$$(\textit{add:})\ \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) \in \{(0, 1), (1, 1)\}\ \textit{i.e.}\ \boldsymbol{s}^i_j \leq \boldsymbol{s}'^i_j$$
$$(\textit{del:})\ \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) \in \{(1, 0), (0, 0)\}\ \textit{i.e.}\ \boldsymbol{s}^i_j \geq \boldsymbol{s}'^i_j$$
$$(\textit{nop:})\ \forall i; (\boldsymbol{s}^i_j, \boldsymbol{s}'^i_j) \in \{(0, 0), (1, 1)\}\ \textit{i.e.}\ \boldsymbol{s}^i_j = \boldsymbol{s}'^i_j$$

This theorem guarantees that each action deterministically sets a certain bit on and off in the binary latent space. Therefore, the actions and the transitions satisfy the STRIPS state transition rule $s' = (s \setminus \text{DEL}(a)) \cup \text{ADD}(a)$, thus enabling a direct translation from neural network weights to PDDL modeling language.

The proof is straightforward from the monotonicity of the BatchNorm and Binary Concrete. Note that we assume BatchNorm's additional scale parameter γ is kept positive or disabled.

Proof. For readability, we omit j and assumes a 1-dimensional case. Let $e = \text{EFFECT}(a) \in \mathbb{R}$. Note that e is a constant for the fixed input a. At the limit of annealing, Binary Concrete BCbecomes a STEP function, which is also monotonic. BNis monotonic because we assumed the scale parameter γ of BNis positive, and the main feature of BNalso only scales the variance of the batch, which is always positive. Then we have

$$\boldsymbol{s}'^i = \text{STEP}(\text{BN}(\boldsymbol{s}^i) + e).$$

The possible values a pair $(\boldsymbol{s}^i, \boldsymbol{s}'^i)$ can have is $(0,0), (0,1), (1,0), (1,1)$. Since both STEP and BNare deterministic at the testing time (See Ioffe & Szegedy (2015)), we consider the deterministic mapping from $\boldsymbol{s}^i$ to $\boldsymbol{s}'^i$. There are only 4 deterministic mappings: $\{(0,1),(1,1)\}$, $\{(1,0),(0,0)\}$, $\{(0,0),(1,1)\}$, and lastly $\{(0,1),(1,0)\}$. Thus our goal is now to show that the last mapping is impossible in latent space $\{\dots(\boldsymbol{s}^i, \boldsymbol{s}'^i)\dots\}$.

To prove this, first, assume $(\boldsymbol{s}^i, \boldsymbol{s}'^i) = (0,1)$ for some index i. Then

$$1 = \text{STEP}(\text{BN}(0) + e). \Rightarrow \text{BN}(0) + e > 0. \Rightarrow \text{BN}(1) + e > 0. \Rightarrow \forall i; \text{BN}(\boldsymbol{s}^i) + e > 0.$$

The second step is due to the monotonicity $\text{BN}(0) < \text{BN}(1)$. This shows $\boldsymbol{s}'^i$ is constantly 1 regardless of $\boldsymbol{s}^i$, therefore it proves that $(\boldsymbol{s}^i, \boldsymbol{s}'^i) = (1,0)$ cannot happen in any i.

Likewise, if $(\boldsymbol{s}^i, \boldsymbol{s}'^i) = (1,0)$ for some index i,

$$0 = \text{STEP}(\text{BN}(1) + e). \Rightarrow \text{BN}(1) + e < 0. \Rightarrow \text{BN}(0) + e < 0. \Rightarrow \forall i; \text{BN}(\boldsymbol{s}^i) + e < 0.$$

Therefore, $\boldsymbol{s}'^i = 0$ regardless of $\boldsymbol{s}^i$, and thus $(\boldsymbol{s}^i, \boldsymbol{s}'^i) = (0,1)$ cannot happen in any i.

Finally, if the data points do not contain $(0,1)$ or $(1,0)$, then by assumption they do not coexist. Therefore, the embedding learned by BTL cannot contain $(0,1)$ and $(1,0)$ at the same time. □

B MACHINE LEARNING EXPERIMENTS

B.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.

B.2 TRAINING DATASET PREPARATION

For the model training, we used 1 Billion Word Language Model Benchmark dataset (Chelba et al., 2014) 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 (Mikolov et al., 2013b). 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.

B.3 TRAINING DETAILS

The training is performed by batched stochastic gradient descent using Rectified Adam optimizer (Liu et al., 2019) 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:

$$\log \sigma(\boldsymbol{e}^i \cdot W'_{x^i}) + \sum_{k=1}^{K} \log \sigma(-\boldsymbol{e}^i \cdot W'_{r^k}).$$

For DSAW, where

$$\begin{aligned}
\boldsymbol{s}^i &= \textsc{apply}(x^i, \boldsymbol{s}^0), \\
\boldsymbol{s}^{i-j} &= \textsc{apply}(x^{i-j}, \textsc{apply}(\ldots \textsc{apply}(x^{i-c}, \boldsymbol{s}^0)\ldots), \\
\boldsymbol{s}^{i+j} &= \textsc{apply}(x^{i+j}, \textsc{apply}(\ldots \textsc{apply}(x^{i+1}, \textsc{apply}(x^{i-1}, \ldots \textsc{apply}(x^{i-c}, \boldsymbol{s}^0))))),
\end{aligned}$$

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

$$\begin{aligned}
\log \sigma((\boldsymbol{s}^{i+c} - \frac{1}{2}) \cdot (\boldsymbol{s}^i - \frac{1}{2})) - \sum_{k=1}^{K} \log \sigma(-(\boldsymbol{s}^{i+c} - \frac{1}{2}) \cdot (\textsc{apply}(r^k, \boldsymbol{s}^0) - \frac{1}{2})) \\
- \sum_{-c \leq j \leq c, j \notin \{0,1\}} \beta D_{\mathrm{KL}}(q(\boldsymbol{s}^{i+j} | x^{i+j}, \boldsymbol{s}^{i+j-1}) || p(\boldsymbol{s}^{i+j})) \\
- \beta D_{\mathrm{KL}}(q(\boldsymbol{s}^{i+1} | x^{i+1}, \boldsymbol{s}^{i-1}) || p(\boldsymbol{s}^{i+1})) \\
- \beta D_{\mathrm{KL}}(q(\boldsymbol{s}^{i} | x^{i}, \boldsymbol{s}^{0}) || p(\boldsymbol{s}^{i}))
\end{aligned}$$

where $p(\boldsymbol{s}^{i+j}) = \text{Bernoulli}(0.5)$ for all i, j, $\boldsymbol{s}^0 \sim \text{Bernoulli}(0.5)$, $D_{\mathrm{KL}}(q(\cdot)||p(\cdot))$ is the KL divergence for each discrete variational layer and β is the scale factor as in β-VAE (Higgins et al., 2017).

The temperature parameter τ 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:

$$\tau(t) = \begin{cases} 5 & (0 \leq t < T) \\ 5 \cdot \exp(\log \frac{0.7}{5} \cdot \lfloor \frac{t-T}{0.2} \rfloor \cdot 0.2) & (T < t \leq 8) \end{cases}$$

where T is a hyperparameter that determines when to start the annealing. τ 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. B.7.

B.4 REBUTTAL EXPERIMENTS: CBOW AVERAGE EMBEDDING VS. SUM EMBEDDING, WITH A FIXED TEXT CLASSIFICATION TEST SPLIT

As the reviewers suggested, our initial results were based on CBOW which aggregates the context word vectors by summation, instead of averaging. Reviewers also pointed out that the test split in the text classification task is different from existing work. We fixed these issues and obtained the results in Table 4. Accuracy scores of CBOW increased across all three tasks. Accuracy scores of DSAW (which is also affected due to the updated test split) was not significantly affected.

For reference, Sec. 5 is the main result table we had during the first submission.

Embedding size E	200		500		1000	
Model	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
Word Similarity	0.557	0.509	**0.563**	0.538	0.551	0.545
Analogy Top1 acc.	0.539	0.273	**0.547**	0.373	0.503	0.373
Analogy Top10 acc.	0.762	0.564	**0.778**	0.683	0.753	0.673
Text Classification Test	0.906	0.869	0.930	0.906	**0.943**	0.929

Table 4: Embedding evaluation task performance with updated implementations. In all tasks, higher scores are better. Best results are in bold.

Embedding size E	200		500		1000	
Model	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
Word Similarity	0.528	0.509	0.518	0.538	0.488	**0.545**
Analogy Top1 acc.	**0.438**	0.273	0.413	0.373	0.333	0.373
Analogy Top10 acc.	0.682	0.564	0.671	**0.683**	0.587	0.673
Text Classification Test	0.890	0.867	0.920	0.908	0.920	**0.930**

Table 5: Results with old implementations for reference.

B.5 REBUTTAL EXPERIMENTS: DETAILED RESULTS ON TEXT CLASSIFICATION WITH AVERAGE-BASED CBOW AND FIXED TEST SPLIT

Table 6 contains the new evaluation results on text classification. We also adopted the standard test/train split for SST2 dataset. For reference, we also show the old table (Table 7).

Data	Sentence len.		200		500		1000	
(Num. documents)	avg.	med.	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
SCI (1994)	276	229	.988	.952	**.996**	.988	**.996**	.995
COMP (1981)	341	255	.829	.938	.913	.980	.946	**.984**
SPORT (1987)	383	269	.959	.941	.975	.985	.984	**.993**
RELI (1995)	447	324	.998	.976	**.999**	.994	**.999**	.995
SST2 (9613)	17	17	.785	.636	.794	.669	**.798**	.705
Total (17571)			.906	.869	.930	.906	**.943**	.929

Table 6: Per-category accuracies for the text classification task. (new results)

Data	Sentence len.		200		500		1000	
(Num. documents)	avg.	med.	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
SCI (1994)	276	229	.976	.952	.983	.988	.990	**.995**
COMP (1981)	341	255	.809	.938	.892	.980	.931	**.984**
SPORT (1987)	383	269	.902	.941	.952	.985	.971	**.993**
RELI (1995)	447	324	.996	.976	**.999**	.994	**.999**	.995
MS (9142)	17	17	.770	.629	**.773**	.666	.741	.704
Total (17099)			.890	.867	.920	.908	.920	**.930**

Table 7: Per-category accuracies for the text classification task. (old results)

B.6 REBUTTAL EXPERIMENTS: DETAILED RESULTS ON WORD SIMILARITY WITH AVERAGE-BASED CBOW

We re-evaluated CBOW model after fixing the word vector aggregation from summation to averaging. The new results are in Table 8. For reference, we also show the old table (Table 9).

	Number of	$E = 200$		500		1000	
Data	word pairs	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
WS	353	**.576**	.548	.575	.556	.575	.568
WSR	252	.511	.493	.528	.503	**.529**	.518
WSS	203	.669	.622	.659	.652	.649	**.680**
MT	287	**.641**	.611	.635	.599	.631	.561
MEN	3000	.721	.657	**.732**	.696	.723	.710
RW	2034	.417	.378	.399	**.394**	.388	.377
SM	999	.362	.328	**.378**	.356	.366	.359
Total	7128	.557	.509	**.563**	.538	.551	.545

Table 8: Word similarity results compared by the datasets (new results).

	Number of	$E = 200$		500		1000	
Data	word pairs	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
WS	353	.540	.548	.506	.556	.478	**.568**
WSR	252	.474	.493	.472	.503	.475	**.518**
WSS	203	.641	.622	.580	.652	.594	**.680**
MT	287	**.617**	.611	.609	.599	.589	.561
MEN	3000	.692	.657	.691	.696	.667	**.710**
RW	2034	.359	.378	.341	**.394**	.302	.377
SM	999	.340	.328	.344	.356	.304	**.359**
Total	7128	.528	.509	.518	.538	.488	**.545**

Table 9: Word similarity results compared by the datasets (old results).

B.7 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 Kocmi & Bojar (2017); Neishi et al. (2017), 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_i} may fail to sufficiently affect the output values if its absolute value $|W_{x_i}|$ 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 10 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	
Initialization	Gaussian	Logistic	Gaussian	Logistic	Gaussian	Logistic
Word Similarity	0.504	0.509	0.531	**0.538**	**0.546**	0.545
Analogy Top1 acc.	0.222	**0.273**	0.332	**0.373**	0.352	**0.373**
Analogy Top10 acc.	0.526	**0.564**	0.662	**0.683**	0.668	**0.673**
Text Classification Test	0.680	**0.867**	0.707	**0.908**	0.758	**0.930**

Table 10: Downstream task performance of DSAW models using Logistic vs. Gaussian weight initialization. The better initialization under the same hyperparameter set is highlighted in bold.

B.8 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 word 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 Asai & Muise (2020) 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 11 shows the summary of the SG model and the SG-BTL model on the tasks we evaluated on.

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

Table 11: Downstream task performance of for SG and SG-BTL.

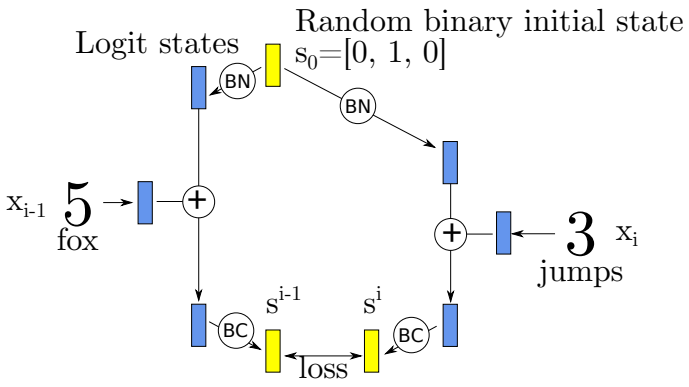

Table 12: Schematic diagram of SG-BTL.

B.9 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 $\boldsymbol{e}$ and the shifted discrete output state $\boldsymbol{s}^{i+c} - 0.5$, then applying the standard Word2Vec loss $\log \sigma(\boldsymbol{x} \cdot \boldsymbol{y})$ Mikolov et al. between the target and predict embedding $\boldsymbol{x}, \boldsymbol{y}$. 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 13 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 Hybrid	500 Hybrid	1000 Hybrid
Word Similarity	0.444	0.498	0.492
Analogy Top1 acc.	0.136	0.283	0.370
Analogy Top10 acc.	0.377	0.596	**0.689**
Text Classification Test	0.836	0.858	0.849

Table 13: Overall performance of the Hybrid models.

B.10 DETAILED, PER-CATEGORY RESULTS FOR THE WORD SIMILARITY TASK

Word similarity task is the standard benchmark for measuring the attributional similarity Miller & Charles (1991); Resnik (1995); Agirre et al. (2009). Given a set of word pairs, each embedding is evaluated by computing the Spearman correlation between the similarity scores assigned by the embedding and those assigned by human Rubenstein & Goodenough (1965); Faruqui & Dyer (2014); Myers et al. (2010). 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 $\text{ADD}(x) - \text{DEL}(x)$, 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**) Bruni et al. (2014); Radinsky et al. (2011); Luong et al. (2013); Hill et al. (2015); Finkelstein et al. (2001). WS353 dataset is further separated into relatedness (**WSR**) and similarity (**WSS**) Agirre et al. (2009). 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 *releated* 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 14.

	Number of	$E = 200$		500		1000	
Data	word pairs	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
WS	353	.540	.548	.506	.556	.478	**.568**
WSR	252	.474	.493	.472	.503	.475	**.518**
WSS	203	.641	.622	.580	.652	.594	**.680**
MT	287	**.617**	.611	.609	.599	.589	.561
MEN	3000	.692	.657	.691	.696	.667	**.710**
RW	2034	.359	.378	.341	**.394**	.302	.377
SM	999	.340	.328	.344	.356	.304	**.359**
Total	7128	.528	.509	.518	.538	.488	**.545**

Table 14: Word similarity results compared by the datasets (CBOW and DSAW).

B.11 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. (2013b). In the ADD column of Table 15, 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 $\boldsymbol{a}^* - \boldsymbol{a} + \boldsymbol{b}$ for the nearest neighbor, and DSAW uses the STRIPS progression $\hat{-}\ \boldsymbol{a}\ \hat{+}\ \boldsymbol{a}^*\ \hat{+}\ \boldsymbol{b}$. 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 Levy & Goldberg (2014); Nissim et al. (2020); Drozd et al. (2016) compared to ADD scheme. Compared to the ADD column in Table 15, which uses all three input words for the analogy (e.g., $\boldsymbol{a}^* - \boldsymbol{a} + \boldsymbol{b}$), Ignore-A does not use $\boldsymbol{a}$ (e.g., $\boldsymbol{a}^* + \boldsymbol{b}$), and Only-B uses $\boldsymbol{b}$ 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 ($\boldsymbol{a}^* - \boldsymbol{a}$), or just from the neighborhood structure of the target word $\boldsymbol{b}$ and $\boldsymbol{a}^*$. 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. 4), confirming the validity to our approach.

Method	ADD		Ignore-A		Only-B	
Model	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
capital-common-countries	**.974**	.970	.002	.957	.324	.957
capital-world	.846	**.975**	.001	.975	.184	.966
city-in-state	**.850**	.793	.001	.779	.092	.687
currency	.087	**.117**	.001	.113	.001	.032
family	.793	**.887**	.002	.899	.504	.913
gram1-adjective-to-adverb	.115	**.128**	.001	.122	.019	.094
gram2-opposite	.245	**.448**	.003	.442	.037	.414
gram3-comparative	**.933**	.758	.002	.730	.102	.541
gram4-superlative	**.548**	.302	.002	.283	.035	.147
gram5-present-participle	.733	**.794**	.001	.783	.257	.818
gram6-nationality-adjective	**.742**	.724	.003	.656	.041	.390
gram7-past-tense	.742	**.840**	.001	.838	.355	.850
gram8-plural	**.658**	.653	.002	.641	.399	.649
gram9-plural-verbs	.644	**.792**	.001	.781	.241	.733

Table 15: Word Analogy accuracies compared by each category. Data from the best performing CBOW and DSAW models with the embedding size 1000.

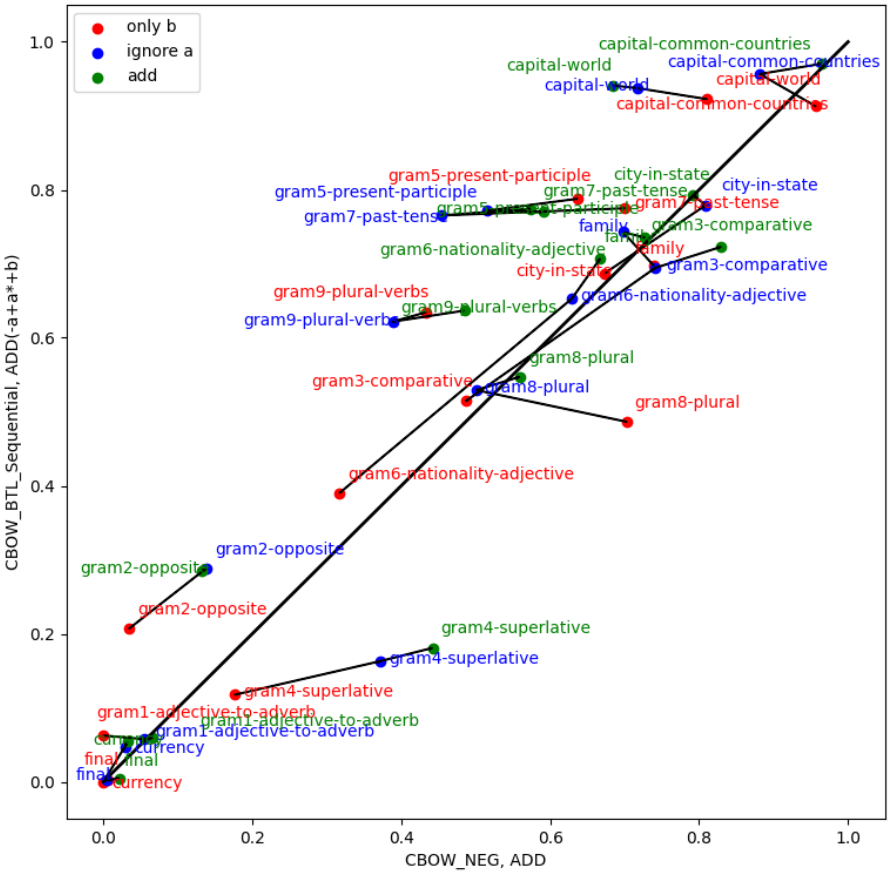

Figure 4: Scatter plot of the best analogy accuracies with CBOW (x-axis) and DSAW (y-axis) for each dataset category. Each path indicates the performance change caused by the different analogy method ADD, Ignore-A, Only-B. The path tends to move toward top-right, indicating that both embeddings are utilizing the differential information in the vectors $\boldsymbol{a}$ and $\boldsymbol{a}^*$ for analogy, not just the neighborhood structure of $\boldsymbol{b}$ and $\boldsymbol{a}^*$.

B.12 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 16, 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, $\hat{-}\ \boldsymbol{a} \hat{+} \boldsymbol{a}^* \hat{+} \boldsymbol{b}^*$.

Method Orders	$\boldsymbol{b}\hat{-}\boldsymbol{a}\hat{+}\boldsymbol{a}^*$	$\boldsymbol{b}\hat{+}\boldsymbol{a}^*\hat{-}\boldsymbol{a}$	$\boldsymbol{a}^*\hat{-}\boldsymbol{a}\hat{+}\boldsymbol{b}$	$\boldsymbol{a}^*\hat{+}\boldsymbol{b}\hat{-}\boldsymbol{a}$	$\hat{-}\boldsymbol{a}\hat{+}\boldsymbol{b}\hat{+}\boldsymbol{a}^*$	$\hat{-}\boldsymbol{a}\hat{+}\boldsymbol{a}^*\hat{+}\boldsymbol{b}$
capital-common-countries	.530	.285	.953	.506	.832	**.970**
capital-world	.255	.163	.961	.362	.523	**.975**
city-in-state	.426	.098	.726	.155	.617	**.793**
currency	.086	.023	.047	.028	**.117**	.065
family	.354	.121	.842	.219	.504	**.887**
gram1-adjective-to-adverb	.064	.011	.114	.011	.126	**.128**
gram2-opposite	.060	.016	.400	.036	.111	**.448**
gram3-comparative	.261	.153	.633	.191	.481	**.758**
gram4-superlative	.174	.032	.210	.064	.285	**.302**
gram5-present-participle	.099	.105	**.794**	.247	.221	.793
gram6-nationality-adjective	.267	.143	.677	.302	.462	**.724**
gram7-past-tense	.124	.144	.838	.273	.289	**.840**
gram8-plural	.074	.058	.644	.169	.143	**.653**
gram9-plural-verbs	.103	.124	.758	.253	.231	**.792**

Table 16: 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.

B.13 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*" Lang (1995) and "*movie sentiment treebank*" Socher et al. (2013). We created binary classification tasks following the existing work Tsvetkov et al. (2015); Yogatama & Smith (2014): 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: ≤ 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 Yogatama & Smith (2014). 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 Yogatama & Smith (2014).

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 ≈ 2000 questions each while **MS** has ≈ 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 17, 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. C.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	Sentence len.		200		500		1000	
(Num. documents)	avg.	med.	CBOW	DSAW	CBOW	DSAW	CBOW	DSAW
SCI (1994)	276	229	.976	.952	.983	.988	.990	**.995**
COMP (1981)	341	255	.809	.938	.892	.980	.931	**.984**
SPORT (1987)	383	269	.902	.941	.952	.985	.971	**.993**
RELI (1995)	447	324	.996	.976	**.999**	.994	**.999**	.995
MS (9142)	17	17	.770	.629	**.773**	.666	.741	.704
Total (17099)			.890	.867	.920	.908	.920	**.930**

Table 17: 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.

B.14 ADDITIONAL EXPERIMENTS: WORD COMPOSITIONALITY

One shortcoming of continuous vector operations in CBOW is that the resulting embedding is easily affected by the syntactic and semantic redundancy. These redundancies should ideally carry no effect on logical understanding, and at most with diminishing effect when repetition is used for subjective emphasis. Consider the phrase "red red apple". While the first "red" has the effect of specifying the color of the apple, the second "red" is logically redundant in the syntactic level. Phrases may also contain semantic redundancy, such as "free gift" and "regular habit". However, in a continuous word embedding, simple summation or averaging would push the result vector toward the repeated words or meanings. That is, for any non-zero vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\cos(\boldsymbol{a} \cdot n + \boldsymbol{b}, \boldsymbol{a}) \to 0, (n \to \infty)$ (Fig. 5). Even with a more sophisticated aggregation method for a vector sequence, such as the recurrent neural networks (Hochreiter & Schmidhuber, 1997), the problem still remains as long as it is based on a continuous representation. The model has to address it somehow, either at word embedding level or at sentence embedding level.

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

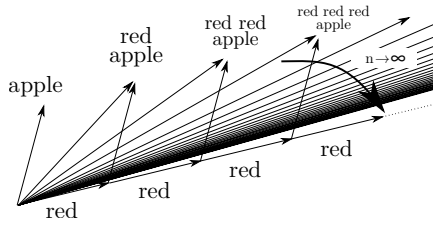

Figure 5: The shortcoming of adding continuous vectors in a cosine vector space.

In this section, we demonstrate this pathological behavior of CBOW and show that DSAW addresses this by visualizing the embeddings of composed words. We used Principal Component Analysis (Pearson, 1901) to visualize the linear projection of the embedding space. Phrase embeddings are obtained by the repeated $\hat{+}$ (DSAW) or averaging (CBOW) – the latter choice is purely for the visualization (length does not affect the cosine distance.) For CBOW and DSAW, we used the models that performed the best in analogy task.

In Fig. 6, 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.

Another 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.

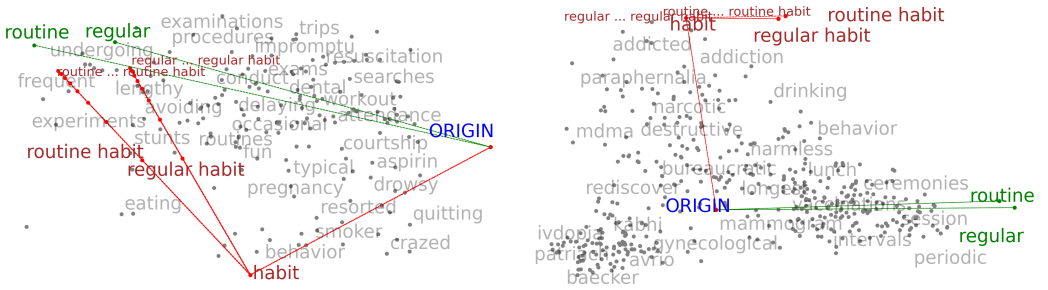

Figure 6: PCA plots of words/phrases in continous/discrete embeddings (best on computer screen). In all plots, we additionally included the union of 50 nearest neighbor words of each dot.

In Fig. 7, 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. 8, 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. 9, we plot Quadrupedal ruminant mammal = sheep. The phrase vector in both embeddings appears to roughly share the direction with “sheep”.

In Fig. 10, 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.

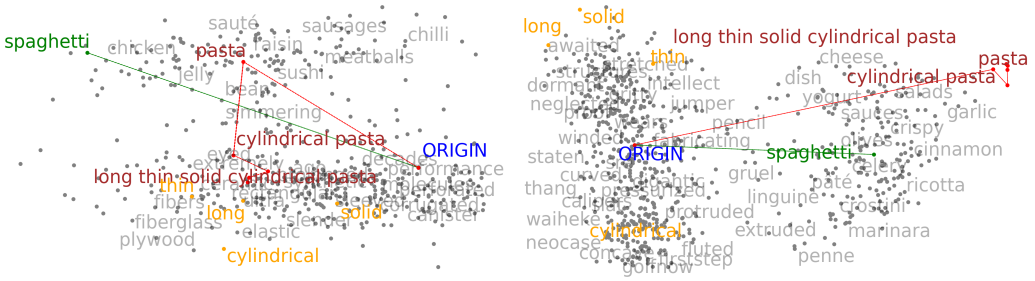

Figure 7: Plotting the compositional phrases with CBOW (left) and DSAW (right): Long thin solid cylindrical pasta = spaghetti, from `https://en.wikipedia.org/wiki/Spaghetti`.

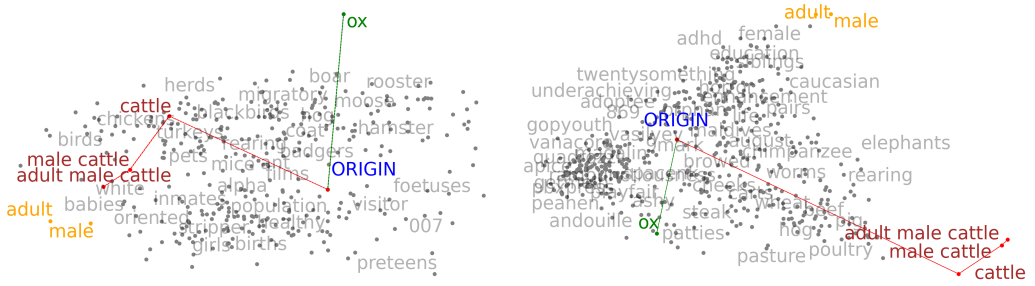

Figure 8: Plotting the compositional phrases with CBOW (left) and DSAW (right): Adult male cattle = ox, from `https://en.wikipedia.org/wiki/Ox`.

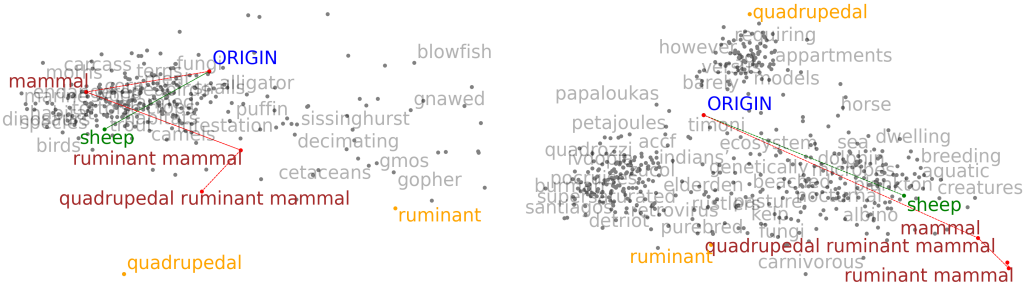

Figure 9: Plotting the compositional phrases with CBOW (left) and DSAW (right): Quadrupedal ruminant mammal = sheep, from `https://en.wikipedia.org/wiki/Sheep`.

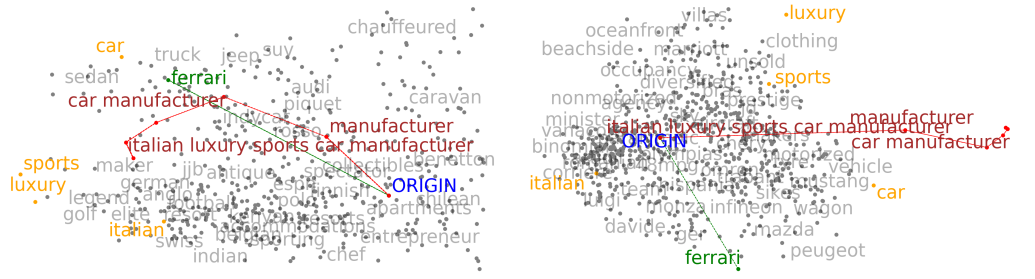

Figure 10: Plotting the compositional phrases with CBOW (left) and DSAW (right): Italian luxury sports car manufacturer = Ferrari, from `https://en.wikipedia.org/wiki/Ferrari`.

C PLANNING / PARAPHRASING EXPERIMENTS

C.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 Hoffmann & Nebel (2001) (reimplementation in Fast Downward Helmert (2006)) that first performs Greedy Best First Search, then continues refining the solution with Weighted A^* with decreasing weights $\{10, 5, 3, 2, 1\}$.

C.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 $\left\langle P', A', I', G', c'\right\rangle$ as follows Keyder & Geffner (2009). We use the slightly different notation from Keyder & Geffner (2009) by assuming a negative precondition extension Haslum et al. (2019) and by assuming all goals are soft:

$$
\begin{aligned}
P' &= P \cup \{\textit{end-mode}\} \cup \{\textit{marked}(p) \mid p \in G\} \\
A' &= A'' \cup \{\textit{end}\} \cup \{\textit{collect}(p), \textit{forgo}(p) \mid p \in G\} \\
A'' &= \{\langle \textsc{pre}(a) \wedge \neg \textit{end-mode}, \textsc{add}(a), \textsc{del}(a)\rangle \mid a \in A\} \\
\forall a'' \in A''; c'(a'') &= c(a) \\
\textit{collect}(p) &= \langle \textit{end-mode} \wedge p \wedge \neg \textit{marked}(p), \textit{marked}(p), \emptyset\rangle \\
c'(\textit{collect}(p)) &= 0 \\
\textit{forgo}(p) &= \langle \textit{end-mode} \wedge \neg p \wedge \neg \textit{marked}(p), \textit{marked}(p), \emptyset\rangle \\
c'(\textit{forgo}(p)) &= u(p) \\
\textit{end} &= \langle \neg \textit{end-mode}, \textit{end-mode}, \emptyset\rangle \\
c'(\textit{end}) &= 0 \\
G' &= \{\textit{marked}(p) \mid p \in G\}.
\end{aligned}
$$

Also, as mentioned in Keyder & Geffner (2009), we add additional preconditions to *collect*, *forgo* that linearize the ordering between the actions, i.e., for $i > 0$, $\textsc{pre}(\textit{collect}(p_i)) = \textit{end-mode} \wedge p_i \wedge \neg \textit{marked}(p_i) \wedge \textit{marked}(p_{i-1})$. (Same for *forgo*.)

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

C.3 THE LIST OF 68 TARGET WORDS USED IN THE PARAPHRASING EXPERIMENT

In the paraphrasing experiments, we hand-picked the words listed in Fig. 11 and generate the paraphrasing planning problem.

C.4 THE STATISTICS OF THE DISCRETE EFFECT VECTORS

Schakel & Wilson Schakel & Wilson (2015); Wilson & Schakel (2015) discussed the relationship between the word frequency and the magnitude (length) of the continuous word embedding vectors. In contrast, DSAW embedding consists of two vectors, $\textsc{add}(x)$ and $\textsc{del}(x)$, 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 presense, i.e., $\frac{1}{E}\sum_{j=1}^{E} \textsc{add}(x)_j$ and $\frac{1}{E}\sum_{j=1}^{E} \textsc{del}(x)_j$. We plotted these statistics for each word x in the order of frequency.

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 11: The list of words used for the paraphrasing experiments.

In Fig. 12, 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.

C.5 Runtime statistics for the word paraphrasing experiment

We visualized the runtime statistics of the paraphrasing task. Fig. 13 (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. 13 (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. 14, 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.

C.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 18-19. The additional data dump can be found in `paraphrasing/target_words_4000_8000-1-100/` directory.

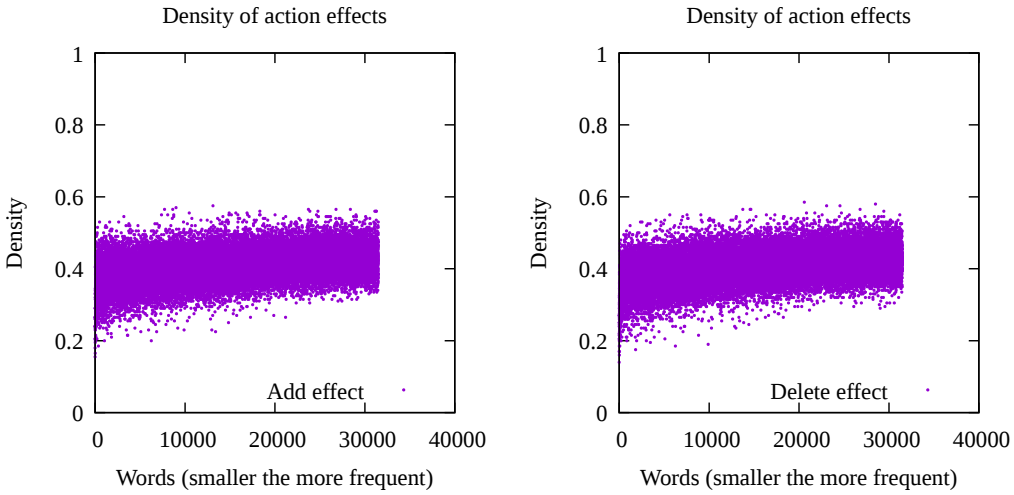

Figure 12: The density of add/delete effects (left, right) in the best DSAW model trained with $E = 200$, where x-axis is the word index sorted according to the frequency (frequent words are assigned the smaller indices), cut off at 32000-th word. We observe that rare words tend to have more effects.

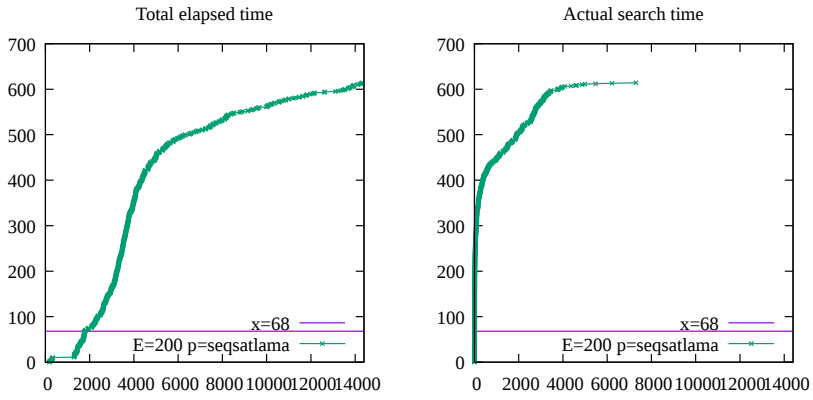

Figure 13: (left) Cumulative plot of the number of solutions found at the total time t, and (right) the same statistics based on the actual search time, i.e., the runtime excluding the time for the input parsing and initialization.

Word y	word sequence π (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
consent	approval written ; written approval prior; formal knowlegde
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; cleveland kentucky *
nutrition	medicine food
offence	criminal cause ; criminal sign
passport	card identity ; account identity

Table 18: (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. Note*: Louisville, Cleveland and Pittsburgh are the central city of Kentucky, Ohio and Pensilvania, respectively.

Word y	word sequence π (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
surgeon	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 19: (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*: Puerto Rico and Taiwan are both islands; Puerto Rico and Argentina both speak spanish.

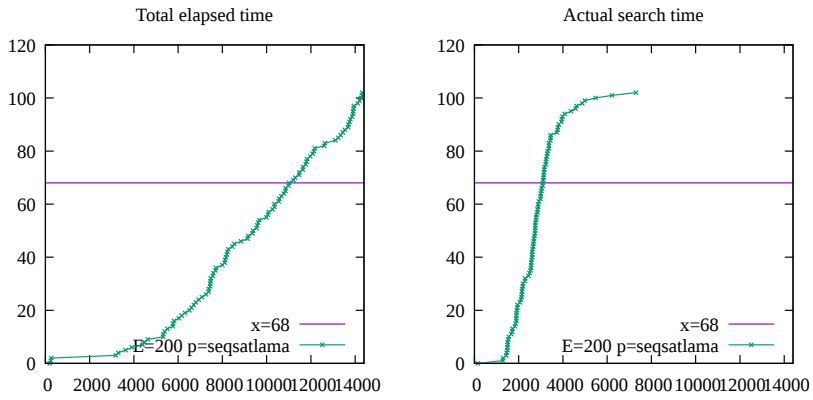

Figure 14: The same plot as Table 13, but the length is restricted to be larger than 2.

C.7 REBUTTAL EXPERIMENTS: AD-HOC PARAPHRASING USING CBOW

Finally, to respond to a request for more paraphrasing experiments, we added zero-shot paraphrasing for continuous embedding (CBOW). We implemented three variants:

1. Mixed Integer Linear Programming (MILP) model that finds a set of words whose average is closest to the target word embedding.
2. A greedy approach which finds a phrase in a sequential fashion: In each iteration, it selects a word that results in the closest averaged phrase embedding as the target word.
3. k-NN model: It simply finds k-Nearest Neighbors of the target words in the embedding space.

Each method have a hyperparameter L which specifies the maximum length/size of the result. The MILP and k-NN methods are order-insensitive, therefore the result output is a set rather than a sequence. The greedy approach is sequential. Since all approaches can be applied to the ternary numeric conversion of DSAW embeddings (using $\{-1, 0, +1\}$), we also performed the experiments for DSAW to show that planning approach is preferable. We discuss the detail of each method below.

C.7.1 BAG-OF-WORD

We find $k = L$ nearest neighbors of the target word.

C.7.2 GREEDY APPROACH

In the greedy approach, we generate a sequence of words of length L that does not contain duplicates. While we could stop the iteration when adding more words no longer makes the distance closer, we believe this feature is not necessary because we could look at the prefix of the result sequence.

Algorithm 1 A greedy solution. Input: target vector $\boldsymbol{y}$, target length L, embedding matrix $\boldsymbol{W}$.

procedure GREEDYPARAPHRASE($\boldsymbol{y}, L, \boldsymbol{W}$)
  Initialize a zero vector $\boldsymbol{x}$
  Initialize vocabulary $V \leftarrow \{1, 2, \cdots, |V|\}$.
  Normalize each embedding vector $\boldsymbol{W}_i$ to length 1 ($\boldsymbol{W}_i \leftarrow \frac{\boldsymbol{W}_i}{||\boldsymbol{W}_i||_2}$)
  Result $R \leftarrow [\,]$
  $i \leftarrow 0$
  while $i < L$ **do** ▷ repeat L times
    $j \leftarrow \arg\min_{j \in V} ||(\boldsymbol{x} + \boldsymbol{W}_j)/i - \boldsymbol{y}||_2^2$
      ▷ Find the word j that makes the average $(\boldsymbol{x} + \boldsymbol{W}_j)/i$ closest to $\boldsymbol{y}$.
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{W}_j$
    $R \leftarrow R$.push(j)
    $V \leftarrow V \setminus \{j\}$ ▷ Avoid using the same word more than twice.
    $i \leftarrow i + 1$
  return R

C.7.3 MILP

The MILP model is formalized by the following variables and constants.

- $\boldsymbol{W}$, a constant, continuous word embedding matrix (size $V \times E$). Each embedding vector $\boldsymbol{W}_i$ is normalized to length 1 ($\boldsymbol{W}_i \leftarrow \frac{\boldsymbol{W}_i}{||\boldsymbol{W}_i||_2}$).
- $\boldsymbol{s}$, a vector of 0/1 variables indicating the word is selected (size V).
- $\boldsymbol{y}$, a constant word vector for the target word (size E).
- L, an integer constant specifying the number of words to be selected.

The model then minimizes $||\boldsymbol{W}\boldsymbol{s}/L - \boldsymbol{y}||_2^2$ subject to $2 \leq \sum_i \boldsymbol{s}_i \leq L$. To make this objective function expressive in MILP, we perform a couple of preprocessing. First, we divide $\boldsymbol{W}$ by L. Next, since the objective function is apparently quadratic, we should transform it into a linear objective while preserving the optima. Imagine each element of the summation included in the squared error.

$$(\boldsymbol{W}_{ij}\boldsymbol{s}_i - \boldsymbol{y}_j)^2 = \boldsymbol{W}_{ij}^2\boldsymbol{s}_i^2 + \boldsymbol{y}_j^2 - 2\boldsymbol{W}_{ij}\boldsymbol{s}_i\boldsymbol{y}_j$$

Since we know $\boldsymbol{s}_i$ is either 0 or 1, we can replace $\boldsymbol{s}_i^2$ with $\boldsymbol{s}_i$. $\boldsymbol{y}_j^2$ is a constant, therefore we do not have to add it to the objective function. $\boldsymbol{W}_{ij}^2$ is also a constant, therefore we can precompute their values. As a result, we have a linear objective

$$\sum_{ij}(\boldsymbol{W}_{ij}^2 - 2\boldsymbol{W}_{ij}\boldsymbol{y}_j)\boldsymbol{s}_i$$

which has the same optima as the quadratic objective.

We performed the experiments using IBM ILOG CPLEX solver version 12.10.0.3. Each experiment typically finishes in a matter of seconds.

C.7.4 Qualitative Comparison

Table 20 compares several hand-picked paraphrasing examples obtained from zero-shot paraphrasing methods. The outputs of the naive methods (MILP, greedy, k-NN) do not sound like a sensible English phrase, regardless of the underlying embedding (CBOW or DSAW). In contrast, definitions generated by planners are much easier to understand. Note that, since we did not allow the paraphrase to contain function words (e.g. "of", "for") and derived words (e.g. past participle) that can be used as adjectives, some results may have to be interpreted by augmenting proper suffixes and function words. For example, "position china" could be "position **of** china", "continue enjoy" could be "continue **to** enjoy", and "guarantee cash" could be "guarante**ed** cash".

Looking into individual outputs generated by naive methods, one can notice that they seem to be merely a set of words similar in the semantic space without any sense of order, even though the greedy generation method is sequential. Specifically, (1) While they may not contain duplicate words, the words included in the result tend to contain overlapping meanings. (Example: "learn learned") (2) They tend to lack adjective-noun structure that can be seen in the planning-based results. (Example: "bag ipod mac cellphone blackberry")

Adjective-noun structure in the DSAW-Plan output indicates that it may have imbued sequential/grammatical information in effect embedding that can be used to generate phrases. Furthermore, planning-based results tend to contain words that are not used by other naive methods. We interpret these results as an evidence that planners can provide a unique explanation from a perspective different from traditional embedding-based approaches. For example, in a paraphrase "cuba situation" for a target word "isolation", neither of the individual words are similar to the target word. However, DSAW-plan have identified that they create an additional layer of meaning when combined.

Word y	adventure	laptop
DSAW-Plan	classic trip; drama movie	personal mac device
CBOW-MILP	fun tale outdoor fantasy epic	bag mac ipod blackberry cellphone
CBOW-greedy	epic fantasy fun outdoor tale	bag ipod mac cellphone blackberry
CBOW-kNN	tale fun epic fantasy outdoor	cellphone bag blackberry ipod mac
DSAW-MILP	tale explore fantasy epic romance	iphone mac ipod blackberry ipad
DSAW-greedy	fantasy tale epic romance explore	ipad ipod mac iphone blackberry
DSAW-kNN	outdoor romance epic fantasy action	iphone ipad mac ipod cellphone
Word y	isolation	learnt
DSAW-Plan	cuba situation	learn yesterday
CBOW-MILP	international pressure diplomatic turmoil acute	how times ve learn learned
CBOW-greedy	acute diplomatic international ease pressure	learn times has believe chair
CBOW-kNN	international turmoil acute pressure diplomatic	learned times how learn ve
DSAW-MILP	extreme absence closure acute midst	learn taught teach lesson midst
DSAW-greedy	acute midst closure extreme absence	lesson learn midst taught forgotten
DSAW-kNN	abandon pressure international fear diplomatic	risen lesson teach learn taught
Word y	dominant	puerto
DSAW-Plan	position china	taiwan argentina *
CBOW-MILP	force strong position dynamic dominance	operation mexico juan costa rico
CBOW-greedy	dominance position force strong popular	costa rico mexico operation juan
CBOW-kNN	dominance strong force position dynamic	operation rico juan mexico costa
DSAW-MILP	position mainstream duo tribe dominance	del universe costa midst rico
DSAW-greedy	position dominance tribe duo mainstream	costa rico midst del universe
DSAW-kNN	dominance fragile position tribe volatile	hawaii rico vacation chile costa
Word y	explore	liquidity
DSAW-Plan	continue enjoy	boost cash ; guarantee cash
CBOW-MILP	discuss develop investigate pursue examine	capital cash crunch leverage volatility
CBOW-greedy	discuss examine investigate develop exploration	cash volatility leverage capital crunch
CBOW-kNN	develop examine investigate discuss pursue	cash crunch capital leverage volatility
DSAW-MILP	investigate examine assess evaluate inform	flow crunch excess squeeze ebitda
DSAW-greedy	evaluate examine investigate assess inform	crunch squeeze excess flow ebitda
DSAW-kNN	pursue investigate discuss examine develop	credit crunch adequate cash facility

Table 20: Comparison of different zero-shot definition construction mechanism. Note*: Puerto Rico and Taiwan are both islands; Puerto Rico and Argentina both speak spanish.

C.8 Details of the Vocabulary Pruning during the Paraphrasing Experiments

In Sec. 5.2, we noted that we removed certain words from the vocabulary used for paraphrasing. This section provides the detail of this filtering.

> A was generated from the 4000 most-frequent words in the vocabulary ($V \approx 219$k) excluding y, function words (e.g., "the", "make"), and compound words (e.g., plurals).

We implemented the multiple criteria as shown below. If any of the criteria is met, the generated paraphrase is not allowed to use the word.

1. The length of string for the word is less than equal to 2. This rules out bogus characters (typos) as well as basic words. Example: "a", "I", "z".
2. All characters are alphabetical; Any word that was accidentally left unprocessed during the tokenization will be pruned. Example: "them.Then" (failed tokenization), "#17" (a sharpsign and a number).
3. Function words and number-related words listed in Fig. 15.
4. Common suffix expressed by a regular expression `(ing | ed | ([^s] | se)s | ly | est | er | ness | ment | sion | ship | able | ible | ful | ous | ive | less | ise | ize)$`.
5. A set of PDDL keywords. This pruning is not essential, but is required as an ad-hoc enginnering solution to avoid the reserved words in PDDL. It consist of the following words: `and or not define problem domain when forall exists imply assign increase decrease object number at over always sometime within minimize maximize start end all preference nil`

a about above across after afterwards again against all almost alone along already also although always am among amongst amoungst an and another any anyhow anyone anything anyway anywhere are around as at be become becomes became because been before beforehand behind being below beside besides between beyond both but by can cannot could dare despite did do does done down during each eg either else elsewhere enough etc even ever every everyone everything everywhere except few first for former formerly fourth fifth sixth seventh eighth ninth tenth from further furthermore had has have he hence her here hereabouts hereafter hereby herein hereinafter heretofore hereunder hereupon herewith hers herself him himself his how however i ie eg if in indeed inside instead into is it its itself last latter latterly least less lot lots many may me meanwhile might mine more moreover most mostly much must my myself namely near need neither never nevertheless next no nobody none noone nor not nothing now nowhere of off often oftentimes on once one only onto or other others otherwise ought our ours ourselves out outside over per perhaps rather re same second several shall she should since so some somehow someone something sometime sometimes somewhat somewhere still such than that the their theirs them themselves then thence there thereabouts thereafter thereby thereforethus therein thereof thereon thereupon these they third this those though through throughout thru to together too top toward towards under until up upon us used very via was we well were what whatever when whence whenever where whereafter whereas whereby wherein whereupon wherever whether which while whither who whoever whole whom whose why whyever will with within without would yes yet you your yours yourself yourselves go goes gone went going say said says saying get got gets gotten getting give gave given giving make made makes making take took taken takes taking way ways reason due grant beneath help one two three four five six seven eight nine ten eleven twelve thirteen fourteen fifteen sixteen seventeen eighteen nineteen twenty thirty fourty fifty sixty seventy eighty ninty hundred thousand million billion double triple quadraple

Figure 15: List of function words and number-related words.

REFERENCES

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*, pp. 19, 2009.

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

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)*, 2018a.

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*, 2018b.

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.

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.

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*, pp. 178–186, 2013.

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*, pp. 4349–4357, 2016.

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`.

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

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

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

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.

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*, pp. 853–862, 2018.

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*, pp. 1724–1734, 2014.

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

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*, pp. 3519–3530, 2016.

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*, pp. 19–24, 2014.

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.

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

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*, pp. 406–414, 2001.

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*, pp. 1426–1436, 2014.

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

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

Patrik Haslum, Nir Lipovetzky, 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.

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

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.

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

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

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

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*, pp. 448–456, 2015.

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

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

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`.

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*, pp. 2369–2385, 2019.

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

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*, pp. 56–64, 2017.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. pp. 282–293, 2006.

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.

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

Ken Lang. Newsweeder: Learning to Filter Netnews. In *Proc. of the International Conference on Machine Learning*, pp. 331–339, 1995.

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

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.

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

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.

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.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

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.

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*, 2013a.

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*, pp. 3111–3119, 2013b.

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*, pp. 746–751, 2013c.

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

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=nbsOIJ_saUAC`.

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*, pp. 99–109, 2017.

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.

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

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.

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*, pp. 337–346, 2011.

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

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.

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)*, pp. 683–691, 2009.

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

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*, pp. 1631–1642, 2013.

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*, pp. 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*, pp. 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.

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*, pp. 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*, pp. 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*, pp. 1098–1107, 2018.