LEARNING MULTI-FACET EMBEDDINGS OF PHRASES AND SENTENCES USING SPARSE CODING FOR UNSUPERVISED SEMANTIC APPLICATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most deep learning for NLP represents each word with a single point or single-mode region in semantic space, while the existing multi-mode word embeddings cannot represent longer word sequences like phrases or sentences. We introduce a phrase representation (also applicable to sentences) where each phrase has a distinct set of multi-mode codebook embeddings to capture different semantic facets of the phrase's meaning. The codebook embeddings can be viewed as the cluster centers which summarize the distribution of possibly co-occurring words in a pre-trained word embedding space. We propose an end-to-end trainable neural model that directly predicts the set of cluster centers from the input text sequence (e.g., a phrase or a sentence) during test time. We find that the per-phrase/sentence codebook embeddings not only provide a more interpretable semantic representation but also outperform strong baselines (by a large margin in some tasks) on benchmark datasets for unsupervised phrase similarity, sentence similarity, hypernym detection, and extractive summarization.

1 INTRODUCTION

Many widely-applicable NLP models learn a representation from only co-occurrence statistics in the raw text without any supervision. Examples include word embedding like word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014), sentence embeddings like skip-thoughts (Kiros et al., 2015), and contextualized word embedding like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). Most of these models use a single embedding to represent one sentence or one phrase and can only provide symmetric similarity measurement when no annotation is available.

However, a word or phrase might have multiple senses, and a sentence can involve multiple topics, which are hard to analyze based on a single embedding without supervision. To address the issue, word sense induction methods (Lau et al., 2012) and recent multi-mode word embeddings (Neelakantan et al., 2014; Athiwaratkun & Wilson, 2017; Singh et al., 2018) represent each target word as multiple points or regions in a distributional semantic space by (explicitly or implicitly) clustering all the words appearing beside the target word.

In Figure 1, the multi-mode representation of *real property* is illustrated as an example. Real property can be observed in legal documents where it usually means a real estate, while a real property can also mean a true characteristic in philosophic discussions. The previous approaches discover those senses by clustering observed neighboring words (e.g., *company* and *tax*). In contrast with topic modeling like LDA (Blei et al., 2003), the approaches need to solve a distinct clustering problem for every target word while topic modeling finds a single set of clusters by clustering all the words in the corpus.

Extending these multi-mode representations to arbitrary sequences like phrases or sentences is difficult due to two efficiency challenges. First, there are usually many more unique phrases and sentences in a corpus than there are words, while the number of parameters for clustering-based approaches is $O(|V| \times |K| \times |E|)$, where $|V|$ is number of unique sequences, $|K|$ is number of modes/clusters, and $|E|$ is the number of embedding dimensions. Estimating and storing such a large number of parameters take time and space. More important, many unique sequences imply much fewer co-occurring words to be clustered for each sequence, especially for long sequences

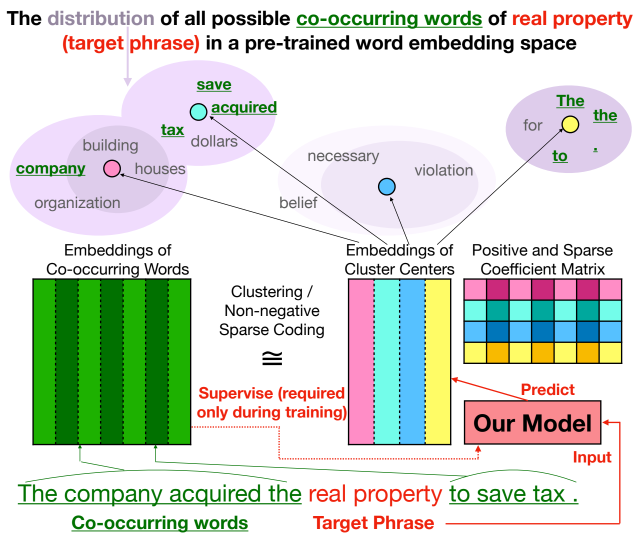
The distribution of all possible co-occurring words of real property (target phrase) in a pre-trained word embedding space
save
acquired
tax
dollars
building
company
houses
organization
necessary
violation
belief
The
the
for
to
.
Embeddings of Co-occurring Words
Clustering / Non-negative Sparse Coding
≅
Embeddings of Cluster Centers
Positive and Sparse Coefficient Matrix
Supervise (required only during training)
Predict
Our Model
Input
The company acquired the real property to save tax .
Co-occurring words
Target Phrase

Figure 1: The target phrase *real property* is represented by four clustering centers. The previous work discovers the four modes by finding clustering centers which well compress the embedding of observed co-occurring words. Instead, our compositional model learns to predict the embeddings of cluster centers from the sequence of words in the target phrase so as to reconstruct the (unseen) co-occurring distribution well.

like sentences, so an effective model needs to overcome this sample efficient challenge (i.e., sparseness in the co-occurring statistics). However, clustering approaches often have too many parameters to learn the compositional meaning of each sequence without overfitting.

Nevertheless, the sentences (or phrases) sharing multiple words tend to have similar cluster centers, so we should be able to compress many redundant parameters in these local clustering problems to circumvent the challenges. In this work, we adopt a neural encoder and decoder to achieve the goal. As shown in Figure 1, instead of clustering co-occurring words beside a target sequence at test time as in previous approaches, we learn a mapping between the target sequence (i.e., phrases or sentences) and the corresponding cluster centers during training so that we can directly predict those cluster centers using a single forward pass of the neural network for an arbitrary unseen input sequences during testing.

To allow the neural network to generate the cluster centers in an arbitrary order, we use a non-negative and sparse coefficient matrix to dynamically match the sequence of predicted cluster centers and the observed set of co-occurring word embeddings during training. After the coefficient matrix is estimated for each input sequence, the gradients are back-propagated to cluster centers (i.e., codebook embeddings) and weights of decoder and encoder, which allows us to train the whole model jointly and end-to-end.

In experiments, we show that the proposed model captures the compositional meanings of words in unsupervised phrase similarity tasks much better than averaging their (contextualized) word embeddings, strong baselines that are widely used in practice. In addition to similarity, our model can also measure asymmetric relations like hypernymy without any supervision. Furthermore, the multi-mode representation is shown to outperform the single-mode alternatives in sentence representation, especially as demonstrated in our extractive summarization experiment.

2 METHOD

In this section, we first formalize our training setup in Section 2.1. Next, our objective function and the architecture of our prediction mode will be described in Section 2.2 and Section 2.3, respectively. The approach is summarized in Figure 2 using an example sentence.

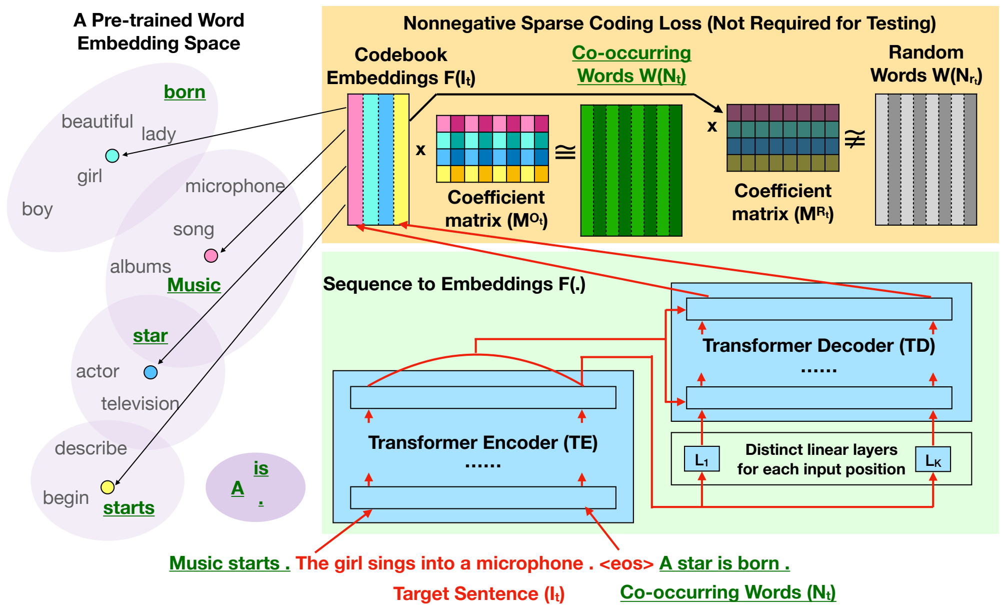
A Pre-trained Word Embedding Space
born
beautiful
lady
girl
boy
microphone
song
albums
Music
star
actor
television
describe
begin
starts
is
A
.
Nonnegative Sparse Coding Loss (Not Required for Testing)
Codebook Embeddings F(I_t)
Co-occurring Words W(N_t)
Random Words W(N_{r_t})
x
≅
≇
Coefficient matrix (M^{O_t})
Coefficient matrix (M^{R_t})
Sequence to Embeddings F(.)
Transformer Decoder (TD)
Transformer Encoder (TE)
Distinct linear layers for each input position
L_1
L_K
Music starts . The girl sings into a microphone . <eos> A star is born .
Target Sentence (I_t)
Co-occurring Words (N_t)

Figure 2: Our model for sentence representation. We represent each sentence as multiple codebook embeddings (i.e., clustering centers) predicted by our sequence to embeddings model. Our loss encourages the model to generate codebook embeddings whose linear combination can well reconstruct the embeddings of co-occurring words (e.g., *Music*), while not able to reconstruct the negatively sampled words (i.e., the co-occurring words from other sentences) to avoid predicting common topics which co-occur with every sentence (e.g., *is* in this example).

2.1 Self-supervision Signal

We express tth sequence of words in the corpus as $I_t = w_{x_t}...w_{y_t}$<eos>, where x_t and y_t are the start and end position of the target sequence, respectively, and <eos> is the end of sequence symbol.

We assume neighboring words beside each target phrase or sentence are related to some aspects of the sequence, so given I_t as input, our training signal is to reconstruct a set of neighboring words, $N_t = \{w_{x_t-d_1^t}, ...w_{x_t-1}, w_{y_t+1}, ...w_{y_t+d_2^t}\}$.[1] For phrase representation, $d_1^t = d_2^t$ is a fixed window size. For sentence representation, N_t is the set of all words in the previous and the next sentence. Notice that the training signal for sentences and phrases are different, which means we need to train one model for phrase and one model for sentence if both representations are desired.

Since there are not many co-occurring words for a long sequence (none are observed for unseen testing sequences), our goal is to cluster the set of words which could "possibly" occur beside the sequence instead of the actual occurring words in the training corpus (e.g., the hidden co-occurring distribution instead of green and underlined words in Figure 2). Although most of the possibly co-occurring words are not observed in the corpus, we can still learn to predict them by observing co-occurring words from similar sequences.

To focus on the semantics rather than syntax, we view the co-occurring words as a set rather than a sequence as in skip-thoughts (Kiros et al., 2015). Notice that our model considers the word order information in the input sequence I_t, but ignores the order of the co-occurring words N_t.

2.2 Non-negative Sparse Coding Loss

In this work, we model the distribution of co-occurring words in a pre-trained word embedding space. The embeddings of co-occurring words N_t are arranged into a matrix $\boldsymbol{W}(\boldsymbol{N_t}) =$

[1]The self-supervised signal is a generalization of the one used in prediction-based word embedding like Word2Vec (Mikolov et al., 2013). They are the same when the length of input sequence $|I_t|$ is 1.

$[\underline{\boldsymbol{w}}_j^t]_{j=1...|N_t|}$ with size $|E| \times |N_t|$, where $|E|$ is the dimension of pre-trained word embedding, and each of its column $\underline{\boldsymbol{w}}_j^t$ is a normalized word embedding whose 2-norm is 1. The normalization makes the cosine distance between two words become half of their Euclidean distance.

Similarly, we denote the predicted cluster centers $\underline{\boldsymbol{c}}_k^t$ of the input sequence I_t as a $|E| \times K$ matrix $\boldsymbol{F(I_t)} = [\underline{\boldsymbol{c}}_k^t]_{k=1...K}$, where $\boldsymbol{F}$ is our neural network model and K is the number of clusters. We fix the number of clusters K in this work to simplify the design of our prediction model and how it is applied to downstream tasks. The effect of different cluster numbers will be discussed in the experimental section.

The reconstruction loss of k-means clustering in the word embedding space (Singh et al., 2018) can be written as $||\boldsymbol{F(I_t)M} - \boldsymbol{W(N_t)}||^2 = \sum_j ||(\sum_k \boldsymbol{M}_{k,j}\underline{\boldsymbol{c}}_k^t) - \underline{\boldsymbol{w}}_j^t||^2$, where $\boldsymbol{M}_{k,j} = 1$ if the jth word belongs to the k cluster and 0 otherwise. That is, $\boldsymbol{M}$ is a permutation matrix which matches the cluster centers and co-occurring words and allow the neural network to generate the centers in an arbitrary order.

Non-negative sparse coding (NNSC) (Hoyer, 2002) relaxes the constraints by allowing the coefficient $\boldsymbol{M}_{k,j}$ to be a positive value but encouraging it to be 0. In this work, we adopt the relaxation because for all neural architectures (including transformers and LSTMs) we tried, the models using a NNSC loss learn to generates diverse K cluster centers while the predicted cluster centers using the kmeans loss collapse to much fewer modes which cannot well capture the conditional co-occurrence distribution. We hypothesize that it is because a NNSC loss is smoother and easier to be optimized for a neural network, while finding the nearest cluster center in the kmeans loss cannot stably encourage predicted clusters to play different roles for reconstructing the embeddings of observed co-occurring words.

Using NNSC, we define our reconstruction error as

$$\begin{aligned} &Er(\boldsymbol{F(I_t)}, \boldsymbol{W(N_t)}) = ||\boldsymbol{F(I_t)M^{O_t}} - \boldsymbol{W(N_t)}||^2 \\ &s.t., \boldsymbol{M^{O_t}} = \underset{\boldsymbol{M}}{\arg\min} \, ||\boldsymbol{F(I_t)M} - \boldsymbol{W(N_t)}||^2 + \lambda||\boldsymbol{M}||_1, \; \forall k,j, \; 0 \leq \boldsymbol{M}_{k,j} \leq 1, \end{aligned} \tag{1}$$

where λ is a hyper-parameter controlling the sparsity of $\boldsymbol{M}$. We force the coefficient value $\boldsymbol{M}_{k,j} \leq 1$ to avoid the neural network learning to predict centers with small magnitudes which makes the optimal values of $\boldsymbol{M}_{k,j}$ large and unstable.

Having multiple outputs and estimating the permutation between the prediction and ground truth words is often computationally expensive (Stern et al., 2018; Qin et al., 2019). However, the proposed loss is efficient because we minimize the L2 distance in a pre-trained embedding space as in Kumar & Tsvetkov (2019) rather than using softmax, and $\boldsymbol{M^{O_t}}$ can be efficiently estimated on the fly using convex optimization (we use RMSprop (Tieleman & Hinton, 2012) in our implementation). After $\boldsymbol{M^{O_t}}$ is estimated, we can treat it as a constant and back-propagate the gradients to our neural network to achieve end-to-end training.

To prevent the neural network from predicting the same global topics regardless of the input, our loss function for tth sequence is defined as

$$L_t(\boldsymbol{F}) = Er(\boldsymbol{F(I_t)}, \boldsymbol{W(N_t)}) - Er(\boldsymbol{F(I_t)}, \boldsymbol{W(N_{r_t})}), \tag{2}$$

where N_{r_t} is a set of co-occurring words of a randomly sampled sequence I_{r_t}. In our experiment, we use SGD to solve $\widehat{\boldsymbol{F}} = \arg\min_{\boldsymbol{F}} \sum_t L_t(\boldsymbol{F})$. Our method could be viewed as a generalization of Word2Vec (Mikolov et al., 2013) that can encode the compositional meaning of the words and decode multiple embeddings.

2.3 Sequence to Embeddings

Our neural network architecture is similar to transformation-based sequence to sequence (seq2seq) model (Vaswani et al., 2017). We use the same encoder $TE(I_t)$, which transforms the input sequence into a contextualized embeddings

$$TE(w_{x_t}...w_{y_t}\text{<eos>}) = \underline{\boldsymbol{e}}_{x_t}...\underline{\boldsymbol{e}}_{y_t}\underline{\boldsymbol{e}}_{\text{<eos>}}, \tag{3}$$

where the goal of the encoder is to map the sentences which are likely to have similar co-occuring word distribution closer together.

Table 1: Examples of the codebook embeddings which are predicted by models with different K given an input phrase or an unseen input sentence. The embedding in each row is visualized by the three words whose GloVe embeddings are closest to the codebook embedding (in terms of cosine distances) and their corresponding cosine similarities.

Target Phrase: civil order <eos>	
Output Embedding (K = 1):	
e1	government 0.817 civil 0.762 citizens 0.748
Output Embeddings (K = 3):	
e1	initiatives 0.736 organizations 0.725 efforts 0.725
e2	army 0.815 troops 0.804 soldiers 0.786
e3	court 0.758 federal 0.757 judicial 0.736

Target Sentence: SMS messages are used in some countries as reminders of hospital appointments . <eos>	
Output Embedding (K = 1):	
e1	information 0.702, use 0.701, specific 0.700
Output Embeddings (K = 3):	
e1	can 0.769, possible 0.767, specific 0.767
e2	hospital 0.857, medical 0.780, hospitals 0.739
e3	SMS 0.791, Mobile 0.635, Messaging 0.631
Output Embeddings (K = 10):	
e1	can 0.854, should 0.834, either 0.821
e2	hospital 0.886, medical 0.771, hospitals 0.745
e3	services 0.768, service 0.749, web 0.722
e4	SMS 0.891, sms 0.745, messaging 0.686
e5	messages 0.891, message 0.801, emails 0.679
e6	systems 0.728, technologies 0.725, integrated 0.723
e7	appointments 0.791, appointment 0.735, duties 0.613
e8	confirmation 0.590, request 0.568, receipt 0.563
e9	countries 0.855, nations 0.737, Europe 0.732
e10	Implementation 0.613, Application 0.610, Programs 0.603

Different from the typical seq2seq model (Sutskever et al., 2014; Vaswani et al., 2017), our decoder does not need to make discrete decisions because our outputs are a sequence of embeddings instead of words. This allows us to predict all the codebook embeddings in a single forward pass while still well capturing the dependency between output without the need of auto-regressive decoding. Similar to BERT, we treat the embedding of <eos> as the sentence representation. To make different codebook embeddings capture different aspects, we pass the embeddings of <eos> to different linear layers L_k before becoming the input of the decoder TD. Specifically, the codebook embeddings

$$\boldsymbol{F}(\boldsymbol{I_t}) = TD(L_1(\underline{\boldsymbol{e}}_{\text{<eos>}})...L_K(\underline{\boldsymbol{e}}_{\text{<eos>}}), \underline{\boldsymbol{e}}_{x_t}...\underline{\boldsymbol{e}}_{y_t}). \tag{4}$$

We find that removing the attention on the $\underline{\boldsymbol{e}}_{x_t}...\underline{\boldsymbol{e}}_{y_t}$, contextualized word embeddings from the encoder, significantly increases our validation loss for sentence representation because there are often too many facets to be compressed into a single embedding. On the other hand, the attention does not change the performance of phrase representation too much, and we remove the attention connection between encoder and decoder (i.e., encoder and decoder have the same architecture) when evaluating our phrase representation.

Notice that the framework is flexible. We can replace the encoder and decoder with other architectures. Besides transformers, we also try (bi-)LSTMs in our experiments. This flexibility also allows us to incorporate other input features (e.g., the author who writes the sentence).

3 Experiments

We first visualize the cluster centers predicted by our model in Table 1 (like we visualize the meaning of the red cluster center in Figure 2 using the word *song* or *Music*). The centers summarize the target sequence well and more codebook embeddings capture more semantic facets of a phrase or a sentence. Due to the difficulty of evaluating the topics conditioned on the input sequence using the classic metrics for global topic modeling, we show that the codebook embeddings can be used to improve the performances of various unsupervised semantic tasks, which indirectly measures the quality of the generated topics.

Table 2: Performance of phrase similarity tasks. Every model is trained on a lowercased corpus. In SemEval 2013, AUC (%) is the area under the precision-recall curve of classifying similar phrase pairs. In Turney, we report the accuracy (%) of predicting the correct similar phrase pair among 5 or 10 candidate pairs. In BiRD and WikiSRS, the correlation coefficient (%) between predicted similarity and ground truth similarity is presented. *The results are taken from Yu & Dredze (2015).

Method		SemEval 2013		Turney (5)	Turney (10)	BiRD	WikiSRS-Sim	WikiSRS-Rel
Model	Score	AUC	F1	Accuracy	Accuracy	Pearson	Spearman	
BERT	CLS	54.7	66.7	29.2	15.5	17.2	5.4	4.2
	Avg	66.5	67.1	43.4	24.3	44.4	43.3	40.7
GloVe	Avg	79.5	73.7	25.9	12.9	47.9[4]	47.7	49.1
Word2Vec	Avg	-	65.5*	39.6*	19.8*	-	-	-
FCT LM	Emb	-	67.2*	42.6*	27.6*	-	-	-
Ours (K=10)	SC	80.3	72.8	45.6	28.8	49.9	58.9	52.5
	Emb	85.6	77.1	49.4	31.8	57.3	**67.4**	**66.9**
Ours (K=1)	SC	81.1	72.7	45.3	28.4	47.3	59.5	55.2
	Emb	**87.8**	**78.6**	**50.3**	**32.5**	**60.6**	67.1	65.8

We use the cased version (840B) of pre-trained GloVe embedding (Pennington et al., 2014) for sentence representation and use the uncased version (42B) for phrase representation.[2] Our model is trained on Wikipedia 2016 while the stop words are removed from the set of co-occurring words.

In the phrase experiments, we only consider noun phrases, and their boundaries are extracted by applying simple regular expression rules to POS tags before training. The sentence boundaries and POS tags are detected using spaCy.[3] Our models do not need resources such as PPDB (Pavlick et al., 2015) or other multi-lingual resources, so our models are compared with the baselines that only use the raw text and sentence/phrase boundaries. This setting is particularly practical for the domains with low resources such as scientific literature.

To control the effect of embedding size, we set the number of dimensions in our transformers as the GloVe embedding size (300). Limited by computational resources, we train all the models using one modern GPU within a week. Because of the relatively small model size, we find that our models underfit the data after a week (i.e., the training loss is very close to the validation loss).

It is hard to make a fair comparison with BERT (Devlin et al., 2019). BERT is trained on a masked language modeling loss, which preserves more syntax information and can produce an effective pre-trained embedding for many supervised downstream tasks. We report the performance of the BERT base model, which is still trained using more parameters, more output dimensions, larger corpus, and more computational resources during training compared with our models. Furthermore, BERT uses a word piece model to alleviate the out-of-vocabulary problem. Nevertheless, we still provide its unsupervised performances based on cosine similarity as a reference.

3.1 Phrase Similarity

Semeval 2013 task 5(a) English (Korkontzelos et al., 2013) and Turney 2012 (Turney, 2012) are two standard benchmarks for evaluating phrase similarity (Yu & Dredze, 2015; Huang et al., 2017). BiRD (Asaadi et al., 2019) and WikiSRS (Newman-Griffis et al., 2018), which are recently collected, contain ground truth phrase similarities derived from human annotations.

The task of Semeval 2013 is to distinguish similar phrase pairs from dissimilar phrase pairs. In Turney (5), given each query bigram, the goal is to identify the unigram that is most similar to the query bigram among 5 unigram candidates,[5] and Turney (10) adds 5 more negative phrase pairs by pairing the reverse of bigram with the 5 unigrams. BiRD and WikiSRS-Rel measure the relatedness of phrases and WikiSRS-Sim measures the similarity of phrases.

[2] `nlp.stanford.edu/projects/glove/`

[3] `spacy.io/`

[4] The number is different from the one reported in Asaadi et al. (2019) because we use the uncased version (42B), the embedding space our model is trained on, and they use the cased version (840B).

[5] We follow the evaluation setup of Yu & Dredze (2015); Huang et al. (2017), which ignores two unigram candidates being contained in the target phrase, because the original setup (Turney, 2012) is too difficult for unsupervised methods to get a meaningful score (e.g., the accuracy of **GloVe Avg** is 0 in the original setting).

For our model, we evaluate two scoring functions that measure phrase similarity. The first way averages the contextualized word embeddings from our transformer encoder as the phrase embedding and computes the cosine similarity between two phrase embeddings. We label the method as **Ours Emb**. The similar phrases should have similar multi-facet embeddings, so we compute the reconstruction error from the set of normalized codebook embeddings of one phrase S_q^1 to the embeddings of the other phrase S_q^2, vice versa, and add them together to become a symmetric distance **SC**:

$$SC(S_q^1, S_q^2) = Er(\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S_q^1}), \widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S_q^2})) + Er(\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S_q^2}), \widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S_q^1})), \tag{5}$$

where $\widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S_q}) = [\frac{\underline{\boldsymbol{c}}_k^q}{||\underline{\boldsymbol{c}}_k^q||}]_{k=1...K}$. When ranking for retrieving similar phrases, we use the negative distance to represent similarity.

We compare our performance with 5 baselines. **GloVe Avg** and **Word2Vec Avg** compute the cosine similarity between two averaged word embeddings, which has been shown to be a strong baseline (Asaadi et al., 2019). **BERT CLS** and **BERT Avg** are the cosine similarities between CLS embeddings and between the averages of contextualized word embeddings from BERT (Devlin et al., 2019), respectively. **FCT LM Emb** (Yu & Dredze, 2015) learns the weights of linearly combining word embeddings based on several linguistic features.

The performances are presented in Table 2. SemEval 2013 and Turney have training and testing split and the performances in test sets are reported. Our models significantly outperform all baselines in the 4 datasets. Furthermore, our strong performances in Turney (10) verify that our encoder incorporates the word order information when producing phrase embeddings. The results indicate the effectiveness of non-linearly composing word embeddings (unlike GloVe, Word2Vec, and FCT baselines) in order to predict the set of co-occurring word embeddings (unlike the BERT baselines).

The performance of **Ours (K=1)** is usually slightly better than **Ours (K=10)**. This result supports the finding of Dubossarsky et al. (2018) that multi-mode embeddings may not improve the performance in word similarity benchmarks even if they capture more senses or aspects of polysemies. Even though being slightly worse, the performances of **Ours (K=10)** remain strong compared with baselines. This indicates that the similarity performance is not sensitive to the number of clusters, which alleviates the problem of selecting K in practice.

3.2 Sentence Similarity

STS benchmark (Cer et al., 2017) is a widely used sentence similarity task. Each model predicts a semantic similarity score between each sentence pair, and the scores are compared with average similarity annotations using the Pearson correlation coefficient.

Intuitively, when two sentences are less similar to each other, humans tend to judge the similarity based on how they are similar in each aspect. Thus, we also compare the performances on lower half the datasets where their ground truth similarities are less than the median similarity score, and we call this benchmark STSB Low.

In addition to **BERT CLS**, **BERT Avg**, and **GloVe Avg**, we compare our method with word mover's distance (**WMD**) (Kusner et al., 2015) and cosine similarity between skip-thought embeddings (**ST Cos**) (Kiros et al., 2015). Recently, Arora et al. (2017) propose to weight the word w in each sentence according to $\frac{\alpha}{\alpha + p(w)}$, where α is a constant and $p(w)$ is the probability of seeing the word w in the corpus. Following its recommendation, we set α to be 10^{-4} in STS benchmark. After the weighting, we adopt the post-processing method from Arora et al. (2017) to remove the first principal component that is estimated using the training distribution and denote the method as **GloVe SIF**. The post-processing is not desired in some applications (Singh et al., 2018), so we also report the performance before removing principal components, which is called **GloVe Prob_avg**.

The strong performance of (weighted) average embedding (Milajevs et al., 2014; Arora et al., 2017) suggests that we should consider the embeddings of words in the sentence in addition to the sentence embedding(s) when measuring the sentence similarity. This is hard to be achieved in other sentence representation methods because their sentence embeddings and word embeddings are in different semantic spaces.

Since our multi-facet embeddings are in the same space of word embeddings, we can use the multi-facet embeddings to estimate the word importance (in terms of predicting possible co-occurring

Table 3: Pearson correlation (%) in the development and test sets in the STS benchmark. The performances of all sentence pairs are indicated as All. Low means the performances on the half of sentence pairs with lower similarity (i.e., STSB Low). † indicates the methods which use training distribution to approximate testing distribution. The best score with and without † are highlighted.

Method		Dev		Test	
Model	Score	All	Low	All	Low
ST	Cos	43.2	28.1	30.4	21.2
BERT	CLS	9.6	-0.4	4.1	0.2
	Avg	62.3	42.1	51.2	39.1
GloVe	WMD	58.8	35.3	40.9	25.4
	Avg	51.6	32.8	36.6	30.9
	Prob_avg	70.7	56.6	59.2	54.8
	SIF†	75.1	65.7	63.2	58.1
Ours K=1	SC	55.7	43.7	47.6	45.4
	Avg	54.5	40.2	44.1	40.6
	Prob_avg	68.5	56.0	58.1	55.2
	SIF†	72.5	64.0	61.7	58.5
Ours LSTM EN-DE K=10	SC	58.9	49.2	49.8	46.4
	Avg	60.4	45.2	46.8	42.8
	Prob_avg	71.7	60.1	61.3	58.3
	SIF†	74.6	66.9	64.3	60.9
Ours K=10	SC	63.0	51.8	52.6	47.8
	Avg	61.7	47.1	50.0	46.5
	Prob_avg	**72.0**	**60.5**	**61.4**	**59.3**
	SIF†	**75.2**	**67.6**	**64.6**	**62.2**

Table 4: Hypernym detection performances in the development and test set of HypeNet. AUC (%) measures the quality of retrieving hypernym phrases. Acc (%) means the accuracy of predicting specificity given a pair of hypernym phrases.

Method	Dev		Test	
	AUC	Acc	AUC	Acc
BERT (CLS)	20.6	0.5	21.3	0.5
BERT (Avg)	25.6	0.5	25.6	0.5
GloVe (Avg)	17.4	0.5	17.7	0.5
Ours (K=10)	**29.4**	78.9	**29.6**	79.1
Ours (K=1)	29.3	**82.7**	**29.6**	**81.0**

Table 5: The ROUGE F1 scores of different methods on CNN/Daily Mail dataset. The results with † are taken from Zheng & Lapata (2019). The results with * are taken from Celikyilmaz et al. (2018).

Setting	Method	R-1	R-2	Len
Unsup, No Order	Random	28.1	8.0	68.7
	Textgraph (tfidf)†	33.2	11.8	-
	Textgraph (BERT)†	30.8	9.6	-
	W Emb (GloVe)	26.6	8.8	40.0
	Sent Emb (GloVe)	32.6	10.7	78.2
	W Emb (BERT)	31.2	11.2	44.9
	Sent Emb (BERT)	32.3	10.6	91.2
	Ours (K=3)	32.2	10.1	75.4
	Ours (K=10)	34.0	11.6	81.3
	Ours (K=100)	**35.0**	**12.8**	92.9
Unsup	Lead-3	40.3	17.6	87.0
	PACSUM (BERT)†	**40.7**	**17.8**	-
Sup	RL*	**41.7**	**19.5**	-

words beside the sentence). To compute the importance of a word in the sentence, we first compute the cosine similarity between the word and all predicted codebook embeddings, truncate the negative similarity to 0, and sum all similarity. Specifically, our simple importance/attention weighting for all the words in the query sentence S_q is defined by

$$\underline{\boldsymbol{a}}_{\boldsymbol{q}} = \max(0, \boldsymbol{W}(\boldsymbol{S_q})^T \widehat{\boldsymbol{F}}_{\boldsymbol{u}}(\boldsymbol{S_q}))\ \underline{\mathbf{1}}, \tag{6}$$

where $\underline{\mathbf{1}}$ is an all-one vector. The importance weighting is multiplied with the original weighting vectors on **GloVe Avg** (uniform weight), **GloVe Prob_avg**, and **GloVe SIF** to generate the results of **Our Avg**, **Our Prob_avg**, and **Our SIF**, respectively.

We compare all the results in the development set and test set in Table 3. **Ours SC**, which matches between two sets of topics outperforms **WMD**, which matches between two sets of words in the sentence, and also outperforms **BERT Avg**, especially in STSB Low. All the scores in **Ours (K=10)** are significantly better than **Ours (K=1)** ,which demonstrates the benefits of multi-mode representation. Multiplying the proposed attention weighting boosts the performance from (weighted) averaging-based methods especially in STSB Low and when we do not rely on the generalization assumption of our training distribution.

We also test a variant of our method which uses a bi-LSTM as the encoder and a LSTM as the decoder. Its average validation loss (-0.1289) in equation 2 is significantly higher than that of the transformer alternative (-0.1439), and it performances on Table 3 are also worse. Notice the architecture of this variant is similar to skip-thoughts except that skip-thoughts decodes a sequence instead of a set. The variant significantly outperforms **ST Cos**, which further justifies our approach of ignoring the order of co-occurring words in our NNSC loss.

3.3 Hypernymy Detection

We apply our model to HypeNet (Shwartz et al., 2016), an unsupervised hypernymy detection dataset, based on an assumption that the co-occurring words of a phrase are often less related to some of its hyponyms. For instance, *fly* is a co-occurring word of *animal* which is less related to *brown dog*.

Thus, the predicted codebook embeddings of a hyponym S_q^{hypo} (e.g., *brown dog*), which cluster the embeddings of co-occurring words (e.g., *eats*), often reconstruct the embeddings of its hypernym S_q^{hyper}(e.g., *animal*) better than the other way around (i.e., $Er(\boldsymbol{\widehat{F}_u}(\boldsymbol{S_q^{hypo}}), \boldsymbol{W}(\boldsymbol{S_q^{hyper}}))$ is smaller than $Er(\boldsymbol{\widehat{F}_u}(\boldsymbol{S_q^{hyper}}), \boldsymbol{W}(\boldsymbol{S_q^{hypo}}))$).

Based on the assumption, our asymmetric scoring function is defined as

$$Asym(S_q^{hyper}, S_q^{hypo}) = Er(\boldsymbol{\widehat{F}_u}(\boldsymbol{S_q^{hyper}}), \boldsymbol{W}(\boldsymbol{S_q^{hypo}})) - Er(\boldsymbol{\widehat{F}_u}(\boldsymbol{S_q^{hypo}}), \boldsymbol{W}(\boldsymbol{S_q^{hyper}})). \quad (7)$$

The AUC of detecting hypernym among other relations and accuracy of detecting the hypernym direction are compared in Table 4. Our methods outperform baselines, which only provide symmetric similarity measurement, and **Ours (K=1)** performs similarly compared with **Ours (K=10)**.

3.4 Extractive Summarization in a Single Document

A good summary should cover multiple aspects that well represent all topics/concepts in the whole document. The objective can be quantified as discovering a summary A with a set of normalized embeddings $C(A)$ which best reconstructs the distribution of normalized word embedding $\underline{\boldsymbol{w}}$ in the document D (Kobayashi et al., 2015). That is,

$$\arg\max_A \sum_{\underline{\boldsymbol{w}} \in D} \gamma_w \max_{\underline{\boldsymbol{s}} \in C(A)} \underline{\boldsymbol{w}}^T \underline{\boldsymbol{s}}, \quad (8)$$

where γ_w is the importance of the word w, which is set as $\frac{\alpha}{\alpha + p(w)}$ as we did in Section 3.2. We further assume that the summary A consists of T sentences A_1 ... A_T and the embedding set of the summary is the union of the embedding sets of the sentences $C(A) = \cup_{t=1}^T C(A_t)$, and we greedily select sentences to optimize equation 8 as did in Kobayashi et al. (2015).

This extractive summarization method provides us a way to evaluate the embedding(s) of sentence. Our model can generate multiple codebook embeddings, which capture its different aspects as we see in Table 1, to represent each sentence in the document, so we let $C(A_t) = \{\widehat{F}_u(A_t)\}$, a set of column vectors in the matrix $\widehat{F}_u(A_t)$.

We compare our approach with other alternative ways of modeling the aspects of sentences. For example, we can compute average word embeddings as a single-aspect sentence embedding. This baseline is labeled as **Sent Emb**. We can also use the embedding of all the words in the sentences as different aspects of the sentences. Since longer sentences have more words, we normalize the gain of the reconstruction loss by the sentence length. The method is denoted as **W Emb**. In contrast, the fixed number of codebook embeddings in our method avoids the problem. We also test the baselines of selecting random sentences (**Rnd**) and first n sentences (**Lead**) in the document.

The results on the testing set of CNN/Daily Mail (Hermann et al., 2015; See et al., 2017) are compared using F1 of ROUGE (Lin & Hovy, 2003) in Table 5. R-1, R-2, and Len mean ROUGE-1, ROUGE-2, and average summary length, respectively. All methods choose 3 sentences by following the setting in Zheng & Lapata (2019). Unsup, No Order means the methods do not use the sentence position information in the documents.

In CNN/Daily Mail or other English news corpora, the sentence order information is a very strong signal. For example, the unsupervised methods such as **Lead-3** are very strong baselines (Bohn & Ling, 2018) with performances similar to supervised methods such as RL (Celikyilmaz et al., 2018), a state-of-the-art approach in this evaluation. To evaluate the quality of unsupervised sentence embeddings, we focus on comparing the unsupervised methods which do not assume the first few sentences form a good summary.

In Table 5, predicting more aspects (i.e., using higher cluster numbers K) yields better results, and setting K = 100 gives us the best performance after selecting 3 sentences. This demonstrates that larger cluster numbers K is desired in this application. Our method allows us to set K to be a relatively large number because of greatly alleviating the computational and sample efficiency challenges.

4 Related Work

Topic modeling (Blei et al., 2003) has been extensively studied and widely applied due to its interpretability and flexibility of incorporating different forms of input features (Mimno & McCallum, 2008). Cao et al. (2015); Srivastava & Sutton (2017) demonstrate that neural networks could be applied to discover semantically coherent topics. However, instead of optimizing a global topic model, our goal is to jointly and efficiently discovering different sets of topics/clusters on the small subsets of words that co-occur with target phrases or sentences.

Sparse coding on word embedding space is used to model the multiple aspects of a word (Faruqui et al., 2015; Arora et al., 2018), and parameterizing word embeddings using neural networks is used to test hypothesis (Han et al., 2018) and save storage space (Shu & Nakayama, 2018). Besides, to capture asymmetric relations such as entailment, words are represented as single or multiple regions in Gaussian embeddings (Vilnis & McCallum, 2015; Athiwaratkun & Wilson, 2017) rather than a single point. However, the challenges of extending these methods to longer sequences are not addressed in these studies.

One of our main challenges is to design a neural decoder for a set rather a sequence while modeling the dependency between the elements. This requires a matching step between two sets and compute the distance loss after the matching (Eiter & Mannila, 1997). One popular loss is called Chamfer distance, which is widely adopted in the auto-encoder models for point clouds (Yang et al., 2018; Liu et al., 2019), while more sophisticated matching loss options are also proposed (Stewart et al., 2016; Balles & Fischbacher, 2019). The goal of the studies focus on measuring symmetric distances between the ground truth set and predicted set (usually with the equal size), while our set decoder tries to reconstruct a set using a set of much fewer bases.

Other ways to achieving the permutation invariants loss for neural networks includes removing the elements in the ground truth set which have been predicted (Welleck et al., 2018), beam search (Qin et al., 2019), or predicting the permutation using a CNN (Rezatofighi et al., 2018), a transformer (Stern et al., 2019; Gu et al., 2019) or reinforcement learning (Welleck et al., 2019). In contrast, our goal is to efficiently predict a set of clustering centers that can well reconstruct the set of observed instances instead of predicting the set or sequence of the observed instances.

5 Conclusions and Future Work

In this work, we overcome the computational and sampling efficiency challenges of learning the multi-mode representation for long sequences like phrases or sentences. We use a neural encoder to model the compositional meaning of the target sequence and use a neural decoder to predict a set of codebook embeddings as the representation of the sentences or phrases. During training, we use a non-negative sparse coefficient matrix to dynamically match the predicted codebook embeddings to a set of observed co-occurring words and allow the neural decoder to predict the clustering centers with an arbitrary permutation.

We demonstrate that the proposed models can learn to predict interpretable clustering centers conditioned on an (unseen) sequence, and the representation outperforms widely-used baselines such as BERT, skip-thoughts and various approaches based on GloVe in several unsupervised benchmarks. The experimental results also suggest that multi-facet embeddings perform the best when the input sequence (e.g., a sentence) involves many aspects, while multi-facet and single-facet embeddings perform similarly good when the input sequence (e.g., a phrase) usually involves only one aspect.

In the future, we would like to train a single model which could generate multi-facet embeddings for both phrases and sentences, and evaluate the method as a pre-trained embedding approach for supervised or semi-supervised settings. Furthermore, we plan to apply this method to other unsupervised learning tasks that heavily rely on co-occurrence statistics such as graph embedding or recommendation.

REFERENCES

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association of Computational Linguistics*, 6:483–495, 2018.

Shima Asaadi, Saif M. Mohammad, and Svetlana Kiritchenko. Big bird: A large, fine-grained, bigram relatedness dataset for examining semantic composition. In *NAACL*, 2019.

Ben Athiwaratkun and Andrew Wilson. Multimodal word distributions. In *ACL*, 2017.

Lukas Balles and Thomas Fischbacher. Holographic and other point set distances for machine learning, 2019. URL `https://openreview.net/forum?id=rJlpUiAcYX`.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Tanner A Bohn and Charles X Ling. Neural sentence location prediction for summarization. *arXiv preprint arXiv:1804.08053*, 2018.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. A novel neural topic model and its supervised extension. In *AAAI*, 2015.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. Deep communicating agents for abstractive summarization. In *NAACL*, 2018.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval-2017*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Haim Dubossarsky, Eitan Grossman, and Daphna Weinshall. Coming to your senses: on controls and evaluation sets in polysemy research. In *EMNLP*, 2018.

Thomas Eiter and Heikki Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133, 1997.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. Sparse overcomplete word vector representations. In *ACL*, 2015.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *arXiv preprint arXiv:1902.01370*, 2019.

Rujun Han, Michael Gill, Arthur Spirling, and Kyunghyun Cho. Conditional word embedding and hypothesis testing via bayes-by-backprop. In *EMNLP*, 2018.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 2015.

Patrik O Hoyer. Non-negative sparse coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002.

Lifu Huang, Heng Ji, et al. Learning phrase embeddings from paraphrases with grus. In *Proceedings of the First Workshop on Curation and Applications of Parallel and Comparable Corpora*, 2017.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, 2015.

Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. Summarization based on embedding distributions. In *EMNLP*, 2015.

Ioannis Korkontzelos, Torsten Zesch, Fabio Massimo Zanzotto, and Chris Biemann. Semeval-2013 task 5: Evaluating phrasal semantics. In *SemEval 2013*, 2013.

Sachin Kumar and Yulia Tsvetkov. Von Mises-Fisher loss for training sequence to sequence models with continuous outputs. In *ICLR*, 2019.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *ICML*, 2015.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. Word sense induction for novel sense detection. In *EACL*, 2012.

Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL*, 2003.

Xinhai Liu, Zhizhong Han, Xin Wen, Yu-Shen Liu, and Matthias Zwicker. L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. *arXiv preprint arXiv:1908.00720*, 2019.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. Evaluating neural word representations in tensor-based compositional settings. In *EMNLP*, 2014.

David M. Mimno and Andrew McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *UAI*, 2008.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, 2014.

Denis Newman-Griffis, Albert M Lai, and Eric Fosler-Lussier. Jointly embedding entities and text with distant supervision. In *Proceedings of the 3rd Workshop on Representation Learning for NLP (Repl4NLP)*, 2018.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*, 2015.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

Kechen Qin, Cheng Li, Virgil Pavlu, and Javed A Aslam. Adapting rnn sequence prediction model to multi-label set prediction. In *NAACL*, 2019.

S Hamid Rezatofighi, Roman Kaskman, Farbod T Motlagh, Qinfeng Shi, Daniel Cremers, Laura Leal-Taixé, and Ian Reid. Deep perm-set net: learn to predict sets with unknown permutation and cardinality using deep neural networks. *arXiv preprint arXiv:1805.00613*, 2018.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *ACL*, 2017.

Raphael Shu and Hideki Nakayama. Compressing word embeddings via deep compositional code learning. In *ICLR*, 2018.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. Improving hypernymy detection with an integrated path-based and distributional method. In *ACL*, 2016.

Sidak Pal Singh, Andreas Hug, Aymeric Dieuleveut, and Martin Jaggi. Context mover's distance & barycenters: Optimal transport of contexts for building representations. *arXiv:1808.09663*, 2018.

Akash Srivastava and Charles A. Sutton. Autoencoding variational inference for topic models. In *ICLR*, 2017.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, 2018.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*, 2019.

Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Peter D Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *ICLR*, 2015.

Sean Welleck, Zixin Yao, Yu Gai, Jialin Mao, Zheng Zhang, and Kyunghyun Cho. Loss functions for multiset prediction. In *Advances in Neural Information Processing Systems*, 2018.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. Non-monotonic sequential text generation. *arXiv preprint arXiv:1902.02192*, 2019.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018.

Mo Yu and Mark Dredze. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242, 2015.

Hao Zheng and Mirella Lapata. Sentence centrality revisited for unsupervised summarization. In *ACL*, 2019.

A Experimental Details

Given the computational resource constraints, we keep our model simple enough to have a nearly converged training loss after 1 or 2 epoch(s). Since training takes a long time, we do not fine-tune the hyper-parameters in our models. We use a much smaller model compared with BERT but the architecture details in our transformer and most of its hyper-parameters are the same as the ones used in BERT.

The sparsity penalty weights on coefficient matrix λ is set to be 0.4. The maximal sentence size is set to be 50 and we ignore the sentences longer than that. The maximal number of co-occurring words is set to be 30 (after removing the stop words), and we sub-sample the words if there are more words in the previous and next sentence.

The number of dimensions in transformers is set to be 300. For sentence representation, the number of transformer layers on the decoder side is 5 and dropout on attention is 0.1 for K = 10 and the number of transformer layer on the decoder side is set to be 1 for K = 1 because we do not need to model the dependency of output basis. For phrase representation, the number of transformer layers on the decoder side is 2 and the dropout on attention is 0.5. The window size $d_t^1 = d_t^2$ is set to be 5. We will release the code to reveal more hyper-parameter setting details.

All the architecture and hyperparameters (except the number of codebook embeddings) in our models are determined by the validation loss of the self-supervised co-occurring word reconstruction task. The number of codebook embeddings K is chosen by the performance of training data in each task, but we observe that the performances are usually not sensitive to the numbers as long as K is large enough. We also suspect that the slight performance drops of models with too large K might just be caused by the fact that larger K needs longer training time and 1 week of training is insufficient to make the model converges.

For skip-thoughts, the hidden embedding of is set to be 600. To make the comparison fair, we retrain the skip-thoughts in Wikipedia 2016 for 2 weeks.

B COMPARISON WITH BERT LARGE

As mentioned in Section 3, our model has fewer parameters than the BERT base model and uses much less computational resources for training, so we only present the BERT base performance in the experiment sections. Nevertheless, we still wonder how well BERT large can perform in these unsupervised semantic tasks, so we compare our method with BERT Large in Table 6, Table 7, and Table 8. As we can see, BERT large is usually better than BERT base in the similarity tasks, but perform worse in the hypernym detection task. The performance gains of BERT in similarity tasks might imply that training a larger version of our model might be a promising future direction.

Although increasing the model size boosts the performance of BERT, our method is still much better in most of the cases, especially in the phrase similarity tasks. One of the main reasons we hypothesize is that BERT is trained by predicting the masked words in the input sequence and the objective function might not be good if sequences are short (like phrases).

C SUMMARIZATION COMPARISON GIVEN THE SAME SUMMARY LENGTH

In Section 3.4, we compare our methods with other baselines when all the methods choose the same number of sentences. We suspect that the bad performances for **W Emb (*)** methods (i.e., representing each sentence using the embedding of words in the sentence) might come from the tendency of selecting shorter sentences.[6] To verify the hypothesis, we plot the R-1 performance of different unsupervised summarization methods that do not use sentence order information versus the sentence length in Figure 3.

In the figure, we first observe that **Ours (K=100)** significantly outperforms **W Emb (GloVe)** and **Sent Emb (GloVe)** when summaries have similar length. In addition, we find that **W Emb (*)** actually usually outperform **Sent Emb (*)** when comparing the summaries with a similar length. Notice that this comparison might not be fair because **W Emb (*)** are allowed to select more sentences given the same length of summary and it might be easier to cover more topics in the document using more sentences. In practice, preventing choosing many short sentences might be preferable in an extractive summarization if fluency is an important factor.

Nevertheless, if our goal is simply to maximize the ROUGE F1 score given a fixed length of summary without accessing the ground truth summary and sentence order information, the figure indicates that **Ours (K=100)** is the best choice when the summary length is less than around 50 words and **W Emb (BERT)** becomes the best method for a longer summary. The BERT in this figure is the BERT base model. The mixed results suggest that combining our method with BERT in a way might be a promising direction to get the best performance in this task (e.g., use contextualized word embedding from BERT as our pre-trained word embedding space).

[6]In our implementation, we define the similarity between each word and an empty set of embeddings from sentences as -1000, so **W Emb (*)** nearly always choose the shortest sentence when selecting the first sentence.

Table 6: Compare BERT Large with Ours in Table 2.

Method		SemEval		Turney (5)	Turney (10)	BiRD	WikiSRS-Sim	WikiSRS-Rel
Model	Score	AUC	F1	Acc	Acc	Pearson	Spearman	
BERT Base	Avg	66.5	67.1	43.4	24.3	44.4	43.3	40.7
BERT Large	Avg	72.4	66.7	**51.3**	32.1	47.5	49.6	48.1
Ours (K=1)	Emb	**87.8**	**78.6**	50.3	**32.5**	**60.6**	**67.1**	**65.8**

Table 7: Compare BERT Large with Ours in Table 3.

Method		Dev		Test	
Model	Score	All	Low	All	Low
BERT Base	Avg	62.3	42.1	51.2	39.1
BERT Large	Avg	68.4	52.6	59.5	49.4
Ours K=10	Prob_avg	**72.0**	**60.5**	**61.4**	**59.3**

Table 8: Compare BERT Large with Ours in Table 4.

Method	Dev		Test	
	AUC	Acc	AUC	Acc
BERT Base (Avg)	25.6	0.5	25.6	0.5
BERT Large (Avg)	20.2	0.5	20.1	0.5
Ours (K=1)	**29.3**	**82.7**	**29.6**	**81.0**

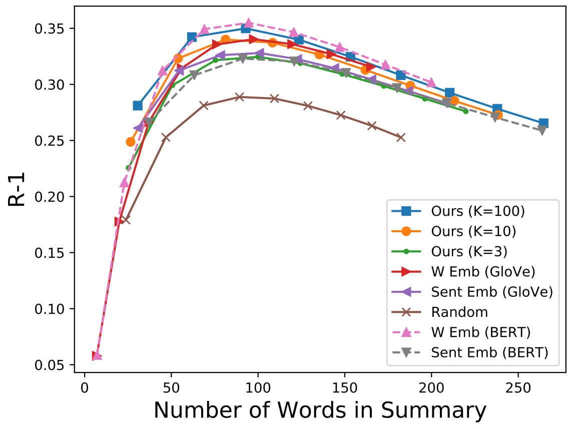

Figure 3: Comparing the F1 of ROUGE-1 score on unsupervised methods that do not access the sentence order information

D More Examples

We visualize predicted embeddings from 10 randomly selected sentences in our validation set. The format of the file is similar to Table 1. The first line of an example is always the prepossessed input sentence, where <unk> means an out-of-vocabulary placeholder. The embedding in each row is visualized by the nearest five neighbors in a GloVe embedding space and their cosine similarities to the vector.

Stanley (1870) , <unk> of American Indians : Proof <unk> , Ithaca , NY : Cornell University <unk> Library Rare Manuscripts , Archives . <eos>
J. 0.871 M. 0.854 R. 0.853 D. 0.849 A. 0.839
York 0.929 New 0.911 NY 0.858 Brooklyn 0.731 NYC 0.716
1874 0.970 1872 0.967 1877 0.964 1876 0.963 1885 0.962
History 0.724 Historical 0.681 Manuscripts 0.665 Histories 0.635 Literature 0.628
American 0.939 America 0.714 U.S. 0.619 States 0.609 Americans 0.593
Indians 0.878 Indian 0.666 Native 0.665 Tribe 0.658 indians 0.615
University 0.933 Faculty 0.734 Sciences 0.727 College 0.725 Stanford 0.684
ISBN 0.825 paperback 0.757 Paperback 0.712 Hardcover 0.712 hardcover 0.693
Stanley 0.925 Merrill 0.576 Rowe 0.554 Raymond 0.535 Phillips 0.528
0 0.821 1 0.702 4 0.663 3 0.658 6 0.649

Edison Oil Field : California Division of Oil and Gas , Summary of Operations . 2 , p

5 - 9 The Edison post office opened in 1903 , closed in 1929 , and re - opened in 1946 . <eos>
California 0.939 San 0.805 Diego 0.791 Sacramento 0.769 Angeles 0.764
1925 0.970 1923 0.969 1921 0.964 1924 0.963 1927 0.963
Edison 0.934 Westinghouse 0.553 Emerson 0.497 Tesla 0.480 Franklin 0.461
Field 0.854 Operations 0.636 Division 0.636 National 0.544 Fields 0.542
located 0.788 downtown 0.732 adjacent 0.709 situated 0.709 east 0.698
opened 0.784 reopened 0.626 relocated 0.614 closed 0.606 operated 0.596
Gas 0.860 gas 0.808 Oil 0.707 fuel 0.689 Fuel 0.688
office 0.839 Office 0.711 offices 0.673 department 0.619 desk 0.566
Oil 0.922 oil 0.835 Oils 0.663 oils 0.651 OIL 0.601
4 0.906 6 0.893 3 0.890 5 0.879 2 0.870

Chris van Wyk (19 July 1957 3 October 2014) was a South African children‘s book author , novelist and poet . Van Wyk is famous for his poem In Detention on the suspicious deaths that befell South African political prisoners during Apartheid . <eos>
Africa 0.893 African 0.876 Kenya 0.722 Zimbabwe 0.701 Ghana 0.699
Wyk 0.940 Louw 0.695 Smit 0.683 Merwe 0.673 Visser 0.639
poem 0.900 poems 0.865 poetry 0.853 poet 0.795 poetic 0.749
book 0.801 author 0.792 books 0.749 novels 0.738 nonfiction 0.715
prisoners 0.829 detention 0.775 prison 0.771 detainees 0.754 prisoner 0.734
Van 0.940 van 0.811 Holland 0.558 VAN 0.553 Gogh 0.513
van 0.933 Van 0.744 een 0.657 op 0.652 het 0.650
born 0.892 married 0.738 daughter 0.708 died 0.700 Born 0.661
1972 0.785 1973 0.781 1971 0.776 1974 0.776 1977 0.773
South 0.917 North 0.767 West 0.689 East 0.682 Africa 0.674

The temple is one of the 275 Paadal Petra Sthalams (Holy <unk> of Shiva) on the continent . <unk> Samhita , Chapter 11 on the <unk> of the North , in Shiva Purana mentions this Shivalinga as the <unk> of all wishes . <eos>
Shiva 0.937 Vishnu 0.821 Siva 0.754 Ganesha 0.723 Krishna 0.719
temple 0.971 temples 0.816 Temple 0.766 shrine 0.741 shrines 0.681
Shivalinga 0.691 Purana 0.685 Skanda 0.669 Bhagavata 0.629 Ranganatha 0.619
Pradesh 0.769 Andhra 0.765 Karnataka 0.757 India 0.750 Nadu 0.749
gods 0.790 divine 0.742 heaven 0.741 heavens 0.723 eternal 0.713
Petra 0.908 Daniela 0.539 Nadia 0.538 Simona 0.527 Katarina 0.521
Sanskrit 0.657 Samhita 0.657 texts 0.637 Vedic 0.619 Puranas 0.619
God 0.697 blessed 0.694 blessing 0.685 heaven 0.672 divine 0.664
southeast 0.732 north 0.720 northeast 0.718 south 0.714 east 0.709
Chapter 0.877 chapter 0.725 CHAPTER 0.674 Chapters 0.654 chapters 0.623

Just prior to World War I , the real estate boom ended suddenly , causing the city ’s population to decline <unk> over <unk> in 1914 to under 54,000 only two years later . <eos>
city 0.868 town 0.837 area 0.751 south 0.735 towns 0.733
1912 0.963 1902 0.961 1921 0.958 1905 0.954 1922 0.954
later 0.754 eventually 0.745 once 0.742 became 0.724 moved 0.716
population 0.857 populations 0.758 declining 0.666 economic 0.660 increasing 0.654
estate 0.883 property 0.776 homes 0.696 Estate 0.685 estates 0.673
100,000 0.834 200,000 0.821 300,000 0.818 60,000 0.817 80,000 0.813
City 0.836 Downtown 0.748 city 0.660 downtown 0.659 Town 0.609
War 0.887 war 0.782 WWII 0.715 WWI 0.681 WW2 0.665
wharfs 0.631 disused 0.625 embankments 0.616 farmsteads 0.604 bulldozed 0.594
Karlsruhe 0.644 Rostock 0.643 Dusseldorf 0.639 Magdeburg 0.630 Erfurt 0.627

The following year he ordered great festivities on the Mount <unk> to celebrate the anniversary of the victory over the Mongols , and massacred there all <unk> nobles . <eos>
army 0.779 troops 0.771 battle 0.766 forces 0.764 war 0.743
Mongols 0.728 Tartars 0.721 Seljuks 0.688 Bulgars 0.676 Manchus 0.667
came 0.811 had 0.788 knew 0.785 took 0.782 finally 0.778
Emperor 0.811 emperor 0.707 Emperors 0.656 Dynasty 0.648 Empire 0.644

Mian 0.696 Ahmad 0.678 Ali 0.671 Jia 0.669 Ahmed 0.665
princes 0.751 nobles 0.722 nobility 0.704 rulers 0.700 monarch 0.687
Xinjiang 0.711 Mongolia 0.666 Mongols 0.651 Mongol 0.650 Tibet 0.643
celebration 0.752 birthday 0.740 anniversary 0.711 celebrations 0.684 celebrate 0.678
842 0.861 794 0.857 782 0.849 823 0.837 758 0.835
Mount 0.912 Mt 0.696 Mt. 0.654 mount 0.577 Mounts 0.542

Administratively Nepal is divided into many Development regions , Zones , Districts , Village development committees , Metropolitan areas and Municipalities . Nepal is divided into 5 Development regions Nepal is divided into 14 Zones . <eos>
municipalities 0.846 municipality 0.778 municipal 0.762 councils 0.716 Municipalities 0.711
Nepal 0.967 Bhutan 0.807 Kathmandu 0.742 India 0.691 Lanka 0.677
region 0.853 regions 0.834 northern 0.811 southern 0.807 southeastern 0.776
Administratively 0.686 Sunsari 0.588 Surkhet 0.581 Badulla 0.578 Baglung 0.571
development 0.857 developing 0.708 projects 0.697 project 0.695 initiatives 0.655
separate 0.706 divided 0.703 comprised 0.691 comprise 0.687 consists 0.663
Development 0.820 Planning 0.693 Management 0.677 Initiatives 0.675 Assessment 0.671
Metropolitan 0.797 Metro 0.601 City 0.581 metropolitan 0.567 Downtown 0.566
Zones 0.882 zones 0.796 Zone 0.684 zone 0.579 Areas 0.437
6 0.909 4 0.906 5 0.893 3 0.886 8 0.866

The main groups that lived or interacted in the Cambridge Bay area were the <unk> (<unk> or <unk>) , <unk> (<unk>) , the <unk> and the <unk> . The first Europeans to see it was Thomas Simpson in 1839 and John Rae in 1851 . <eos>
area 0.833 north 0.760 south 0.756 east 0.754 northeast 0.752
1835 0.961 1839 0.961 1842 0.961 1838 0.960 1834 0.959
Cambridge 0.877 Oxford 0.799 London 0.740 Nottingham 0.691 Exeter 0.665
Rae 0.912 Carly 0.682 Mandy 0.661 Brooke 0.658 Melanie 0.651
Europeans 0.746 indigenous 0.695 Africans 0.658 peoples 0.641 natives 0.625
became 0.757 discovered 0.693 was 0.688 lived 0.686 existed 0.685
Bay 0.929 Harbor 0.733 Harbour 0.720 Beach 0.691 Cove 0.674
groups 0.907 group 0.879 members 0.639 Groups 0.625 organizations 0.550
groups 0.874 group 0.857 Groups 0.621 members 0.569 Group 0.496
Thomas 0.903 William 0.838 John 0.787 James 0.773 Henry 0.771

So , at the Queen of Empires stop at <unk> , Desh 's Peace Brigade forces attack the pleasure yacht and attempt to kidnap Elan and <unk> . <eos>
attack 0.846 attacks 0.783 enemy 0.781 attacked 0.749 fighting 0.723
let 0.857 want 0.828 trying 0.817 go 0.813 get 0.810
Desh 0.910 Mera 0.564 Apna 0.543 Aana 0.542 Prabhat 0.542
Brigade 0.900 Battalion 0.881 Infantry 0.863 Regiment 0.858 battalion 0.856
unspeakable 0.635 treachery 0.619 deceit 0.618 diabolical 0.603 malevolent 0.601
Kiran 0.712 Rajesh 0.702 Prakash 0.700 Chaudhary 0.698 Sharma 0.694
yacht 0.859 sailing 0.809 boat 0.803 boats 0.796 sail 0.790
Empires 0.904 Empire 0.719 Kingdoms 0.705 Civilization 0.689 empires 0.671
discovers 0.713 finds 0.678 realizes 0.677 sees 0.674 learns 0.661
Queen 0.912 King 0.797 Prince 0.692 Princess 0.686 queen 0.676

Retrieved 6 December 2014 In general , <unk> use of flowers in her manuscripts were primarily used to produce attractive manuscripts , which would create employment or reward opportunities . However , this may not have been the only reason Inglis decided to draw flowers . <eos>
manuscript 0.816 manuscripts 0.814 books 0.736 publications 0.721 journals 0.712
flowers 0.925 flower 0.903 roses 0.822 blossoms 0.804 blooms 0.771
however 0.696 mentioned 0.689 perhaps 0.673 though 0.670 simply 0.665
Margaret 0.804 Elizabeth 0.784 Frances 0.781 William 0.765 Jane 0.765
Inglis 0.933 Innes 0.575 Cowan 0.564 Mackinnon 0.553 Mackay 0.552
manuscripts 0.718 Manuscripts 0.709 Manuscript 0.702 Texts 0.623 Bibliography 0.614
February 0.936 April 0.929 January 0.927 March 0.926 October 0.925

Wales 0.733 Perth 0.699 Australian 0.671 Melbourne 0.669 Australia 0.669
September 0.910 October 0.909 January 0.907 February 0.906 August 0.904
2014 0.937 2015 0.866 2016 0.769 2017 0.727 2018 0.691