Con-S2V: A Generic Framework for Incorporating Extra-Sentential Context into Sen2Vec

Tanay Kumar Saha[1](✉), Shafiq Joty[2], and Mohammad Al Hasan[1]

[1] Indiana University – Purdue University Indianapolis, Indianapolis, IN 46202, USA
{tksaha,alhasan}@iupui.edu
[2] Nanyang Technological University, Singapore, Singapore
srjoty@ntu.edu.sg

Abstract. We present a novel approach to learn distributed representation of sentences from unlabeled data by modeling both content and context of a sentence. The content model learns sentence representation by predicting its words. On the other hand, the context model comprises a neighbor prediction component and a regularizer to model distributional and proximity hypotheses, respectively. We propose an online algorithm to train the model components jointly. We evaluate the models in a setup, where contextual information is available. The experimental results on tasks involving classification, clustering, and ranking of sentences show that our model outperforms the best existing models by a wide margin across multiple datasets.
Code related to this chapter is available at:
https://github.com/tksaha/con-s2v/tree/jointlearning
Data related to this chapter are available at: https://www.dropbox.com/sh/ruhsi3c0unn0nko/AAAgVnZpojvXx9loQ21WP_MYa?dl=0

Keywords: Sen2Vec · Extra-sentential context
Embedding of sentences

1 Introduction

For many text processing tasks that involve classification, clustering, or ranking of sentences, vector representation of sentences is a prerequisite. Bag-of-words (BOW) based vector representation has been used traditionally in these tasks, but in recent years, it has been shown that *distributed representation*, in the form of condensed real-valued vectors, learned from *unlabeled* data outperforms BOW based representations [1]. It is now well established that distributed representation captures semantic properties of linguistic units and yields better generalization [2,3].

However, most of the existing methods to devise distributed representation for sentences consider only the content of a sentence or its grammatical structure [1,4] disregarding its context. But, sentences rarely stand on their own in a text, rather the meaning of one sentence depends on the meaning of others within its context. For example, sentences in a text segment address a common topic [5]. At a finer level, sentences are connected by certain coherence relations (e.g., *elaboration*, *contrast*) and acts together to express a coherent message holistically [6].

Our work is built on the following hypothesis: *since the meaning of a sentence can be best interpreted within its context, its representation should also be inferred from its context.* Several recent works attempt to learn sentence representations which support the above hypothesis by utilizing words or word sequences of neighboring sentences [7,8]. However, by learning representations to predict content of neighboring sentences, existing methods may learn semantic and syntactic properties that are more specific to the neighbors rather than the sentence under consideration. Furthermore, these methods either make a simple BOW assumption or disregard context when extracting a sentence vector.

In contrast to the existing works, we consider neighboring sentences as *atomic* linguistic units, and propose novel methods to learn the representations of a given sentence by jointly modeling content and context of a sentence. Our work considers two types of context: *discourse* and *similarity.* The discourse context of a given sentence $\mathbf{v}$ comprises with its previous and the following sentence in the text. On the other hand, the similarity context is based on a user defined similarity function; thus it allows any sentences in the text to be in the context of $\mathbf{v}$ depending on how similar that sentence is with $\mathbf{v}$ based on the chosen function.

Our proposed computational model for learning the vector representation of a sentence comprises three components. The first component models the content by asking the sentence vector to predict its constituent words. The second component models the *distributional* hypotheses [9] of a context. The distributional hypothesis conveys that the sentences occurring in similar contexts should have similar representations. Our computation model captures this preference by using a context prediction component. Finally, the third component models the *proximity* hypotheses of a context, which also suggests that sentences that are proximal should have similar representations. Our method achieves this preference by using a Laplacian regularizer. To this end, we consider the sentence representation learning problem as an optimization problem whose objective function is built with expressions from the above three components and we solve this optimization problem by using an efficient online algorithm.

1.1 Summary of Results

We evaluate our sentence representation for learning models on multiple information retrieval tasks: topic classification and clustering, and single-document summarization. Our evaluation on these tasks across multiple datasets shows impressive results for our model, which outperforms the best existing models by up to 7.7 F_1-score in classification, 15.1 V-score in clustering, 3.2 ROUGE-1 score in summarization. We found that the discourse context performs better on topic classification and clustering tasks, while similarity context performs better on summarization. We make our code[1] and pre-processed dataset[2] publicly available.

2 Related Work

Extensive research has been conducted on learning distributed representation of linguistic units both in supervised (task-specific) and in unsupervised (task-agnostic) settings. In this paper, we focus on learning *sentence* representations from *unlabeled* data.

Two log-linear models are proposed in [10] for learning representations of words: continuous bag-of-words (CBOW) and continuous skip-gram. CBOW learns word representations by predicting a word given its (intra-sentential) context. The skip-gram model on the other hand learns representation of a word by predicting the words in a context. [11] proposed C-PHRASE, an extension of CBOW, where the context is extracted from a syntactic parse of the sentence. Simple averaging or addition of word vectors to construct sentence vectors often works well [12], and serves as baselines in our experiments.

CBOW and skip-gram models are extended in [1] to sentences and documents by proposing distributed memory (DM) and distributed bag-of-words (DBOW) models. In these models, similar to words, a sentence is mapped to an unique id and its representation is learned using contexts of words in the sentence. DM predicts a word given a context and the sentence id, where DBOW predicts all words in a context independently given the sentence id. Since these models are agnostic to sentence structure, they are quite fast to train. However, they disregard extra-sentential context of a sentence.

Sequential denoising autoencoder (SDAE) and FastSent are proposed in [8] for modeling sentences. SDAE employs an encoder-decoder framework, similar to neural machine translation (NMT) [13], to denoise an original sentence (target) from its corrupted version (source). FastSent is an additive model to learn sentence representation from word vectors. Given a sentence as BOW, it predicts the words of its adjacent sentences. The auto-encode version of FastSent also predicts the words of the current sentence. SDAE composes sentence vectors sequentially, but it disregards context of the sentence. FastSent, on the other hand, is a BOW model that considers neighboring sentences.

Another context-sensitive model is Skip-Thought [7], which uses the NMT framework to predict adjacent sentences (target) given a sentence (source). Since the encoder and the decoder use recurrent layers to compose vectors sequentially, SDAE and Skip-Thought are very slow to train. Furthermore, by learning representations to predict content of neighboring sentences, these methods (FastSent and Skip-Thought) may learn linguistic properties that are more specific to the neighbors rather than the sentence under consideration.

[1] https://github.com/tksaha/con-s2v/tree/jointlearning.
[2] https://www.dropbox.com/sh/ruhsi3c0unn0nko/AAAgVnZpojvXx9loQ21WP_MYa?dl=0.

By contrast, we encode a sentence by treating it as an atomic unit like word, and similar to DBOW, we predict the words to model its content. Similarly, context is considered in our model by treating neighboring sentences as atomic units. This abstraction makes our model quite fast to train.

3 The Model

We hypothesize that the representation of a sentence depends not only on its content words, but also on other sentences in its context. It will be convenient to present our learning model using graph.

Let $G = (V, E)$ be a graph, where $V = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{|V|}\}$ represents the set of sentences in our corpus, and edge $(\mathbf{v}_i, \mathbf{v}_j) \in E$ reflects some relation between sentences $\mathbf{v}_i$ and $\mathbf{v}_j$. A sentence $\mathbf{v}_i \in V$ is a sequence of words $(v_i^1, v_i^2, \cdots, v_i^M)$, each coming from a dictionary $\mathcal{D}$. We define $\mathcal{N}(\mathbf{v}_i)$ as the set of neighboring sentences of $\mathbf{v}_i$, which constitutes extra-sentential context for sentence $\mathbf{v}_i$. We formalize relation between sentences and context later in Sect. 3.3.

Let $\phi : V \to \mathbb{R}^d$ be the mapping function from sentences to their distributed representations, i.e., real-valued vectors of d dimensions. Equivalently, ϕ can be thought of as a look-up matrix of size $|V| \times d$, where $|V|$ is the total number of sentences. Our aim is to learn $\phi(\mathbf{v}_i)$ by incorporating information from two different sources: (i) the content of the sentence, $\mathbf{v}_i = (v_i^1, v_i^2, \cdots, v_i^M)$; and ($ii$) the context of the sentence in the graph, i.e., $\mathcal{N}(\mathbf{v}_i)$. Let $\langle v_i \rangle_t^l = (v_i^{t-l}, \ldots, v_i^t, \ldots, v_i^{t+l})$ denote a window of $2l + 1$ words around the word v_i^t in sentence $\mathbf{v}_i$, and $C_i = |\mathcal{N}(\mathbf{v}_i)|$ denote the context size for sentence $\mathbf{v}_i$. We define our model as a combination of three different loss functions:

$$J(\phi) = \sum_{\mathbf{v}_i \in V} \sum_{\substack{v \in \langle v_i \rangle_t^l \\ j \sim \mathcal{U}(1, C_i)}} \big[\mathcal{L}_c(\mathbf{v}_i, v) + \mathcal{L}_g(\mathbf{v}_i, \mathbf{v}_j) + \mathcal{L}_r(\mathbf{v}_i, \mathcal{N}(\mathbf{v}_i))\big] \tag{1}$$

where loss $\mathcal{L}_c(\mathbf{v}_i, v)$ is used to model the content of a sentence $\mathbf{v}_i$, and other two loss functions are for modeling the context of the sentence. We define $\mathcal{L}_c(\mathbf{v}_i, v)$ as the cost for predicting the content word v using the sentence vector $\phi(\mathbf{v}_i)$ as input features. Similarly, $\mathcal{L}_g(\mathbf{v}_i, \mathbf{v}_j)$ is defined as the cost for predicting a neighboring node $\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)$, again using the sentence vector $\phi(\mathbf{v}_i)$ as input. The third loss $\mathcal{L}_r(\mathbf{v}_i, \mathcal{N}(\mathbf{v}_i))$ is a graph smoothing regularizer defined over the context of $\mathbf{v}_i$, which encourages two proximal sentences to have similar representations.

To learn the representation of a sentence $\mathbf{v}_i$ using Eq. 1, for each content word v in a window $\langle v_i \rangle_t^l$, we sample a neighboring node $\mathbf{v}_j$ from $\mathcal{N}(\mathbf{v}_i)$, uniformly at random, with replacement. We use the sentence vector $\phi(\mathbf{v}_i)$ (under estimation) to predict v and $\mathbf{v}_j$, respectively. A regularization is performed to smooth the estimated vector with respect to the neighboring vectors. Figure 1 shows instances of our model for learning the representation of sentence $\mathbf{v}_2$ within a context of two other sentences: $\mathbf{v}_1$ and $\mathbf{v}_3$.

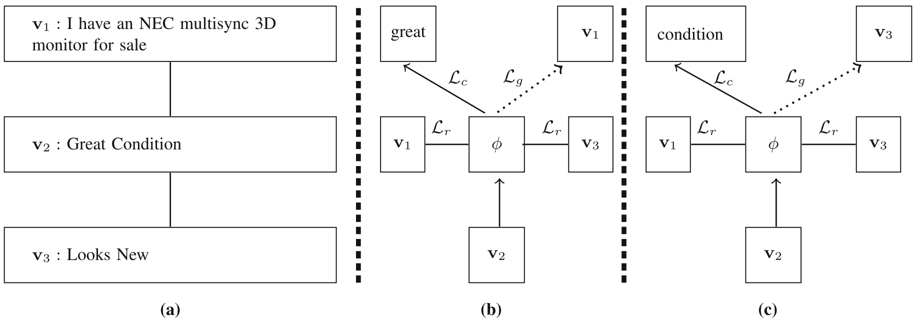

Fig. 1. Two instances (see (b) and (c)) of our model for learning representation of sentence $\mathbf{v}_2$ within a context of two other sentences: $\mathbf{v}_1$ and $\mathbf{v}_3$ (see (a)). Directed and undirected edges indicate prediction loss and regularization loss, respectively, and dashed edges indicate that the node being predicted is randomly sampled. (Collected from: 20news-bydate-train/misc.forsale/74732. The central topic is "forsale".)

We can use the standard *softmax* function for the prediction tasks. Formally, the negative log probability of an item o (can be a content word or a neighboring node) given the sentence vector $\phi(\mathbf{v}_i)$ is

$$-\log p(o|\mathbf{v}_i) = -\mathbf{w}_o^T \phi(\mathbf{v}_i) + \log \sum_{o' \in \mathcal{O}} \exp\left(\mathbf{w}_{o'}^T \phi(\mathbf{v}_i)\right) \tag{2}$$

where $\mathcal{O}$ is the set of all possible items (i.e., vocabulary of words or set of all nodes), and $\mathbf{w}$'s are the weight parameters. Optimization is typically performed using gradient-based online methods, such as stochastic gradient descend (SGD), where gradients are obtained via backpropagation.

Unfortunately, training could be impractically slow on large corpora due to summation over all items in $\mathcal{O}$ (Eq. 2), which needs to be performed for every training instance $(\mathbf{v}_i, o)$. Several methods have been proposed to address this issue including *hierarchical softmax* [14], *noise contrastive estimation* [15], and *negative sampling* [16]. We use negative sampling, which samples negative examples to approximate the summation term. Specifically, for each training instance $(\mathbf{v}_i, o)$, we add S negative examples $\{(\mathbf{v}_i, o^s)\}_{s=1}^S$ by sampling o^s from a known noise distribution ψ (e.g., *unigram*, *uniform*). The negative log probability in Eq. 2 is then formulated as such to discriminate a *positive* instance o from a *negative* one o^s:

$$-\log \sigma\left(\mathbf{w}_o^T \phi(\mathbf{v}_i)\right) - \log \sum_{s=1}^{S} \mathbb{E}_{o^s \sim \psi}\, \sigma\left(-\mathbf{w}_{o^s}^T \phi(\mathbf{v}_i)\right) \tag{3}$$

where σ is the sigmoid function defined as $\sigma(x) = 1/(1+e^{-x})$, and $\mathbf{w}$'s and $\phi(\mathbf{v}_i)$ are similarly defined as before. Negative sampling thus reduces the number of computations needed from $|\mathcal{O}|$ to $S+1$, where S is a small number (5 – 10) compared to the vocabulary size $|\mathcal{O}|$ (26K – 139K).

In the following, we elaborate on our methods for modeling content and context of a sentence.

3.1 Modeling Content

Our approach for modeling content of a sentence is similar to the distributed bag-of-words (DBOW) model of [1]. Given an input sentence $\mathbf{v}_i$, we first map it to a unique vector $\phi(\mathbf{v}_i)$ by looking up the corresponding vector in the sentence embedding matrix ϕ. We then use $\phi(\mathbf{v}_i)$ to predict each word v sampled from a window of words in $\mathbf{v}_i$. Formally, the loss for modeling content using negative sampling is

$$\mathcal{L}_c(\mathbf{v}_i, v) = -\log\sigma\left(\mathbf{w}_v^T \phi(\mathbf{v}_i)\right) - \log \sum_{s=1}^{S} \mathbb{E}_{v^s \sim \psi_c}\, \sigma\left(-\mathbf{w}_{v^s}^T \phi(\mathbf{v}_i)\right) \tag{4}$$

where σ is the sigmoid function as defined before, $\mathbf{w}_v$ and $\mathbf{w}_{v^s}$ are the weight vectors associated with words v and v^s, respectively, and ψ_c is the noise distribution from which v^s is sampled. In our experiments, we use unigram distribution of words raised to the 3/4 power as our noise distribution, in accordance to [16].

By asking the same sentence vector (under estimation) to predict its words, the content model captures the overall semantics of the sentence. The model has $O(d \times (|V| + |\mathcal{D}|))$ parameters.

3.2 Modeling Context

Our content model above attempts to capture the overall meaning of a sentence by looking at its words. However, sentences in a text are not independent, rather the meaning of a sentence depends on its neighboring sentences. For instance, consider the second and the third sentences in Fig. 1(a). When the sentences are considered in isolation, one cannot understand what they are talking about (i.e., *monitor for sale*). This suggests, since meaning of a sentence can be best interpreted within its context, the representation of the sentence should also be inferred from its context. We distinguish between two types of contextual relations between sentences: (*i*) distributional similarity, and (*ii*) proximity. Each of these corresponds to a loss in our model (Eq. 1), as we describe them below.

Modeling Distributional Similarity: Our sentence-level distributional hypothesis [9] is that if two sentences share many neighbors in the graph, their representations should be similar. We formulate this in our model by asking the sentence vector to predict its neighboring nodes. More formally, the loss for predicting a neighboring node $\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)$ using the sentence vector $\phi(\mathbf{v}_i)$ is

$$\mathcal{L}_g(\mathbf{v}_i, \mathbf{v}_j) = -\log\sigma\left(\mathbf{w}_j^T \phi(\mathbf{v}_i)\right) - \log \sum_{s=1}^{S} \mathbb{E}_{j^s \sim \psi_g}\, \sigma\left(-\mathbf{w}_{j^s}^T \phi(\mathbf{v}_i)\right) \tag{5}$$

where $\mathbf{w}_j$ and $\mathbf{w}_j^s$ are the weight vectors associated with nodes $\mathbf{v}_j$ and $\mathbf{v}_j^s$, respectively, and ψ_g is the noise distribution over nodes from which $\mathbf{v}_j^s$ is sampled. Similar to our content model, ψ_g is defined as unigram distribution of nodes raised to the 3/4 power. The unigram distribution is computed based on the occurrences of the nodes in the neighborhood sets, $\{\mathcal{N}(\mathbf{v}_i)\}_{i=1}^{|V|}$. This model has $O(d \times (|V| + |V|))$ parameters.

Modeling Proximity: According to our proximity hypothesis, sentences that are proximal in their contexts, should have similar representations. We use a Laplacian regularizer to model this. Formally, the regularization loss for modeling proximity for a sentence $\mathbf{v}_i$ in its context $\mathcal{N}(\mathbf{v}_i)$ is

$$\mathcal{L}_r(\mathbf{v}_i, \mathcal{N}(\mathbf{v}_i)) = \frac{\lambda}{C_i} \sum_{\mathbf{v}_k \in \mathcal{N}(\mathbf{v}_i)} ||\phi(\mathbf{v}_i) - \phi(\mathbf{v}_k)||^2 \tag{6}$$

where $C_i = |\mathcal{N}(\mathbf{v}_i)|$ as defined before, and λ is a hyper-parameter to control regularization strength.

Rather than including the Laplacian as a regularizer in the objective function, another option is to first learn sentence embeddings using other components of the model (e.g., first two loss functions in Eq. 1), and then *retrofit* them using the Laplacian as a post-processing step. [17] adopted this approach to incorporate lexical semantics (e.g., synonymy, hypernymy) into word representations. We compare our approach with retrofitting in Sect. 5.

3.3 Context Types

In this section we characterize context of a sentence. We distinguish between two types of context: discourse context and similarity context.

Discourse Context: The discourse context of a sentence is formed by the previous and the following sentences in the text. As explained before, the order of the sentences carries important information. For example, adjacent sentences in a text are logically connected by certain coherence relations (e.g., *elaboration*, *contrast*) to express the meaning [6]. On a coarser level, sentences in a text segment (e.g., paragraph) address a common (sub)topic [5]. The discourse context thus captures both coherence and topic structures of a text.

Similarity Context: While the discourse context covers important discourse phenomena like *coherence* and *cohesion* [18], some applications might require a context type that is based on more direct measures of similarity, and considers relations between all possible sentences in a document and possibly across multiple documents. For example, graph-based methods for topic segmentation [19] and summarization [20] rely on complete graphs of sentences, where edge weights represent cosine similarity between sentences. In an empirical evaluation of data structures for representing discourse coherence, [21] advocate for a graph representation of discourse allowing non-adjacent connections.

Our similarity context allows any other sentence in the corpus to be in the context of a sentence depending on how similar they are. To measure the similarity, we first represent the sentences with vectors learned by Sen2Vec [1], then we measure the cosine between the vectors. We restrict the context size of a sentence for computational efficiency, while still ensuring that it is informative enough. We achieve this by imposing two kinds of constraints. First, we set thresholds for intra- and across-document connections: sentences in a document are connected only if their similarity value is above 0.5, and sentences across documents are connected only if their similarity is above 0.8. Second, we allow up to 20 most similar neighbors.

3.4 Training

Algorithm 1 illustrates the SGD-based algorithm to train our model. We first initialize the model parameters; the sentence vectors ϕ are initialized with small random numbers sampled from uniform distribution $\mathcal{U}(-0.5/d, 0.5/d)$, and the weight parameters $\mathbf{w}$'s are initialized with zero. We then compute the noise distributions ψ_c and ψ_g for $\mathcal{L}_c(\mathbf{v}_i, v)$ and $\mathcal{L}_g(\mathbf{v}_i, \mathbf{v}_j)$ losses, respectively.

We iterate over the sentences in our corpus in each epoch of SGD, as we learn their representations. Specifically, to estimate the representation of a sentence, for each word token in the sentence, we take three gradient steps to account for the three loss functions in Eq. 1. By making the same number of gradient updates, the algorithm weights equally the contributions of content and context.

Algorithm 1. Training Con-S2V with SGD

```
Input  : set of sentences $V$, graph $G = (V, E)$
Output: learned sentence vectors $\phi$
1. Initialize model parameters: $\phi$ and $\mathbf{w}$'s;
2. Compute noise distributions: $\psi_c$ and $\psi_g$
3. repeat
    for each sentence $\mathbf{v}_i \in V$ do
        for each content word $v \in \mathbf{v}_i$ do
            a) Generate a positive pair $(\mathbf{v}_i, v)$ and $S$ negative pairs $\{(\mathbf{v}_i, v^s)\}_{s=1}^{S}$
               using $\psi_c$;
            b) Take a gradient step for $\mathcal{L}_c(\mathbf{v}_i, v)$;
            c) Sample a neighboring node $\mathbf{v}_j$ from $\mathcal{N}(\mathbf{v}_i)$;
            d) Generate a positive pair $(\mathbf{v}_i, \mathbf{v}_j)$ and $S$ negative pairs $\{(\mathbf{v}_i, \mathbf{v}_j^s)\}_{s=1}^{S}$
               using $\psi_g$;
            e) Take a gradient step for $\mathcal{L}_g(\mathbf{v}_i, \mathbf{v}_j)$;
            f) Take a gradient step for $\mathcal{L}_r(\mathbf{v}_i, \mathcal{N}(\mathbf{v}_i))$;
        end
    end
until convergence;
```

4 Evaluation Tasks

Different methods have been proposed to evaluate sentence representation models [8]. However, unlike most existing methods, our model learns representation of a sentence by exploiting contextual information in addition to the content.[3] To be able to evaluate our models, we thus require corpora of annotated sentences with ordering and document boundaries preserved, i.e., documents with sentence-level annotations. To the best of our knowledge, no previous work has used or released such corpora for learning sentence representation. In this work, we automatically create large corpora of documents with sentence-level topic annotations, which are then used to evaluate our models on topic *classification* and *clustering* tasks. Additionally, we evaluate our models on a *ranking* task of generating *extractive* single-document summaries. In the interest of coherence, we present the summarization task, followed by topic classification and clustering.

4.1 Extractive Summarization

Extractive summarization is often considered as a ranking problem, where the goal is to select the most important sentences to form an abridged version of the source document(s) [22]. Unsupervised methods are the predominant paradigm for determining sentence importance. We use the popular graph-based algorithm LexRank [20]. The input to LexRank is a graph, where nodes represent sentences and edges represent cosine similarity between *vector representations* (learned by models) of the two corresponding sentences. We run the PageRank [23] on the graph to compute importance of each sentence in the graph.[4] The top-ranked sentences are extracted as the summary sentences.

Data: We use the benchmark datasets from DUC-2001 and DUC-2002, and evaluate our representation models on the official task of generating a 100-words summary for each document in the datasets.[5] The sentence representations are learned independently a priori from the same source documents. Table 1 shows some basic statistics about the datasets. For each document, 2–3 short (≈100 words) human authored reference summaries are available, which we use as gold summaries for automatic evaluation.

Metric: We use the widely used automatic evaluation metric ROUGE [24] to evaluate the system-generated summaries. ROUGE computes n-gram recall between a system-generated summary and a set of human-authored reference summaries. Among the variants, ROUGE-1 (i.e., $n = 1$) has been shown to correlate well with human judgments for short summaries [24]. Therefore, we only report ROUGE-1 in this paper.

Table 1. Basic statistics about the DUC datasets

Dataset	#Doc.	#Avg. sen.	#Avg. sum.
DUC 2001	486	40	2.17
DUC 2002	471	28	2.04

[3] For this reason, we did not evaluate our models on tasks previously used to evaluate sentence representation models.

[4] The dumping factor in the PageRank was set to 0.85.

[5] http://www-nlpir.nist.gov/projects/duc/guidelines.

4.2 Topic Classification and Clustering

We evaluate our models by measuring how effective the learned vectors are when they are used as features for classifying or clustering the sentences into topics. Text categorization has now become a standard in evaluating cross-lingual word embeddings [25]. We use a MaxEnt classifier and a K-means++ [26] clustering algorithm for classification and clustering tasks, respectively.

Data: We use the standard text categorization corpora: *Reuters-21578* and *20-Newsgroups*. Reuters-21578 (henceforth Reuters) is a collection of $21,578$ news documents covering 672 topics.[6] 20-Newsgroups (henceforth Newsgroups) is a collection of about $20,000$ news articles organized into 20 different topics.[7] We used the standard train-test splits (*ModApte* split for Reuters) split, and selected documents only from the 8 most frequent topics in both datasets. The selected topics for Reuters dataset are: *acq, crude, earn, grain, interest, money-fx, ship,* and *trade*. The topics selected for Newsgroups dataset are: *sci.space*, *sci.med*, *talk.politics.guns*, *talk.politics.mideast*, *rec.autos*, *rec.sport.baseball*, *comp.graphics*, and *soc.religion.christian*.

Table 2. Statistics about Reuters and Newsgroups.

Dataset	#Doc.	Total #sen.	Annot. #sen.	Train #sen.	Test #sen.	#Class
Reuters	9,001	42,192	13,305	7,738	3,618	8
Newsgroups	7,781	95,809	22,374	10,594	9,075	8

Generating Sentence-level Topic Annotations: As mentioned above, both *Newsgroups* and *Reuters* datasets come with document-level topic annotations. However, we need sentence-level annotations for our evaluation. One option is to assume that all the sentences of a document share the same topic label as the document. However, this naive assumption induces a lot of noise. Although sentences in a document collectively address a common topic, not all sentences are directly linked to that topic, rather they play supporting roles. To minimize this noise, we employ our extractive summarizer introduced in Sect. 4.1 to select the top 20% sentences of each document as representatives of the document, and assign them the same topic label as the topic of the document. We used Sen2Vec [1] representation to compute cosine similarity between two sentences in LexRank. Table 2 shows statistics of the resulting datasets. Note that the sentence vectors are learned independently from an entire dataset (#Total Sen. column in Table 2).

Metrics: We report raw **acc**uracy, macro-averaged $\mathbf{F_1}$-score, and Cohen's $\boldsymbol{\kappa}$ for comparing classification performance. For clustering, we report **V**-measure [27] and adjusted mutual information or **AMI** [28]. We use all the annotated sentences (train+test in Table 2) for comparing clustering performance.

[6] http://kdd.ics.uci.edu/databases/reuters21578/.
[7] http://qwone.com/~jason/20Newsgroups/.

5 Experiments

In this section, we present our experiments — the models we compare, their settings, and the results.

5.1 Models Compared

We compare our representation learning model against several baselines and existing models. We also experiment with a number of variations of our proposed model considering which components of the model are active, types of context, and how we incorporate the context. For clarity, in our tables we show results divided into five evaluation groups:

(I) Existing Distributed Models: This group includes Sen2Vec [1], W2V-avg, C-Phrase [11], FastSent [8], and Skip-Thought [7].

We used Mikolov's implementation[8] of **Sen2Vec**, which gave better results than gensim's version when validated on the sentiment treebank [29]. Following the recommendation by [1], we concatenate the vectors learned by DM and DBOW models. The concatenated vectors also performed better on our tasks.

For **W2V-avg**, we obtain a sentence vector by averaging the word vectors learned by training a skip-gram Word2Vec [16] on our training set. Since code for **C-Phrase** is not publicly available, we use pre-trained word vectors (of 300 dimensions) available from author's webpage.[9] We first add the word vectors to obtain a sentence vector, then we normalize the vector with l_2 normalization. Normalized vectors performed better on our tasks than the ones obtained by simple addition.

We use the auto-encode version of **FastSent** (FastSent+AE) since it considers both content and context of a sentence. For **Skip-Thought**, we use the pre-trained combine-skip model that concatenates the vectors encoded by uni- and bi-skip models.[10] The resultant vectors are of 4800 dimensions. The model was originally trained on the BookCorpus[11] with a vocabulary size of $20K$ words, however, it uses publicly available CBOW Word2Vec vectors to expand the vocabulary size to $930,911$ words.

(II) Non-distributed Model: We use **Tf-Idf** as our non-distributed baseline, where a sentence is represented by tf*idf weighting of its words.

(III) Retrofitted Models: We compare our approach of modeling context with the retrofitting method of [17]. We first learn sentence vectors using the content model only (i.e., by turning off contextual components in Eq. 1). Then we retrofit these vectors with the graph Laplacian $\mathcal{L}_r(\mathbf{v}_i, \mathcal{N}(\mathbf{v}_i))$ to encourage the revised

[8] https://code.google.com/archive/p/word2vec/.
[9] http://clic.cimec.unitn.it/composes/cphrase-vectors.html.
[10] https://github.com/ryankiros/skip-thoughts.
[11] http://yknzhu.wixsite.com/mbweb.

vectors to be similar to the vectors of neighboring sentences and also similar to their prior representations. We consider two types of graph contexts: discourse (Ret-dis) and similarity (Ret-sim).

(IV) Regularized Models: We compare with a variant of our model, where the loss to capture distributional similarity $\mathcal{L}_g(\mathbf{v}_i, \mathbf{v}_j)$ is turned off. This model considers the same information as the retrofitting model (i.e., content and proximity), but trains the vectors in a single step. Its comparison with our complete model will tell us how much distributional similarity contributes to the overall performance. We define regularizers on two types of contexts: discourse (Reg-dis) and similarity (Reg-sim).

(V) Our Models: We experiment with two variants of our combined model, Con-S2V: one with discourse context (Con-S2V-dis), and the other with similarity context (Con-S2V-sim).

Table 3. Optimal values of the hyper-parameters for different models on different tasks.

Dataset	Task	Sen2Vec	FastSent	W2V-avg	Reg-sim	Reg-dis	Con-S2V-sim	Con-S2V-dis
		(win. size)			*(win. size, reg. str.)*		*(win. size, reg. str.)*	
Reuters	Clas.	8	10	10	(8, 1.0)	(8, 1.0)	(8, 0.8)	(8, 1.0)
	Clus.	12	8	12	(12, 0.3)	(12, 1.0)	(12,0.8)	(12, 0.8)
Newsgroups	Clas.	10	8	10	(10, 1.0)	(10, 1.0)	(10, 1.0)	(10, 1.0)
	Clus.	12	12	12	(12, 1.0)	(12, 1.0)	(12, 0.8)	(10, 1.0)
DUC 2001	Sum.	10	12	12	(10, 0.8)	(10, 0.5)	(10, 0.3)	(10, 0.3)
DUC 2002	Sum.	8	8	10	(8, 0.8)	(8, 0.3)	(8, 0.3)	(8, 0.3)

5.2 Model Settings

The representation dimensions were set to 300 in DM and DBOW models. The concatenation of the two vectors yields 600 dimensions for Sen2Vec. For a fair comparison, the dimensions in all other models that we train (except pre-trained C-Phrase and Skip-Thought) were fixed to 600. All the prediction-based models were trained with SGD. Retrofitting was done using iterative method [17] with 20 iterations. The number of noise samples (S) in negative sampling was set to 5. We also used subsampling of frequent words [16], which together with negative sampling give significant speed-ups in training.

For each dataset described in Sect. 4, we randomly selected 20% documents from the training set to form a held-out validation set on which we tune the hyper-parameters. *Window size* (k) is a hyper-parameter that is common to all models. The regularized models have an additional hyper-parameter, *regularization strength*(λ). We tuned with $k \in \{8, 10, 12\}$ and $\lambda \in \{0.3, 0.6, 0.8, 1\}$, and we optimized F_1 for classification, AMI for clustering, and ROUGE-1 for summarization. Table 3 shows the hyper-parameters and their optimal values for each task. We evaluated our models on the test sets with these optimal values. We ran each experiment five times and take the average of the evaluation measures to avoid any randomness in results.

5.3 Classification and Clustering Results

Table 4 shows the results of the models on topic classification and clustering tasks, respectively. The scores are shown in comparison to Sen2Vec.

Table 4. Performance of our models on topic classification and clustering tasks in comparison to Sen2Vec.

	Topic classification results						Topic clustering results			
	Reuters			Newsgroups			Reuters		Newsgroups	
	F_1	Acc	κ	F_1	Acc	κ	V	AMI	V	AMI
Sen2Vec	83.25	83.91	79.37	79.38	79.47	76.16	42.74	40.00	35.30	34.74
W2V-avg	(+) 2.06	(+) 1.91	(+) 2.51	(−) 0.42	(−) 0.44	(−) 0.50	(−) 11.96	(−) 10.18	(−) 17.90	(−) 18.50
C-Phrase	(−) 2.33	(−) 2.01	(−) 2.78	(−) 2.49	(−) 2.38	(−) 2.86	(−) 11.94	(−) 10.80	(−) 1.70	(−) 1.44
FastSent	(−) 0.37	(−) 0.29	(−) 0.41	(−) 12.23	(−) 12.17	(−) 14.21	(−) 15.54	(−) 13.06	(−) 34.40	(−) 34.16
Skip-Thought	(−) 19.13	(−) 15.61	(−) 21.8	(−) 13.79	(−) 13.47	(−)15.76	(−) 29.94	(−) 28.00	(−) 27.50	(−) 27.04
Tf-Idf	(−) 3.51	(−) 2.68	(−) 3.85	(−) 9.95	(−) 9.72	(−) 11.55	(−) 21.34	(−) 20.14	(−) 29.20	(−) 30.60
Ret-sim	(+) 0.92	(+) 1.28	(+) 1.65	(+) 2.00	(+) 1.97	(+) 2.27	(+) 3.72	(+) 3.34	(+) 5.22	(+) 5.70
Ret-dis	(+) 1.66	(+) 1.79	(+) 2.30	(+) 5.00	(+) 4.91	(+) 5.71	(+) 4.56	(+) 4.12	(+) 6.28	(+) 6.76
Reg-sim	(+) 2.53	(+) 2.53	(+) 3.28	(+) 3.31	(+) 3.29	(+) 3.81	(+) 4.76	(+) 4.40	(+) 12.78	(+) 12.18
Reg-dis	(+) 2.52	(+) 2.43	(+) 3.17	(+) 5.41	(+) 5.34	(+) 6.20	(+) 7.40	(+) 6.82	(+) 12.54	(+) 12.44
Con-S2V-sim	(+) 3.83	(+) 3.55	(+) 4.62	(+) 4.52	(+) 4.50	(+) 5.21	(+) **14.98**	(+) **14.38**	(+) 13.68	(+) 13.56
Con-S2V-dis	(+) **4.29**	(+) **4.04**	(+) **5.22**	(+) **7.68**	(+) **7.56**	(+) **8.80**	(+) 9.30	(+) 8.36	(+) **15.10**	(+) **15.20**

Unsurprisingly, Sen2Vec outperforms Tf-Idf representation (row 6) by a good margin on both tasks. It gets improvements of up to 11.6 points on classification, and up to 30.6 points on clustering. This is inline with the finding of [1], and demonstrates the benefits of using distributed representation over sparse BOW representations.

Simple averaging of Word2Vec vectors performs quite well for classification, especially, on Reuters, where it outperforms Sen2Vec by 1.9 to 2.5 points. [8] also reported similar findings on five out of six datasets. However, averaging does not perform well on clustering, where the scores are 10.2 to 18.5 points below than Sen2Vec.

Simple addition-based composition of C-Phrase word vectors performs poorly on both tasks – lower than Sen2Vec by 2 to 3 points on classification and by 1.4 to 11.9 points on clustering.

Unexpectedly, FastSent and Skip-Thought perform quite poorly on both tasks. Skip-Thought, in particular, has the worst performance on both tasks. These results contradict the claim made by [7] that skip-thought vectors are generic. To investigate if the poor results are due to shift of domains (*book* vs. *news*), we also trained Skip-Thought on our training corpora with vector size 600 and vocabulary size $30K$. The performance was even worse. We hypothesize, this is due to our training set size, which may not be enough for the heavy model. Also, Skip-Thought does not perform any inference to extract the vector using a context – although the model was trained to generate neighboring sentences, context was ignored when the encoder was used to extract the sentence vector.

Regarding FastSent, although its classification performance on Reuters is comparable to Sen2Vec, it performs poorly on Newsgroups, where the measures are 12.2 to 14.3 points lower than Sen2Vec. The differences get bigger in clustering. The reason could be that FastSent does not learn sentence representations directly, rather it simply adds the word vectors. Note that FastSent was outperformed by Tf-Idf in all classification tasks in [8]. Since both Skip-Thought and FastSent learn representations by predicting contents of adjacent sentences, the learned vectors might capture linguistic properties that are more specific to the neighbors.

We also experimented with SAE and SDAE auto-encoders proposed in [8]. However, they performed poorly on our tasks (thus not shown in the table). For example, SAE gave accuracies of around 40% on reuters and 18% on newsgroups. This is similar to what [8] observed. They propose to use pretrained word embeddings to improve the results. We did not achieve significant gains by using pretrained embeddings on our tasks.

Interestingly, the retrofitting and regularized models improve over Sen2Vec on both tasks, showing gains of up to 6.2 points on classification and up to 12.8 points on clustering. The improvements in most cases are significant. This demonstrates that proximity hypothesis is beneficial for these tasks.

When we compare regularized models with retrofitted ones, we observe that regularized models consistently outperform the retrofitted counterparts on both tasks with improvements of up to 1.6 points on classification and up to 7.6 points on clustering. This demonstrates that incorporating contextual information by means of regularization is more effective than retrofitting. This could be due to the fact that regularization approach induces contextual information while learning the vectors from scratch as opposed to revising them in a post-processing step.

Finally, we observe further improvements for our complete models (Con-S2V variants) on both tasks. Compared to the best regularized models, our models deliver improvements of up to 2.6 points on classification and up to 7.6 points on clustering. This demonstrates that by including the neighbor prediction component to model distributional similarity, our model captures complementary contextual information to what is captured by the regularized models. A comparison between the context types reveals that discourse context is more beneficial than similarity context in most cases, especially for classification. For clustering, similarity context gives better results in a few cases (e.g., on Reuters). Overall, our best model outperforms the best existing model by up to 8.8 and 15.20 points on classification and clustering tasks, respectively.

5.4 Summarization Results

Table 5 shows ROUGE-1 scores of our models on DUC datasets for the summary length of 100 words. W2V-avg performs well achieving comparable score to Sen2Vec on DUC'01 and 1.4 points improvement on DUC'02. C-Phrase outperforms Sen2Vec by 2.5 and 1.7 points on DUC'01 and DUC'02, respectively. FastSent and Skip-Thought again perform disappointingly. Sen2Vec outperforms FastSent by 4.15 and 7.53 points on DUC'01 and DUC'02, respectively. Skip-Thought performs comparably to Sen2Vec on DUC'01, but gets worse on DUC'02.

Interestingly, Tf-Idf performs quite well on this task. It gives the top score on DUC'01 (i.e., 48.7 ROUGE-1), and an improvement of 1.5 points over Sen2Vec on DUC'02. These results suggest that existing distributed representation methods are inferior to traditional methods in modeling aspects that are necessary for measuring sentence importance.

Retrofitted models give mixed results and fail to get significant improvement over Sen2Vec. On the other hand, with similarity context, regularized model improves over Sen2Vec by 2 to 3 points. This again suggests that regularization is a better method to incorporate context proximity. By including the neighbor prediction component to incorporate distributional similarity, our combined model improves the scores further; it achieves the second best result on DUC'01, and becomes top-performer on DUC'02. It is not surprising that similarity context is more suitable than discourse context for this task. From a context of topically similar sentences, our model learns representations that capture linguistic aspects related to information centrality. Given that the existing models fail to beat the Tf-Idf baseline on this task, our results are rather encouraging.

Table 5. ROUGE-1 scores of the models on DUC datasets in comparison with Sen2Vec.

	DUC'01	DUC'02
Sen2Vec	43.88	54.01
W2V-avg	(−) 0.62	(+) 1.44
C-Phrase	(+) 2.52	(+) 1.68
FastSent	(−) 4.15	(−) 7.53
Skip-Thought	(+) 0.88	(−) 2.65
Tf-Idf	(+) **4.83**	(+) 1.51
Ret-sim	(−) 0.62	(+) 0.42
Ret-dis	(+) 0.45	(−) 0.37
Reg-sim	(+) 2.90	(+) 2.02
Reg-dis	(−) 1.92	(−) 8.77
Con-S2V-sim	(+) 3.16	(+) **2.71**
Con-S2V-dis	(+) 1.15	(−) 4.46

6 Discussion and Future Directions

We have presented a novel model to learn distributed representation of sentences by considering content as well as context of a sentence. Our results on tasks involving classifying, clustering and ranking sentences confirm that extra-sentential contextual information is crucial for modeling sentences, and this information is best captured by our model that comprises a neighbor-based prediction component and a regularization component to capture distributional similarity and contextual proximity, respectively.

One important property of our model is that it encodes a sentence directly, and it considers neighboring sentences as atomic units. Apart from the improvements that we achieve in various tasks, this property makes our model quite efficient to train compared to *compositional* methods like encoder-decoder models (e.g., SDAE, Skip-Thought) that compose a sentence vector from the word vectors. Encoder-decoder approaches attempt to capture the structure of a sentence, which could be beneficial to model long distance relations between words (e.g., negation in sentiment classification). It would be interesting to see how our model compares with compositional models on sentiment classification task. However, this would require creating a new dataset of comments with sentence-level sentiment annotations. We intend to create such datasets and evaluate the models in the future.

Acknowledgments. This research is partially supported by Mohammad Hasan's NSF CAREER Award (IIS-1149851).

References

1. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, vol. 14, pp. 1188–1196 (2014)
2. Mikolov, T., Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, Georgia, pp. 746–751. Association for Computational Linguistics, June 2013
3. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
4. Socher, R., Lin, C.C.Y., Ng, A.Y., Manning, C.D.: Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 129–136 (2011)
5. Stede, M.: Discourse Processing. Morgan & Claypool Publishers, San Rafael (2011)
6. Hobbs, J.R.: Coherence and coreference. Cogn. Sci. **3**(1), 67–90 (1979)
7. Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Torralba, A., Urtasun, R., Fidler, S.: Skip-thought vectors. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, Montreal, Canada, pp. 3294–3302. MIT Press (2015)
8. Hill, F., Cho, K., Korhonen, A.: Learning distributed representations of sentences from unlabelled data. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, pp. 1367–1377. Association for Computational Linguistics, June 2016
9. Harris, Z.: Distributional structure. Word **10**, 146–162 (1954)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
11. Pham, N.T., Kruszewski, G., Lazaridou, A., Baroni, M.: Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In: ACL 2015, 26–31 July 2015, Beijing, China, vol. 1: Long Papers, pp. 971–981 (2015)
12. Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cogn. Sci. **34**(8), 1388–1439 (2010)
13. Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics, October 2014

14. Morin, F., Bengio, Y.: Hierarchical probabilistic neural network language model. In: Cowell, R.G., Ghahramani, Z. (eds.) Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Society for Artificial Intelligence and Statistics, pp. 246–252 (2005)
15. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation: a new estimation principle for unnormalized statistical models. In: Teh, Y., Titterington, M. (eds.) Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS). JMLR W&CP, vol. 9, pp. 297–304 (2010)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS 2013, pp. 3111–3119 (2013)
17. Faruqui, M., Dodge, J., Jauhar, S.K., Dyer, C., Hovy, E., Smith, N.A.: Retrofitting word vectors to semantic lexicons. In: Proceedings of NAACL (2015)
18. Halliday, M., Hasan, R.: Cohesion in English. Longman, London (1976)
19. Malioutov, I., Barzilay, R.: Minimum cut model for spoken lecture segmentation. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, ACL-44, Sydney, Australia, pp. 25–32. Association for Computational Linguistics (2006)
20. Erkan, G., Radev, D.R.: LexRank: graph-based lexical centrality as salience in text summarization. J. Artif. Int. Res. **22**(1), 457–479 (2004)
21. Wolf, F., Gibson, E.: Representing discourse coherence: a corpus-based study. Comput. Linguist. **31**, 249–288 (2005)
22. Nenkova, A., McKeown, K.: Automatic summarization. Found. Trends Inf. Retrieval **5**(23), 103–233 (2011)
23. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web (1999)
24. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, Barcelona, Spain, vol. 8 (2004)
25. Hermann, K.M., Blunsom, P.: Multilingual models for compositional distributed semantics. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, vol. 1: Long Papers, pp. 58–68. Association for Computational Linguistics, June 2014
26. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035 (2007)
27. Rosenberg, A., Hirschberg, J.: V-measure: a conditional entropy-based external cluster evaluation measure. In: EMNLP-CoNLL, vol. 7, pp. 410–420 (2007)
28. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J. Mach. Learn. Res. **11**, 2837–2854 (2010)
29. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Stroudsburg, PA, pp. 1631–1642. Association for Computational Linguistics, October 2013