Extending Multi-Sense Word Embedding to Phrases and Sentences for Unsupervised Semantic Applications

Haw-Shiuan Chang, Amol Agrawal, Andrew McCallum
CICS, University of Massachusetts Amherst
{hschang,amolagrawal,mccallum}@cs.umass.edu

Abstract

Most unsupervised NLP models represent each word with a single point or single region in semantic space, while the existing multi-sense word embeddings cannot represent longer word sequences like phrases or sentences. We propose a novel embedding method for a text sequence (a phrase or a sentence) where each sequence is represented by a distinct set of multi-mode codebook embeddings to capture different semantic facets of its 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 introduce an end-to-end trainable neural model that directly predicts the set of cluster centers from the input text sequence during test time. Our experiments show that the per-sentence codebook embeddings significantly improve the performances in unsupervised sentence similarity and extractive summarization benchmarks. In phrase similarity experiments, we discover that the multi-facet embeddings provide an interpretable semantic representation but do not outperform the single-facet baseline.

1 Introduction

Collecting manually labeled data is an expensive and tedious process for new or low-resource NLP applications. Many of these applications require the text similarity measurement based on the text representation learned from the raw text without any supervision. Examples of the representation include word embedding like Word2Vec [Mikolov et al. 2013] or GloVe (Pennington, Socher, and Manning 2014), sentence embeddings like skip-thoughts (Kiros et al. 2015), contextualized word embedding like ELMo (Peters et al. 2018) and BERT (Devlin et al. 2019) without fine-tuning.

The existing work often represents a word sequence (e.g., a sentence or a phrase) as a single embedding. However, when squeezing all the information into a single embedding (e.g., by averaging the word embeddings or using CLS embedding in BERT), the representation might lose some important information of different facets in the sequence.

Inspired by the multi-sense word embeddings (Lau et al. 2012; Neelakantan et al. 2014; Athiwaratkun and Wilson 2017; Singh et al. 2020), we propose a multi-facet representation that characterizes a phrase or a sentence as a fixed number of embeddings, where each embedding is a clustering center of the words co-occurring with the input word sequence.

In this work, a facet refers to a mode of the co-occurring word distribution, which might be multimodal. For example, the multi-facet representation of real property is illustrated in Figure 1. Real property can be observed in legal documents where it usually means real estate, while real property can also mean a true characteristic in philosophic discussions. The previous unsupervised multi-sense embeddings discover those senses by clustering the observed neighboring words (e.g., acquired, save, and tax) and an important facet, a mode with high probability, could be represented by several close cluster centers. Notice that the approaches need to solve a distinct local clustering problem for each phrase in contrast with the topic modeling like LDA (Blei, Ng, and Jordan 2003), which clusters all the words in the corpus into a global set of topics.

Figure 1: The input phrase real property is represented by \( K = 5 \) cluster centers. The previous work discovers the multiple senses by clustering 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 input phrase so as to reconstruct the (unseen) co-occurring distribution well.
In addition to a phrase, we can also cluster the nearby words of a sentence which appears frequently in the corpus. The cluster centers usually correspond to important aspects rather than senses (see an example in Figure 2) because a sentence usually has multiple aspects but only one sense. However, extending the clustering-based multi-sense word embeddings to long sequences such as sentences is difficult in practice 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 the number of unique sequences, $|K|$ is the number of clusters, and $|E|$ is the embedding dimensions. Estimating and storing such a large number of parameters takes time and space. More importantly, much more unique sequences imply much fewer co-occurring words to be clustered for each sequence, especially for sentences. An effective model needs to overcome this sample efficiency challenge (i.e., sparseness in the co-occurring statistics), but 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 often lead to similar cluster centers, so we should be able to solve these local clustering problems using much fewer parameters to circumvent the challenges. To achieve the goal, we develop a novel Transformer-based neural encoder and decoder. As shown in Figure 1, instead of clustering co-occurring words beside an input sequence at test time as in previous approaches, we learn a mapping between the input 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 sequence during testing.

To train the neural model that predicts the clustering centers, we match the sequence of predicted cluster centers and the observed set of co-occurring word embeddings using a non-negative and sparse permutation matrix. After the permutation matrix is estimated for each input sequence, the gradients are back-propagated to cluster centers (i.e., codebook embeddings) and to the weights of our neural model, which allows us to train the whole model end-to-end.

In the experiments, we evaluate whether the proposed multi-facet embeddings could improve the similarity measurement between two sentences, between a sentence and a document (i.e., extractive summarization), and between phrases. The results demonstrate multi-facet embeddings significantly outperform the classic single embedding baseline when the input sequence is a sentence.

We also demonstrate several advantages of the proposed multi-facet embeddings over the (contextualized) embeddings of all the words in a sentence. First, we discover that our model tends to use more embeddings to represent an important facet or important words. This tendency provides an unsupervised estimation of word importance, which improves various similarity measurements between a sentence pair. Second, our model outputs a fixed number of facets by compressing long sentences and extending short sentences. In unsupervised extractive summarization, this capability prevents the scoring function from biasing toward longer or shorter sentences. Finally, in the phrase similarity experiments, our methods capture the compositional meaning (e.g., a hot dog is a food) of a word sequence well and the quality of our similarity estimation is not sensitive to the choice of $K$, the number of our codebook embeddings.

### 1.1 Main Contributions

1. As shown in Figure 1, we propose a novel framework that predicts the cluster centers of co-occurring word embeddings to overcome the sparsity challenges in our self-supervised training signals. This allows us to extend the idea of clustering-based multi-sense embeddings to phrases or sentences.

2. We propose a deep architecture that can effectively encode a sequence and decode a set of embeddings. We also propose non-negative sparse coding (NNSC) loss to train our neural encoder and decoder end-to-end.

3. We demonstrate how the multi-facet embeddings could be used in unsupervised ways to improve the similarity between sentences/phrases, infer word importance in a sentence, extract important sentences in a document. In Appendix B.1, we show that our model could provide asymmetric similarity measurement for hypernym detection.

4. We conduct comprehensive experiments in the main paper and appendix to show that multi-facet embedding is consistently better than classic single-facet embedding for modeling the co-occurring word distribution of sentences, while multi-facet phrase embeddings do not yield a clear advantage against the single embedding baseline, which supports the finding in Dubossarsky, Grossman, and Weinshall (2018).

### 2 Method

In this section, we first formalize our training setup and next describe our objective function and neural architecture. Our approach is visually summarized in Figure 2.

#### 2.1 Self-supervision Signal

We express $t$th sequence of words in the corpus as $I_t = w_{x_1}...w_{y_t}<\text{eos}>$, where $x_1$ and $y_t$ are the start and end position of the input sequence, respectively, and $<\text{eos}>$ is the end of sequence symbol.

We assume neighboring words beside each input phrase or sentence are related to some facets of the sequence, so given $I_t$ as input, our training signal is to reconstruct a set of co-occurring words, $N_t = \{w_{x_1}, w_{x_2}, ..., w_{x_t}, w_{y_t+1}, ..., w_{y_t+d_2} \}$. In our experiments, we train our multi-facet sentence embeddings by setting $N_t$ as the set of all words in the previous and the next sentence, and train multi-facet phrase embeddings by setting a fixed window size $d_1 = d_2 = 5$.

Since there are not many co-occurring words for a long sequence (none are observed for unseen testing sequences), the goal of our model is to predict the cluster centers of the

$^4$The self-supervised signal is a generalization of the loss for prediction-based word embedding like Word2Vec (Mikolov et al. 2013). They are the same when the input sequence length $|I_t|$ is 1.
words that could "possibly" occur beside the text sequence rather than the cluster centers of the actual occurring words in \( N_t \) (e.g., the hidden co-occurring distribution instead of green and underlined words in Figure 2). The cluster centers of an unseen testing sequence are predictable because the model could learn from similar sequences and their co-occurring words in the training corpus.

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 a pre-trained word embedding space, we predict the cluster centers of the co-occurring word embeddings. The embeddings of co-occurring words \( N_t \) are arranged into a matrix \( W(N_t) = [w_{ij}]_{j=1...|N_t|} \) with size \(|E| \times |N_t|\), where \(|E|\) is the dimension of pre-trained word embedding, and each of its columns \( w_j \) is a normalized word embedding whose 2-norm is 1. The normalization makes the cosine distance between two words become half of their squared Euclidean distance.

Similarly, we denote the predicted cluster centers \( \mathbf{c}_k \) of the input sequence \( I_t \) as a \(|E| \times K\) matrix \( F(I_t) = [c_{k,j}]_{j=1...K} \), where \( F \) is our neural network model and \( K \) is the number of clusters.

To simplify the design of our prediction model and the unsupervised scoring functions used in the downstream tasks. When the number of modes in the (multimodal) co-occurring distribution is smaller than \( K \), the model can output multiple cluster centers to represent a mode (e.g., the music facet in Figure 2 is represented by two close cluster centers). As a result, the performances in our downstream applications are not sensitive to the setting of \( K \) when \( K \) is larger than the number of facets in most input word sequences.

The reconstruction loss of k-means clustering in the word embedding space can be written as

\[
E_{r}(F(I_t), W(N_t)) = \|F(I_t)M - W(N_t)\|^2 = \sum_j ||(\sum_k M_{k,j} c_k) - w_j||^2,
\]

where \( M_{k,j} = 1 \) if the \( j \)th word belongs to the \( k \) cluster and 0 otherwise. That is, \( M \) is a permutation matrix which matches the cluster centers and co-occurring words and allow the cluster centers to be predicted in an arbitrary order.

Non-negative sparse coding (NNSC) (Hoyer 2002) relaxes the constraints by allowing the coefficient \( M_{k,j} \) to be a positive value but encouraging it to be 0. We adopt NNSC in our work because we observe that the neural network trained by NNSC loss generates more diverse topics than k-means loss does. We hypothesize that it is because the loss is smoother and easier to be optimized for a neural network. Using NNSC, we define our reconstruction error as

\[
E_{r}(F(I_t), W(N_t)) = \|F(I_t)M^{\ast} - W(N_t)\|^2
\]

s.t.,

\[
M^{\ast} = \text{arg min}_{M} \|F(I_t)M - W(N_t)\|^2 + \lambda \|M\|_1,
\]

\[
\forall k,j, \quad 0 \leq M_{k,j} \leq 1,
\]

Figure 2: Our model for sentence representation. We represent each sentence as multiple codebook embeddings (i.e., cluster 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., centers) predicted by our sequence to embeddings model. Our loss encourages the model to generate codebook embeddings whose 2-norm is 1. The normalization makes the cosine distance between two words become half of their squared Euclidean distance.
where $\lambda$ is a hyper-parameter controlling the sparsity of $M$. We force the coefficient value $M_{k,j} \leq 1$ to avoid the neural network learning to predict centers with small magnitudes which makes the optimal values of $M_{k,j}$ large and unstable.

We adopt an alternating optimization strategy similar to the EM algorithm for k-means. At each iteration, our E-step estimates the permutation coefficient $M_{O_t}$ after fixing our neural model, while our M-step treats $M_{O_t}$ as constants to back-propagate the gradients of NNSC loss to our neural network. A pseudo-code of our training procedure could be found in Algorithm 1 in the appendix. Estimating the permutation between the prediction and ground truth words is often computationally expensive (Qin et al. 2019). Nevertheless, optimizing the proposed loss is efficient because for each training sequence $I_t$, $M_{O_t}$ can be efficiently estimated using convex optimization (our implementation uses RMSProp (Tieleman and Hinton 2012)). Besides, we minimize the L2 distance, $||F(I_t)M_{O_t} − W(N_t)||^2$, in a pre-trained embedding space as in Kumar and Tsvetkov (2019); Li et al. (2019) rather than computing softmax.

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

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

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 $F = \arg \min_F \sum I_t(F)$. 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 Transformer-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

$$[e_{x_1} ... e_{x_t} e_{<eos>}] = TE(w_{x_1} ... w_{x_t}, <eos>), \tag{3}$$

where the goal of the encoder is to map the similar sentences, which are likely to have similar co-occurring word distribution, to similar contextualized embeddings.

Different from the typical seq2seq model (Sutskever, Vinyals, and Le 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 as in Lee et al. (2019) without requiring an expensive softmax layer or auto-regressive decoding.\footnote{The decoder can also be viewed as another Transformer encoder which attends the output of the first encoder and models the dependency between predicted cluster centers.}

To make different codebook embeddings capture different facets, we pass the embeddings of $<eos>$, $e_{<eos>}$, to different linear layers $L_k$ before becoming the input of the decoder $TD$. The decoder allows the input embeddings to attend each other to model the dependency among the facets and attend the contextualized word embeddings from the encoder, $e_{x_1} ... e_{x_t} e_{<eos>}$, to copy the embeddings of some keywords in the word sequence as our facet embeddings more easily. Specifically, the codebook embeddings

$$F(I_t) = TD(L_1(e_{<eos>}) ... L_k(e_{<eos>}), e_{x_1} ... e_{x_t} e_{<eos>}). \tag{4}$$

We find that removing the attention on the $e_{x_1} ... e_{x_t} e_{<eos>}$ significantly deteriorates 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 encoder-decoder attention does not significantly change the performance of phrase representation, so we remove the connection (i.e., encoder and decoder have the same architecture) in models for phrase representation. Notice that the framework is flexible. For example, we can encode the genre of the document containing the sentence if desired.

### 3 Experiments

Quantitatively evaluating the quality of our predicted cluster centers is difficult because the existing label data and metrics are built for global clustering. The previous multi-sense word embedding studies often show that multiple embeddings could improve the single word embedding in the unsupervised word similarity task to demonstrate its effectiveness. Thus, our goal of experiments is to discover when and how the multi-facet embeddings can improve the similarity measurement in various unsupervised semantic tasks upon the widely-used general-purpose representations, such as single embedding and (contextualized) word embeddings.

#### 3.1 Experiment Setup

Our models only require the raw corpus and sentence/phrase boundaries, so we will only compare them with other unsupervised alternatives that do not require any manual labels or multi-lingual resources such as PPDB (Pavlick et al. 2019). To simplify the comparison, we also omit the comparison with the methods using character-level information such as fastText (Bojanowski et al. 2017) or bigram information such as Sent2Vec (Pagliardini, Gupta, and Jaggi 2018a).

It is hard to make a fair comparison with BERT (Devlin et al. 2019). Its masked language modeling loss is designed for downstream supervised tasks and preserves more syntax information which might be distractive in unsupervised semantic applications. Furthermore, BERT uses word piece tokenization while other models use word tokenization. Nevertheless, we still present the performances of the BERT Base model as a reference even though it is trained using more parameters, larger embedding size, larger corpus, and more computational resources compared with our models. Since we focus on unsupervised setting, we directly use the final hidden states of the BERT models without supervised fine-tuning in most of the comparisons. One exception is that we also report the performance of sentence-BERT (Reimers and Gurevych 2019) in a low-resource setting.

Our model is trained on English 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. We use
Table 1: Examples of the codebook embeddings predicted by our models with different K. The embedding in each row is visualized by the three words whose GloVe embeddings have the highest cosine similarities (also presented) with the codebook embedding.

| K = 10: | can 0.854, should 0.834, either 0.821 | hospital 0.886, medical 0.771, hospitals 0.745 | services 0.768, service 0.749, web 0.722 | SMS 0.891, sms 0.745, messaging 0.686 | systems 0.728, technologies 0.725, integrated 0.723 | appointments 0.791, appointment 0.735, duties 0.613 | confirmation 0.590, request 0.568, receipt 0.563 | countries 0.855, nations 0.737, Europe 0.732 | implementation 0.613, application 0.610, programs 0.603 |
| K = 3: | initiatives 0.736, organizations 0.725, efforts 0.725 | hospital 0.857, medical 0.780, hospitals 0.739 | SMS 0.791, mobile 0.635, messaging 0.631 | hospital 0.886, medical 0.771, hospitals 0.745 | service 0.768, service 0.749, web 0.722 | SMS 0.891, sms 0.745, messaging 0.686 | systems 0.728, technologies 0.725, integrated 0.723 | appointments 0.791, appointment 0.735, duties 0.613 |
| K = 1: | can 0.769, possible 0.767, specific 0.767 | hospital 0.857, medical 0.780, hospitals 0.739 | SMS 0.791, mobile 0.635, messaging 0.631 | information 0.702, use 0.701, specific 0.700 | can 0.769, possible 0.767, specific 0.767 | hospital 0.857, medical 0.780, hospitals 0.739 | SMS 0.791, mobile 0.635, messaging 0.631 | information 0.702, use 0.701, specific 0.700 |

3.2 Qualitative Evaluation

The cluster centers predicted by our model are visualized in Table 1 (as using girl and lady to visualize the red cluster center in Figure 2). Some randomly chosen examples are also visualized in Appendix D.

The centers summarize the input sequence well and more codebook embeddings capture more fine-grained semantic facets of a phrase or a sentence. Furthermore, the embeddings capture the compositional meaning of words. For example, each word in the phrase civil order does not mean initiatives, army, or court, which are facets of the whole phrase. When the input is a sentence, we can see that the output embeddings are sometimes close to the embeddings of words in the input sentence, which explains why attending the contextualized word embeddings in our decoder could improve the quality of the output embeddings.

3.3 Unsupervised Sentence Similarity

We propose two ways to evaluate the multi-facet embeddings using sentence similarity tasks.

First way: Since similar sentences should have similar word distribution in nearby sentences and thus similar codebook embeddings, the codebook embeddings of a query sentence $\hat{F}_u(S_q^1)$ should be able to well reconstruct the codebook embeddings of its similar sentence $\hat{F}_u(S_q^2)$. We compute the reconstruction error of both directions and add them as a symmetric distance $SC$:

$$SC(S_q^1, S_q^2) = Er(\hat{F}_u(S_q^1), \hat{F}_u(S_q^2)) + Er(\hat{F}_u(S_q^2), \hat{F}_u(S_q^1)),$$

where $\hat{F}_u(S_q) = \frac{1}{k}W(\hat{a}_q)\hat{a}_q$, $k$ is a matrix of normalized codebook embeddings and $Er$ function is defined in equation [1]. We use the negative distance to represent similarity.

Second way: One of the main challenges in unsupervised sentence similarity tasks is that we do not know which words are more important in each sentence. Intuitively, if one word in a query sentence is more important, the chance of observing related/similar words in the nearby sentences should be higher. Thus, we should pay more attention to the words in a sentence that have higher cosine similarity with its multi-facet embeddings, a summary of the co-occurring word distribution. Specifically, our importance/attention weighting for all the words in the query sentence $S_q$ is defined by

$$a_q = \max(0, W(S_q)^T\hat{F}_u(S_q)) \mathbf{1}.$$

where $\mathbf{1}$ is an all-one vector. We show that the attention vector (denoted as Our a in Table 2) could be combined with various scoring functions and boost their performances. As a baseline, we also report the performance of the attention weights derived from the k-means loss rather than NNSC loss and call it Our a (k-means).

Setup: STS benchmark (Cer et al. 2017) is a widely used sentence similarity task. We compare the correlations between the predicted semantic similarity and the manually labeled similarity. We report Pearson correlation coefficient, which is strongly correlated with Spearman correlation in all our experiments. Intuitively, when two sentences are less similar to each other, humans tend to judge the similarity based on how similar their facets are. Thus, we also compare the performances on the lower half of the datasets where their ground truth similarities are less than the median similarity in the dataset, and we call this benchmark STSB Low.

A simple but effective way to measure sentence similarity is to compute the cosine similarity between the average (contextualized) word embedding $\bar{e}(m)$ (denoted as CLS and Skip-thought Cosine, respectively).

nlp.stanford.edu/projects/glove/
We also test word mover’s distance (WMD) [Kusner et al. 2015], which explicitly matches every word in a pair of sentences. As we do in Prob_avg, we apply $\frac{\alpha}{\alpha + p(w)}$ to WMD to down-weight the importance of functional words, and call this scoring function as Prob_WMD. When using Our a, we multiple our attention vector with the weights of every word (e.g., $\frac{\alpha}{\alpha + p(w)}$ for Prob_avg and Prob_WMD).

To motivate the unsupervised setting, we present the performance of sentence-BERT [Reimers and Gurevych 2019] that are trained by 100 sentence pairs. We randomly sample the sentence pairs from a data source that is not included in STSB (e.g., headlines in STS 2014), and report the testing performance averaged across all the sources from STS 2012 to 2016. More details are included in Appendix B.2.

**Results:** In Figure 3, we first visualize our attention weights in equation 6 and our output codebook embeddings for a pair of similar sentences from STSB to intuitively explain why modeling co-occurring distribution could improve the similarity measurement.

Many similar sentences might use different word choices or using extra words to describe details, but their possible nearby words are often similar. For example, appending in the garage to A man is lifting weights does not significantly change the facets of the sentences and thus the word garage receives relatively a lower attention weight. This makes its similarity measurement from our methods, Our c and Our a, closer to the human judgment than other baselines.

In Table 2, Our c SC, which matches between two sets of facets, outperforms WMD, which matches between two sets of words in the sentence, and also outperforms BERT Avg, especially in STSB Low. The significantly worse performances from Skip-thought Cosine justify our choice of ignoring the other in the co-occurring words.

All the scores in Our * K10 are significantly better than Our * K1, which demonstrates the co-occurring word distribution is hard to be modeled well using a single embedding. Multiplying the proposed attention weighting consistently boosts the performance in all the scoring functions especially in STSB Low and without relying on the generalization assumption of the training distribution. Finally, using k-means loss, Our a (k-means) K10, significantly degrades the performance compared to Our a K10, which justify the proposed NNNSC loss. In Appendix B.2 our methods are compared with more baselines using more datasets to test the effectiveness of multi-facets embeddings and our design.
of the sentences. Since longer sentences have more words, we normalize the gain of the reconstruction similarity by the sentence length. The method is denoted as W Emb. We also test the baselines of selecting random sentences (Rnd) and first 3 sentences (Lead-3) in the document.

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

In CNN/Daily Mail, the unsupervised methods which access sentence order information such as Lead-3 have performances similar to supervised methods such as RL (Celikyilmaz et al. [2018]). 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.

**Results:** In Table 5, predicting 100 clusters yields the best results. Notice that our method greatly alleviates the computational and sample efficiency challenges, which allows us to set cluster numbers K to be a relatively large number.

The results highlight the limitation of classic representations. The single sentence embedding cannot capture its multiple facets. On the other hand, if a sentence is represented by the embeddings of its words, it is difficult to eliminate the bias of selecting longer or shorter sentences and a facet might be composed by multiple words (e.g., the input sentence in Table 1 describes a service, but there is not a single word in the sentence that means service).

### 3.4 Unsupervised Extractive Summarization

The classic representation of a sentence uses either a single embedding or the (contextualized) embeddings of all the words in the sentence. In this section, we would like to show that both options are not ideal for extracting a set of sentences as a document summary.

Table 1 indicates that our multiple codebook embeddings of a sentence capture its different facets well, so we represent a document summary S as the union of the multifacet embeddings of the sentences in the summary \( R(S) = \bigcup_{t=1}^T \{ \hat{F}_u(S_t) \} \), where \( \{ \hat{F}_u(S_t) \} \) is the set of column vectors in the matrix \( \hat{F}_u \) of sentence \( S_t \).

A good summary should cover multiple facets that represent all topics/concepts in the document (Kobayashi, Noguchi, and Yatsuka 2015). The objective can be quantified as discovering a summary \( S \) whose multiple embeddings \( R(S) \) best reconstruct the distribution of normalized word embedding \( w \) in the document \( D \) (Kobayashi, Noguchi, and Yatsuka 2015). That is,

\[
\arg \max_S \sum_{w \in D} \frac{\alpha}{\alpha + p(w)} \max_{u \in R(S)} w^T \hat{s}_u, \tag{7}
\]

where \( \frac{\alpha}{\alpha + p(w)} \) is the weights of words we used in the sentence similarity experiments (Arora, Liang, and Ma 2017). We greedily select sentences to optimize equation 7 as in Kobayashi, Noguchi, and Yatsuka [2015].

**Setup:** We compare our multi-facet embeddings with other alternative ways of modeling the facets of sentences. A simple way is to compute the average word embedding as a single-facet sentence embedding. This baseline is labeled as Sent Emb. Another way is to use the (contextualized) embedding of all the words in the sentences as different facets of the sentences. Since longer sentences have more words,

Table 3: The ROUGE F1 scores of different methods on CNN/Daily Mail dataset. The results with † are taken from Zheng and Lapata [2019]. The results with * are taken from Celikyilmaz et al. (2018).

| Setting | Method           | R-1  | R-2  | Len  |
|---------|-----------------|------|------|------|
| Unsup.  | Random          | 28.1 | 8.0  | 68.7 |
| Textgram (tfidf)† | 33.2 | 11.8 | -    |
| Textgram (BERT)† | 30.8 | 9.6  | -    |
| W Emb (GloVe) | 26.6 | 8.8  | 37.0 |
| No Sent Emb (GloVe) | 32.6 | 10.7 | 78.2 |
| Sent Emb (BERT) | 31.3 | 11.2 | 45.0 |
| Order   | Our c (K=3)     | 32.3 | 10.6 | 91.2 |
|         | Our c (K=10)    | 34.0 | 11.6 | 81.3 |
|         | Our c (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 | -    |
be modeled by a single embedding well. In Appendix B.1, the hypernym detection results also support this hypothesis.

Even though being slightly worse, the performances of **Ours (K=10)** remain strong compared with baselines. This implies that the similarity performances are not sensitive to the number of clusters as long as sufficiently large K is used because the model is able to output multiple nearly duplicated codebook embeddings to represent one facet (e.g., using two centers to represent the facet related to company in Figure [1]). The flexibility alleviates the issues of selecting K in practice. Finally, the strong performances in Turney (10) verify that our encoder respects the word order when composing the input sequence.

### 4 Related Work

Topic modeling (Blei, Ng, and Jordan 2003) has been extensively studied and widely applied due to its interpretability and flexibility of incorporating different forms of input features (Mimno and McCallum 2008). Cao et al. (2015); Srivastava and Sutton (2017) demonstrate that neural networks could be applied to discover semantically coherent topics. Instead of optimizing a global topic model, our goal is to efficiently discover different sets of topics/clusters on the words beside each (unseen) phrase or sentence.

Recently, Gupta et al. (2019) and Gupta et al. (2020) discover that global clustering could improve the representation of sentences and documents. In our work, we show that a local clustering could be used in several downstream applications, including word importance estimation for measuring sentence similarity. Whether combining global clustering and local clustering could lead to a further improvement is an interesting future research direction.

Sparse coding on word embedding space is used to model the multiple facets of a word (Faruqui et al. 2013; 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 and Nakayama 2018). Besides, to capture asymmetric relations such as hypernyms, words are represented as single or multiple regions in Gaussian embeddings (Vilnis and McCallum 2015; Athiwaratkun and Wil-

| Method | SemEval 2013 AUC | Turney (5) Accuracy | Turney (10) Accuracy |
|--------|------------------|---------------------|---------------------|
| BERT   | 54.7             | 66.7                | 29.2                | 15.5                |
|       | Avg 66.5         | 67.1                | 43.4                | 24.3                |
| GloVe  | Avg 79.5         | 73.7                | 25.9                | 12.9                |
| FCT LMT | Emb             | 80.3                | 72.8                | 45.6                |
|        | (K=10) Emb       | 85.6                | 77.1                | 49.4                |
| Ours   | SC               | 81.1                | 72.7                | 45.3                |
|        | (K=1) Emb        | 87.8                | 78.6                | 50.3                | 32.5 |

Table 4: 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. The results with † are taken from Yu and Dredze (2015).

In this work, we propose a framework for learning the co-occurring distribution of the words surrounding a sentence or a phrase. Even though there are usually only a few words that co-occur with each sentence, we demonstrate that the proposed models can learn to predict interpretable cluster centers conditioned on an (unseen) sentence.

In the sentence similarity tasks, the results indicate that the similarity between two sets of multi-facet embeddings well correlates with human judgments, and we can use the multi-facet embeddings to estimate the word importance and improve various widely-used similarity measurements in a pre-trained word embedding space such as GloVe. In a single-document extractive summarization task, we demonstrate multi-facet embeddings significantly outperform classic unsupervised sentence embedding or individual word embeddings. Finally, the results of phrase similarity tasks suggest that a single embedding might be sufficient to represent the co-occurring word distribution of a phrase.

### 5 Conclusions

In this work, we propose a framework for learning the co-occurring distribution of the words surrounding a sentence or a phrase. Even though there are usually only a few words that co-occur with each sentence, we demonstrate that the proposed models can learn to predict interpretable cluster centers conditioned on an (unseen) sentence.

In the sentence similarity tasks, the results indicate that the similarity between two sets of multi-facet embeddings well correlates with human judgments, and we can use the multi-facet embeddings to estimate the word importance and improve various widely-used similarity measurements in a pre-trained word embedding space such as GloVe. In a single-document extractive summarization task, we demonstrate multi-facet embeddings significantly outperform classic unsupervised sentence embedding or individual word embeddings. Finally, the results of phrase similarity tasks suggest that a single embedding might be sufficient to represent the co-occurring word distribution of a phrase.

### Acknowledgements

We thank Ao Liu and Mohit Iyyer for many helpful discussions and Nishant Yadav for suggesting several related work. We also thank the anonymous reviewers for their constructive feedback.

This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, in part using high
performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative, in part by the National Science Foundation (NSF) grant numbers DMR-1534431 and IIS-1514053. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

Ethics Statement

We propose a novel framework, neural architecture, and loss to learn multi-facet embedding from the co-occurring statistics in NLP. In this study, we exploit the co-occurring relation between a sentence and its nearby words to improve the sentence representation. In our follow-up studies, we discover that the multi-facet embeddings could also be used to learn other types of co-occurring statistics. For example, we can use the co-occurring relation between a scientific paper and its citing paper to improve paper recommendation methods in Bansal, Belanger, and McCallum (2016). Paul, Chang, and McCallum (2021) use the co-occurring relation between a sentence pattern and its entity pair to improve relation extraction in Verga et al. (2016). Chang et al. (2021) use the co-occurring relation between a context paragraph and its subsequent words to control the topics of language generation. In the future, the approach might also be used to improve the efficiency of document similarity estimation (Luan et al. 2020).

On the other hand, we improve the sentence similarity and summarization tasks in this work using the assumption that important words are more likely to appear in the nearby sentences. The assumption might be violated in some domains and our method might degrade the performances in such domains if the practitioner applies our methods without considering the validity of the assumption.

References

Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Lopez-Gazpio, I.; Maritxalar, M.; Mihalcea, R.; Rigau, G.; Uria, L.; and Wiebe, J. 2015. SemEval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In SemEval.

Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Mihalcea, R.; Rigau, G.; and Wiebe, J. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In SemEval.

Agirre, E.; Banea, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Mihalcea, R.; Rigau, G.; and Wiebe, J. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In SemEval.

Agirre, E.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; and Guo, W. 2013. * SEM 2013 shared task: Semantic textual similarity. In * SEM.*

Agirre, E.; Diab, M.; Cer, D.; and Gonzalez-Agirre, A. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In SemEval.

Arora, S.; Li, Y.; Liang, Y.; Ma, T.; and Risteski, A. 2018. Linear algebraic structure of word senses, with applications to polysemy. Transactions of the Association of Computational Linguistics 6: 483–495.

Arora, S.; Liang, Y.; and Ma, T. 2017. A Simple but Tough-to-beat Baseline for Sentence Embeddings. In ICLR.

Asaadi, S.; Mohammad, S. M.; and Kiritchenko, S. 2019. Big BiRD: A Large, Fine-Grained, Bigram Relatedness Dataset for Examining Semantic Composition. In NAACL-HLT.

Athiwaratkun, B.; and Wilson, A. 2017. Multimodal Word Distributions. In ACL.

Balles, L.; and Fischbacher, T. 2019. Holographic and other Point Set Distances for Machine Learning. URL https://openreview.net/forum?id=rJlpUiAcYX

Bansal, T.; Belanger, D.; and McCallum, A. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In RecSys.

Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9): 509–517.

Bird, S.; Klein, E.; and Loper, E. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. " O’Reilly Media, Inc."

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. Journal of machine Learning research 3(Jan): 993–1022.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5: 135–146.

Cao, Z.; Li, S.; Liu, Y.; Li, W.; and Ji, H. 2015. A novel neural topic model and its supervised extension. In AAAI.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-End Object Detection with Transformers. arXiv preprint arXiv:2005.12872 .

Celikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep Communicating Agents for Abractive Summarization. In NAACL-HLT.

Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; and Specia, L. 2017. SemEval-2017 Task 1: Semantic Text Similarity Multilingual and Crosslingual Focused Evaluation. In SemEval-2017.

Chang, H.-S.; Yuan, J.; Iyyer, M.; and McCallum, A. 2021. Changing the Mind of Transformers for Topically-Controllable Language Generation. In EACL.

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

Dubossarsky, H.; Grossman, E.; and Weinshall, D. 2018. Coming to your senses: on controls and evaluation sets in polysemy research. In EMNLP.
Eiter, T.; and Mannila, H. 1997. Distance measures for point sets and their computation. *Acta Informatica* 34(2): 109–133.

Faruqui, M.; Tsvetkov, Y.; Yogatama, D.; Dyer, C.; and Smith, N. A. 2015. Sparse Overcomplete Word Vector Representations. In *ACL*.

Gu, J.; Liu, Q.; and Cho, K. 2019. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics* 7: 661–676.

Gupta, V.; Saw, A.; Nokhiz, P.; Gupta, H.; and Talukdar, P. 2019. Improving document classification with multi-sense embeddings. In *ECAI*.

Gupta, V.; Saw, A.; Nokhiz, P.; Netrapali, P.; Rai, P.; and Talukdar, P. 2020. P-SIF: Document embeddings using partition averaging. In *AAAI*.

Han, R.; Gill, M.; Spirling, A.; and Cho, K. 2018. Conditional Word Embedding and Hypothesis Testing via Bayes-by-Backprop. In *EMNLP*.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *NeurIPS*.

Hoyer, P. O. 2002. Non-negative Sparse Coding. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*.

Huang, L.; Ji, H.; et al. 2017. Learning Phrase Embeddings from Paraphrases with GRUs. In *Proceedings of the First Workshop on Curation and Applications of Parallel and Comparable Corpora*.

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

Kobayashi, H.; Noguchi, M.; and Yatsuka, T. 2015. Summarization based on embedding distributions. In *EMNLP*.

Korkontzelos, I.; Tesch, T.; Zanzotto, F. M.; and Biemann, C. 2013. Semeval-2013 task 5: Evaluating phrasal semantics. In *SemEval 2013*.

Kumar, S.; and Tsvetkov, Y. 2019. Von Mises-Fisher Loss for Training Sequence to Sequence Models with Continuous Outputs. In *ICLR*.

Kusner, M.; Sun, Y.; Kolkin, N.; and Weinberger, K. 2015. From word embeddings to document distances. In *ICML*.

Lau, J. H.; Cook, P.; McCarthy, D.; Newman, D.; and Baldwin, T. 2012. Word sense induction for novel sense detection. In *EACL*.

Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A. R.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*.

Li, L. H.; Chen, P. H.; Hsieh, C.-J.; and Chang, K.-W. 2019. Efficient Contextual Representation Learning With Continuous Outputs. *Transactions of the Association for Computational Linguistics* 7: 611–624.

Lin, C.-Y.; and Hovy, E. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL-HLT*.

Liu, X.; Han, Z.; Wen, X.; Liu, Y.-S.; and Zwicker, M. 2019. L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In *Proceedings of the 27th ACM International Conference on Multimedia*.

Luan, Y.; Eisenstein, J.; Toutanova, K.; and Collins, M. 2020. Sparse, Dense, and Attentional Representations for Text Retrieval. *arXiv preprint arXiv:2005.00181*.

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

Milajevs, D.; Kartsaklis, D.; Sadrzadeh, M.; and Purver, M. 2014. Evaluating Neural Word Representations in Tensor-Based Compositional Settings. In *EMNLP*.

Mimno, D. M.; and McCallum, A. 2008. Topic Models Conditioned on Arbitrary Features with Dirichlet-multinomial Regression. In *UAI*.

Neelakantan, A.; Shankar, J.; Passos, A.; and McCallum, A. 2014. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. In *EMNLP*.

Newman-Griffis, D.; Lai, A. M.; and Fosler-Lussier, E. 2018. Jointly Embedding Entities and Text with Distant Supervision. In *Proceedings of the 3rd Workshop on Representation Learning for NLP (Repl4NLP)*.

Pagliardini, M.; Gupta, P.; and Jaggi, M. 2018a. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL-HLT*.

Pagliardini, M.; Gupta, P.; and Jaggi, M. 2018b. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL*.

Paul, R.; Chang, H.-S.; and McCallum, A. 2021. Multi-facet Universal Schema. In *EACL*.

Pavlíček, E.; Rastogi, P.; Ganitkevitch, J.; Van Durme, B.; and Callison-Burch, C. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*.

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

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Qin, K.; Li, C.; Pavlu, V.; and Aslam, J. A. 2019. Adapting RNN Sequence Prediction Model to Multi-label Set Prediction. In *NAACL-HLT*.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

Rezatofighi, S. H.; Kaskman, R.; Motlagh, F. T.; Shi, Q.; Cremers, D.; Leal-Taixé, L.; and Reid, I. 2018. Deep permutation net: learn to predict sets with unknown permutation...
and cardinality using deep neural networks. *arXiv preprint arXiv:1805.00613*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

Shu, R.; and Nakayama, H. 2018. Compressing Word Embeddings via Deep Compositional Code Learning. In *ICLR*.

Shwartz, V.; Goldberg, Y.; and Dagan, I. 2016. Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. In *ACL*.

Singh, S. P.; Hug, A.; Dieuleveut, A.; and Jaggi, M. 2020. Context mover’s distance & barycenters: Optimal transport of contexts for building representations. In *International Conference on Artificial Intelligence and Statistics*.

Srivastava, A.; and Sutton, C. A. 2017. Autoencoding Variational Inference For Topic Models. In *ICLR*.

Stern, M.; Chan, W.; Kiros, J.; and Uszkoreit, J. 2019. Insertion Transformer: Flexible Sequence Generation via Insertion Operations. In *ICML*.

Stewart, R.; Andriluka, M.; and Ng, A. Y. 2016. End-to-end people detection in crowded scenes. In *CVPR*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

Tieleman, T.; and Hinton, G. 2012. 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.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Verga, P.; Belanger, D.; Strubell, E.; Roth, B.; and McCallum, A. 2016. Multilingual Relation Extraction using Compositional Universal Schema. In *NAACL-HLT*.

Vilnis, L.; and McCallum, A. 2015. Word Representations via Gaussian Embedding. In *ICLR*.

Wang, T.; Cho, K.; and Wen, M. 2019. Attention-based mixture density recurrent networks for history-based recommendation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*.

Welleck, S.; Brantley, K.; Daumé III, H.; and Cho, K. 2019. Non-Monotonic Sequential Text Generation. In *ICML*.

Welleck, S.; Yao, Z.; Gai, Y.; Mao, J.; Zhang, Z.; and Cho, K. 2018. Loss Functions for Multiset Prediction. In *NeurIPS*.

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D. 2018a. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*.

Yang, Z.; Dai, Z.; Salakhutdinov, R.; and Cohen, W. W. 2018b. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*.

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

Zheng, H.; and Lapata, M. 2019. Sentence Centrality Revisited for Unsupervised Summarization. In *ACL*.
A Structure of Appendix

We conduct more comprehensive experiments and analyses in Section B. The details of our method and experiments (e.g., training algorithm, preprocessing, and hyperparameter settings) are presented in Section C and we visualize more codebook embeddings and the derived attention weights of the sentences in Section D.

B More Experiments

In the main paper, we show that multi-facet embeddings can improve the estimation of symmetric relations like similarity. To know whether they are also useful in asymmetric relations like entailment, we test our method on a hypernym detection dataset in Section B.2.

Due to the page limits, we cannot present all of our results in the main paper, so we put more comprehensive analyses for sentence similarity tasks in Section B.2, for extractive summarization in Section B.3, and for phrase similarity tasks in Section B.4. We also present the results of BERT Large model in Section B.5 as a reference. Section B.6 and B.7 provide some motivating examples for a sentence similarity task and for the extractive summarization, respectively.

B.1 Unsupervised Hypernymy Detection

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

Accordingly, the predicted codebook embeddings of a hyponym $S^h_{q^hypo}$ (e.g., brown dog), which cluster the embeddings of co-occurring words (e.g., eats), often reconstruct the embeddings of its hypernym $S^h_{q^{hyper}}$ (e.g., animal) better than the other way around (e.g., the embedding of flies cannot reconstruct the embeddings of brown dog well). That is, $Er(\tilde{F}_u(S^h_{q^{hyper}}), W(S^h_{q^{hyper}}))$ is smaller than $Er(\tilde{F}_u(S^h_{q^{hypo}}), W(S^h_{q^{hypo}}))$.

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

$$\text{Diff}(S^h_{q^{hypo}}, S^h_{q^{hyper}}) = Er(\tilde{F}_u(S^h_{q^{hyper}}), W(S^h_{q^{hypo}})) - Er(\tilde{F}_u(S^h_{q^{hypo}}), W(S^h_{q^{hypo}})).$$

(8)

where Er function is defined in equation [1].

The AUC of detecting hypernym among other relations and accuracy of detecting the hypernym direction are compared in Table [3]. Our methods outperform baselines, which only provide symmetric similarity measurement, and Ours (K=1) performs similarly compared with Ours (K=10).

B.2 More Analysis on Sentence Similarity

We design more experiments and present the results in Table 6 and Table 7 in order to answer the following research questions.

1. Is ignoring the order of co-occurring words effective in emphasizing the semantic side of the sentences?

| Method | Score | Dev | Test |
|--------|-------|-----|------|
| Model  | AUC   | Acc | AUC  | Acc  |
| BERT   | CLS   | 20.6| 50   | 21.3| 50 |
|        | Avg   | 25.6| 50   | 25.6| 50 |
| GloVe  | Avg   | 17.4| 50   | 17.7| 50 |
| Our c K10 | Diff | 29.4| 78.9| 29.6| 79.1|
| Our c K1 | Diff | 29.3| 82.7| 29.6| 81.0|

Table 5: Hypernym detection performances in the development and test set of HypeNet. AUC (%) refers to the area under precision and recall curve, which measures the quality of retrieving hypernym phrases. Acc (%) means the accuracy of predicting specificity given a pair of hypernym phrases.

| Method  | Score | Dev | Test |
|---------|-------|-----|------|
| Model   | All   | Low | All  | Low |
| Cosine  | Skip-thought | 43.2| 28.1| 30.4| 21.2|
| Avg     | ELMo  | 65.2| 62.4| 58.1| 58.1|
|         | Prob_avg | 70.5| 55.9| 61.1| 56.6|
|         | Avg    | 62.3| 42.1| 51.2| 39.1|
|         | Prob_avg | 72.1| 57.0| 57.8| 55.1|
| Avg     | Sent2Vec | 71.9| 63.6| 64.0| 61.0|
|         | Our a (GloVe) K10 | 76.1| 62.9| 71.5| 62.7|
|         | Our a (GloVe) K1 | 72.0| 56.6| 68.8| 55.7|
| SC      | NNSC clustering K10 | 38.6| 37.8| 23.4| 38.9|
|         | Our a (w2v) K10 | 54.7| 38.8| 43.9| 36.0|
|         | Our a (k-means) K10 | 37.8| 25.9| 29.5| 19.7|
|         | Our c (LSTM) K10 | 58.9| 49.2| 49.8| 46.4|
|         | Our c (GloVe) K10 | 63.0| 51.8| 52.6| 47.8|
| Prob_WMD | w2v | 72.9| 56.6| 62.1| 54.0|
|         | Our a (w2v) K10 | 73.6| 60.1| 63.5| 57.8|
| Prob_avg | w2v | 68.3| 53.7| 54.3| 50.9|
|         | Our a (w2v) K10 | 68.3| 56.8| 55.1| 53.1|
| SIF†    | w2v | 70.5| 56.9| 59.4| 54.7|
|         | Our a (w2v) K10 | 71.6| 60.9| 61.3| 57.6|
| Prob_WMD | GloVe | 75.1| 59.6| 63.1| 52.9|
|         | Our a (k-means) K10 | 72.5| 57.9| 60.3| 52.9|
|         | Our a (LSTM) K10 | 76.3| 63.2| 65.8| 57.4|
|         | Our a (GloVe) K10 | 76.2| 66.1| 66.3| 57.7|
|         | GloVe | 70.7| 56.6| 59.2| 54.8|
|         | Our a (k-means) K10 | 66.6| 53.4| 55.8| 51.8|
|         | Our a (LSTM) K10 | 71.7| 60.1| 61.3| 58.3|
|         | Our a (GloVe) K10 | 72.0| 60.5| 61.4| 59.3|
| SIF†    | GloVe | 75.1| 65.7| 63.2| 58.1|
|         | Our a (k-means) K10 | 71.5| 62.3| 61.5| 57.2|
|         | Our a (LSTM) K10 | 74.6| 66.9| 64.3| 60.9|
|         | Our a (GloVe) K10 | 75.2| 67.6| 64.6| 62.2|

Table 6: The Pearson correlation (%) in STS benchmarks. w2v means Word2Vec. Our * (k-means) means using the k-means loss rather than the NNSC loss. Our * (LSTM) means replacing the transformers in our encoder with bi-LSTM and replacing our transformer decoder with LSTM. Other abbreviations and symbols share the same meaning in Table 2.

To answer this question, we replace our transformer encoder with bi-LSTM and our transformer decoder with LSTM. Then, this architecture becomes very similar to skip-thought (Kiros et al. 2015) except that skip-thoughts decodes a sequence instead of a set, and we ignore the word order in the nearby sentences. As we can see in Table 3, Our c (LSTM) K10 SC performs much better than Skip-thought.
Table 7: Comparing Pearson correlation (%) of different unsupervised methods from STS 2012 to STS 2016. We highlight the best performance in each of the three blocks.

Cosine, which compute the cosine similarity between their sentence embeddings. This result further justifies our approach of ignoring the order of co-occurring words in our NNSC loss.

2. Is our word importance estimation generally useful for composing (contextualized) word embedding models?

We cannot apply our attention weights (i.e., Our a) to BERT because BERT uses word piece tokenization. Instead, we use the top layer of ELMo [Peters et al., 2018] as the contextualized word embedding, apply $\alpha + p(w)$ weighting multiplied with our attention weights in equation [8]. The results in Table [6] show that the performance of ELMo Prob Avg could also be boosted by our attention weighting even though our model is trained on GloVe semantic space. The importance weights from multiple embeddings can also help boost the performance of a version of Sent2Vec [Pagliardini, Gupta, and Jaggi, 2018b] that uses only unigram information.

3. Could our model be trained on word embedding space other than GloVe?

First, we train Word2Vec [Mikolov et al., 2013] (denoted as w2v) on the Wikipedia 2016 corpus. We then train our multi-facet embeddings to fit the Word2Vec embedding of co-occurring words in the Wikipedia 2016 corpus. The results in Table [6] show that Our a (w2v) K10 improves the performance using different scoring functions as we did in GloVe space.

4. How well could clustering-based multi-facet embeddings perform on long text sequences such as sentences?

Lots of the testing sentences in the STS benchmark are not observed in our training corpus. To test clustering-based multi-facet embeddings, we first average word embedding in every sentence into sentence embedding, and for each testing query sentence, we perform approximated nearest neighbor search using KDTree [Bentley, 1975] to retrieve 1000 most similar sentences. Then, we remove the stop words in the 1000 sentences and perform NNSC clustering on the rest of the words. Finally, we compute SC distance between two sets of cluster centers derived from testing sentence pairs and denote the baseline as NNSC clustering K10 SC in Table [6].

The testing time of this baseline is much slower than the proposed method due to the need for the nearest neighbor search, and its performance is also much worse. This result justifies our approach of predicting clustering centers.
5. How much better is NNSC loss compared with k-means loss?

In the method section, we mention that we adopt NNSC rather than k-means in our loss because k-means loss cannot generate diverse cluster centers in all of the neural architectures (including transformers and bi-LSTMs) we tried. We hypothesize that the k-means loss does not stably encourage predicted clusters to play different roles for reconstructing the embeddings of observed co-occurring words. We present the much worse results of the model using k-means loss in Table 6 to justify our usage of NNSC in our loss.

6. Could our method improve the similarity estimation of all kinds of datasets?

In Table 7, we compare the performance before and after applying our attention weights in the English part of STS 2012 (Agirre et al. 2012), 2013 (Agirre et al. 2013), 2014 (Agirre et al. 2014), 2015 (Agirre et al. 2015), and 2016 (Agirre et al. 2016). We categorize each of the dataset in different years based on either its source (forum, news, definition, caption, and education) or its characteristic (out of domain or similar).

Out of domain means the testing sentences are very different from our training corpus, Wikipedia 2016. deft-news from STS 2014 is included in this category because all the sentences in the dataset are lowercasing. Similar means there are lots of pairs in the datasets whose two sentences have almost the identical meaning.

From the Table 7, we can see that GloVe Prob_avg and GloVe Prob_WMD perform well compare with other baselines, and the attention weights from our multi-facet embedding stably boost GloVe Prob_avg and GloVe Prob_WMD except in the categories education, out of domain, and similar. Thus, we recommend adopting our method when the source of training and testing sentences are not too different from each other, and the task is not to identify duplicated sentences.

7. Are supervised methods such as sentence-BERT sensitive to the training data?

Table 8 compares the performance of sentence-BERT (Reimers and Gurevych 2019) trained on different data sources. We observe that the performance of sentence-BERT could be degraded when the distribution of training data is very different from that of testing data. For example, Sentence-BERT also does not perform well when the training sentence pairs tend to be similar with each other (e.g., in postediting and SMTeuroparl) or come from a writing style that is different from the style of testing sentence pairs (e.g., tweet-news and answers-students).

Furthermore, a supervised model trained by a limited amount of labels could perform worse than the unsupervised alternatives. For example, on STSB Dev, the weighted average of word embedding (Prob_avg) outputted by the BERT base model outperforms the sentence-BERT trained by 100 labels on average. Sentence-BERT model trained by SMTeuroparl is even worse than just averaging all the contextualized word embeddings in BERT on STSB Test.

### Table 8: The Pearson correlation (%) of sentence-BERT on STS benchmark.

| Data Source      | Dev     | Test    |
|------------------|---------|---------|
|                  | All Low | All Low |
| SMTeuroparl      | 2012    | 63.5    | 41.7   | 49.6  | 43.3  |
| surprise.SMTeuroparl | 2012    | 67.1    | 49.2   | 58.4  | 56.5  |
| postediting      | 2016    | 70.3    | 53.9   | 60.5  | 56.3  |
| tweet-news       | 2014    | 68.8    | 52.4   | 61.4  | 57.1  |
| answers-students | 2015    | 67.5    | 53.1   | 62.5  | 56.4  |
| headlines        | 2014    | 69.2    | 52.8   | 62.6  | 53.8  |
| plagiarism       | 2016    | 72.1    | 59.5   | 65.4  | 62.4  |
| belief           | 2015    | 71.9    | 53.5   | 65.7  | 59.1  |
| FNWN             | 2013    | 71.2    | 54.7   | 67.1  | 61.9  |
| headlines        | 2015    | 73.8    | 58.9   | 67.9  | 53.1  |
| question-question| 2016    | 75.0    | 63.2   | 69.3  | 65.5  |
| OnWN             | 2013    | 75.9    | 63.4   | 70.3  | 64.0  |
| OnWN             | 2014    | 74.9    | 63.5   | 70.6  | 65.6  |
| surprise.OnWN    | 2012    | 75.5    | 57.5   | 71.5  | 60.6  |
| Average          |         | 71.2    | 55.5   | 64.5  | 58.2  |

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 (*) usually outperforms 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, suppose our goal is simply to maximize the ROUGE F1 score given a fixed length of the summary without accessing the ground truth summary and sentence order information. In that case, the figure indicates that Ours (K=100) significantly outperform W Emb (GloVe) and 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 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).
### B.4 Experiments on More Phrase Similarity Datasets

We conduct the phrase similarity experiments on two recently proposed datasets, BiRD (Asaadi, Mohammad, and Kiritchenko 2019), and WikiSRS (Newman-Griffis, Lai, and Fosler-Lussier 2018), which contain ground truth phrase similarities derived from human annotations. BiRD and WikiSRS-Rel measure the relatedness of phrases and WikiSRS-Sim measures the similarity of phrases. The phrases are proper nouns in WikiSRS and are mostly common nouns in BiRD. Since the main goal of WikiSRS is to test the entity representation, we also test the different models trained on the corpus without lowercasing all the words.

The results are presented in Table 9. The multi-facet embedding performs similarly compared with single-facet embedding and is better than other baselines. This result confirms our findings in the main paper that the phrase similarity performance is not sensitive to the number of clusters \( K \).

### B.5 Comparison with BERT Large

In Table 12, we compare the size and running time of different models for sentence representation. As mentioned in Section 3.1, our model has fewer parameters than the BERT base model and uses much fewer 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 13 Table 14 Table 15 Table 16. As we can see, BERT large is usually better than BERT base in the similarity tasks but performs worse in the hyponym detection task. The BERT’s performance gains in similarity tasks might imply that training a larger version of our model might be a promising future direction.

### B.6 Motivating Examples in Sentence Similarity

In order to further understand when and why our methods perform well, we present some sentence pairs from the MSRvid dataset in STS 2012 in Table 17 and 18 on which our methods perform well. In Table 17, the first two sentence pairs have relatively high similarities but a lower ratio of overlapping words, so the baseline based on average word embedding (i.e., \( \text{Avg} \)) underestimates the similarities. Softly removing the stop words (i.e., \( \text{Prob} \text{Avg} \)) alleviates the problem, but the inverse frequency of words do not completely align with the importance of words in the sentences.

We visualize our predicted word importance and codebook embeddings in Table 18. Combining the estimated word importance with the inverse word frequency (i.e., \( \text{Prob} \text{Avg} + \text{Our a} \)) improves the performance. Finally, computing the similarity between the codebook embeddings (i.e., \( \text{Our c} \)) leads to the best results. The reason of the improvement might be that the unimportant words in the sentence often do not significantly affect the co-occurring word distribution. Take the second sentence pair as an example, mentioning “with the big eyes” does not change the sentence’s meaning and facets too much.

On the contrary, the last sentence pair in Table 17 has a low similarity but relatively higher word overlapping. Our model could infer that \( \text{riding a horse} \) is very different from \( \text{riding an elephant} \) because their co-occurring word distributions are different. The appearance of \( \text{riding a horse} \) implies that we are more likely to observe a race topic in nearby sentences, but \( \text{riding an elephant} \) increases the chance of seeing a movie topic instead.

### B.7 Motivating Examples in Extractive Summarization

In Table 19, we show the top three sentences that different methods choose to summarize a story about a photographer, Erik Johansson, and his artwork.

In this document, Lead-3 does not cover its main points because this article starts with a preamble. Our method selects the first sentence as a good summary because it highlights the main character of the story, Erik Johansson, and his art style. The selected sentences contain the aspects that cover several topics in the whole document.

Average word embedding baselines, Sent_Emb (GloVe) and Sent_Emb (BERT), select the sentences that focus on describing how his artwork is created. Nevertheless, the sentences are hard to understand without the context in the article. We hypothesize that the methods tend to avoid selecting the sentences with diverse aspects because after averaging the word embeddings, the resulting single embedding is not close to the embedding of words in the documents.

Finally, W_Emb (GloVe) and W_Emb (BERT) tend to select shorter sentences because we normalize the objective function by the sentence lengths. It is hard to remove the bias.

---

5The number is different from the one reported in Asaadi, Mohammad, and Kiritchenko (2019) because we use the uncased version (42B), the embedding space our model is trained on, and they use the cased version (840B).
of selecting shorter or longer sentences because each sentence is represented by a different number of embeddings.

### C Experimental Details

#### C.1 Training

The training algorithm of non-negative sparse coding (NNSC) loss can be seen in Algorithm [1]. Given the computational resource constraints, we keep our model simple enough to have the training loss nearly converged 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 than BERT but the architecture details in our transformer and most of its hyper-parameters are the same as those used in BERT.

The sparsity penalty weights on coefficient matrix $\lambda$ in equation [1] is set to be 0.4. 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. All words occurring less than 100 times in the training corpus are mapped to $\texttt{<unk>}$. The number of dimensions in transformers is set to be 300. For sentence representation, dropout on attention is 0.1. Its number of transformer layers on the decoder side is 5 for $K = 10$, and the number of transformer layers on the decoder side is set to be 1 for $K = 1$ because we do not need to model the dependency of output codebook embeddings. For phrase representation, the number of transformer layers on the decoder side is 2, and the dropout on attention is 0.5.

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 in equation [2]. 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 as shown in our phrase experiments. Furthermore, we 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 converge.

We use RegexpParser in NLTK [Bird, Klein, and Loper 2009] to detect the phrase boundary. We use the grammar $\texttt{NP: \langle JJ.*\rangle*<VBG>*<NN.*>+.}$ The sentence boundaries are detected using the rule-based pipeline in spaCy [spacy.io] and POS tags are also detected using spaCy [spacy.io].

The lowercased list we use for removing stop words includes $\texttt{@-@, =, <eos>, <unk>, disambiguation, etc. etc., }$

| Model    | Hidden size | #Parameters | Testing Time |
|----------|-------------|-------------|--------------|
| K=1      | 300         | 6.7M        | 9 ms         |
| K=10     | 300         | 13.7M       | 18 ms        |
| BERT Base| 768         | 86.0M       | 18 ms        |
| BERT Large| 1024      | 303.9M      | 65 ms        |
| Method                  | Score | Dev         | Test         |
|------------------------|-------|-------------|--------------|
| BERT Base              | Prob_avg | 72.1 | 57.0 | 57.8 | 55.1 |
| BERT Large             | Prob_avg | 74.3 | 61.0 | 65.0 | 60.0 |
| Our a (GloVe) K10      | Prob_avg | 72.0 | 60.5 | 61.4 | 59.3 |

Table 13: Compare BERT Large with Ours in Table 2

| Method                  | Score | Dev         | Test         |
|------------------------|-------|-------------|--------------|
| BERT Base              | W Emb | 31.2 | 11.2 | 44.9 |
| BERT Large             | W Emb | 31.1 | 11.0 | 46.8 |
| Our c (K=100)          | Bases | 35.0 | 12.8 | 92.9 |

Table 14: Compare BERT Large with Ours in Table 3

| Method                  | Score | Dev         | Test         |
|------------------------|-------|-------------|--------------|
| BERT Base              | V W   | 32.7 | 31.2 | 32.7 | 31.2 |
| BERT Large             | V W   | 29.6 | 81.0 | 12.8 | 29.3 |
| Our (GloVe) K10        | V W   | 76.2 | 62.6 | 58.1 | 60.0 |

Table 15: Compare BERT Large with Ours in Table 4

Algorithm 1: Training using NNSC loss

**Input**: Training corpus, sequence boundaries, and pre-trained word embedding.

**Output**: $F$

**Initialize**: $F$

**foreach** $I_i, W(N_i), W(N_{r_i})$ in training corpus do

- Run forward pass on encoder and decoder to compute $F(I_i)$
  - Compute $M^{D_i} = \arg\min_M ||F(I_i) - W(N_i)||^2 + \lambda ||M||_1 \forall k,j, 0 \leq M_{k,j} \leq 1$
  - Compute $M^{R_i} = \arg\min_M ||F(I_i) - W(N_{r_i})||^2 + \lambda ||M||_1 \forall k,j, 0 \leq M_{k,j} \leq 1$

- Run forward pass to compute $L_t$ in equation 2

- Treat $M^{D_i}$ and $M^{R_i}$ as constants, update $F$ by backpropagation

end

| Method                  | Dev         | Test         |
|------------------------|-------------|--------------|
| BERT Base (Avg)        | AUC 65.5    | 67.1         |
| BERT Large (Avg)       | 74.3        | 66.7         |
| Ours (K=1)             | 87.8        | 78.6         |

Table 16: Compare BERT Large with Ours in Table 5

C.2 Testing

The dataset sizes for sentence representation and phrase representation are summarized in Table 10 and Table 11 respectively. In our phrase experiments, we report the test sets of SemEval 2013 and Turney. For Turney dataset, we follow the evaluation setup of Yu and Dredze (2015); Huang, Ji 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 skip-thoughts, the hidden embedding size is set to be 600. To make the comparison fair, we retrain the skip-thoughts in Wikipedia 2016 for 2 weeks.

D Randomly Sampled Examples

We visualize the predicted codebook embeddings and the attention weights computed using equation 6 from 10 randomly selected sentences in our validation set (so most of them are unseen in our training corpus).

The first line of each example is always the preprocessed input sentence, where <unk> means an out-of-vocabulary placeholder. The attention weights are visualized using a red background. If one word is more likely to be similar to the words in the nearby sentences, it will get more attention and thus highlighted using a darker red color.

The format of visualized embeddings is similar to Table 1. Each row’s embedding is visualized by the nearest five neighbors in a GloVe embedding space and their cosine similarities to the codebook embedding.

```
<unk> Device or a backboard can be used to stabilize the remainder of the spinal column.
<eos>
```

**K=10**

- use 0.844 can 0.816 used 0.801
- bottom 0.757 front 0.703 sides 0.691
- spinal 0.914 nerve 0.739 Spinal 0.700
- increases 0.766 decreasing 0.756 increasing 0.749
- devices 0.836 device 0.787 wireless 0.703
- symptoms 0.726 chronic 0.715 disease 0.692
- polymeric 0.674 hydrophilic 0.644 hydrophobic 0.636
- backboard 0.899 hoop 0.555 dunks 0.547
- column 0.898 columns 0.771 Column 0.584
Table 17: Motivating examples for a sentence similarity task. The sentences are image captions from MSRvid dataset in STS 2012. GT means ground truth. All our methods here set $K = 10$.

Table 18: The predicted word importance and codebook embeddings on sentences from Table 17. The way of visualization is the same as that in Section D.
station. operate’s (located
Congress building Scribner of 01630 1908. York of - a)

(station 0.853 railway 0.834 stations 0.748

—K=1—

Aachen 0.706 Eifel 0.701 Freiburg 0.661

—K=3—

services 0.909 service 0.879 provider 0.708

—K=10—

station 0.989 stations 0.847 Station 0.756
Eifel 0.855 Harz 0.673 Cochem 0.642
railway 0.820 railways 0.794 trains 0.782
Germany 0.813 Berlin 0.766 Munich 0.751
north 0.885 south 0.877 east 0.869
building 0.867 buildings 0.794 construction 0.706
located 0.696 operated 0.669 operates 0.668
line 0.877 lines 0.739 Line 0.712
Railway 0.751 Rail 0.622 Railways 0.591
services 0.909 service 0.879 provider 0.708

—K=3—

station 0.918 stations 0.807 railway 0.779
located 0.802 area 0.770 situated 0.734
Aachen 0.706 Eifel 0.701 Freiburg 0.661

—K=1—

station 0.853 railway 0.834 stations 0.748

Table 19: Motivating examples for extractive summarization. The sentences come from a document in the validation set of CNN/Daily Mail. Index indicates the sentence order in the document. Ground truth means the summary from humans. The sentences in each method are ranked by its selection order. For example, our method selects the 6th sentence in the document first.

actor 0.672 starring 0.650 comedy 0.639
dancing 0.647 singing 0.598 raucous 0.593

K=3
really 0.857 thought 0.853 never 0.846
Carrie 0.881 Amanda 0.801 Rebecca 0.792
waitress 0.596 hostess 0.595 maid 0.591

K=1
knew 0.806 she 0.793 thought 0.788

The station building is located in the district of <unk>. These services operate on the Eifel Railway ( <unk> )

—K=10—

station 0.989 stations 0.847 Station 0.756
Eifel 0.855 Harz 0.673 Cochem 0.642
railway 0.820 railways 0.794 trains 0.782
Germany 0.813 Berlin 0.766 Munich 0.751
north 0.885 south 0.877 east 0.869
building 0.867 buildings 0.794 construction 0.706
located 0.696 operated 0.669 operates 0.668
line 0.877 lines 0.739 Line 0.712
Railway 0.751 Rail 0.622 Railways 0.591
services 0.909 service 0.879 provider 0.708

—K=3—

station 0.918 stations 0.807 railway 0.779
located 0.802 area 0.770 situated 0.734
Aachen 0.706 Eifel 0.701 Freiburg 0.661

—K=1—

station 0.853 railway 0.834 stations 0.748

President 0.778 Chairman 0.765 Committee 0.760
Ghana 0.924 Zambia 0.808 Cameroon 0.796

K=3
Committee 0.824 Council 0.740 Commission 0.733
election 0.810 elections 0.788 elected 0.774
Austria 0.923 Germany 0.793 Austrian 0.739
February 0.868 2011 0.865 2012 0.858
Consl 0.898 consul 0.761 consular 0.596
1995 0.970 1994 0.970 1993 0.969
Party 0.886 party 0.707 Parties 0.649
University 0.858 College 0.728 Graduate 0.727
retired 0.575 born 0.572 emeritus 0.543

K=1

Ghana 0.928 Zambia 0.807 Cameroon 0.796
Committee 0.824 Council 0.740 Commission 0.733
election 0.810 elections 0.788 elected 0.774
Austria 0.923 Germany 0.793 Austrian 0.739
February 0.868 2011 0.865 2012 0.858
Consl 0.898 consul 0.761 consular 0.596
1995 0.970 1994 0.970 1993 0.969
Party 0.886 party 0.707 Parties 0.649
University 0.858 College 0.728 Graduate 0.727
retired 0.575 born 0.572 emeritus 0.543

K=3

President 0.778 Chairman 0.765 Committee 0.760
Ghana 0.924 Zambia 0.808 Cameroon 0.806
1999 0.956 1998 0.953 1997 0.951

K=1

President 0.723 Affairs 0.664 Minister 0.658

 ISBN 0 = 8063 = 1367 = 6 Winthrop

John Winthrop’s Journal of New

England 1630 = 1649 New York NY

Charles Scribner’s Sons 1908 <eos>
The commune is represented in the Senate by Soledad Alvear (PDC) and Pablo <unk> (UDI) as part of the 8th senatorial constituency (Santiago = East). Haha Sound is the second album by the British indie electronic band Broadcast. <eos>

As a result, the Hellfire Club believed that it would be in their best interests to summon the Phoenix and merge it with Jean Grey via a ritual. <eos>
He designed a system of streets which generally followed the contours of the area's topography. Residential neighborhoods stretched out from a commercial and service-sector core creating a commercial and service-sector core.

development 0.739 significant 0.720 providing 0.719 streets 0.831 city 0.793 neighborhoods 0.786 County 0.669 Riverside 0.662 City 0.649

buildings 0.765 area 0.761 areas 0.753