Unsupervised Sentence Embedding Using Document Structure-based Context

Taesung Lee ✉ and Youngja Park

IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
Taesung.Lee@ibm.com, young_park@us.ibm.com

Abstract. We present a new unsupervised method for learning general-purpose sentence embeddings. Unlike existing methods which rely on local contexts, such as words inside the sentence or immediately neighboring sentences, our method selects, for each target sentence, influential sentences from the entire document based on the document structure. We identify a dependency structure of sentences using metadata and text styles. Additionally, we propose an out-of-vocabulary word handling technique for the neural network outputs to model many domain-specific terms which were mostly discarded by existing sentence embedding training methods. We empirically show that the model relies on the proposed dependencies more than the sequential dependency in many cases. We also validate our model on several NLP tasks showing 23% F1-score improvement in coreference resolution in a technical domain and 5% accuracy increase in paraphrase detection compared to baselines.

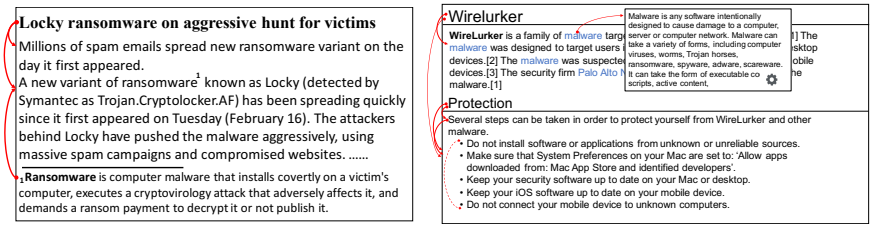
Keywords: sentence embedding · document structure · out-of-vocabulary

1 Introduction

Distributed representations of words and sentences are ever more leveraged to understand text [15, 16, 11, 19, 8, 2, 23]. These methods embed a word or sentence by training a neural network to predict the next word or sentence without supervision. However, unlike human reading with broader context and structure in mind, the existing approaches focus on a small continuous context of neighboring sentences. These approaches work well on continuous but less structured text like movie transcripts, but do not work well on structured documents like encylopedic articles and technical reports.

To better support semantic understanding of such technical documents, we propose a new sentence embedding framework to learn general-purpose sentence representations by leveraging long-distance dependencies between sentences in a document. We observe that understanding a sentence often requires understanding of more comprehensive context as well as the immediate context, including the document title, previous paragraphs, or even related articles as shown in Figure 1. For instance, all the sentences in the document can be related to the document title (Figure 1(a)). The items in a list structure can be influenced by the sentence introducing the list, and, HTML documents can contain hyperlinks

to provide more information on certain terms (Figure 1(b)). Using these document structure-based contexts, we can connect 'ransomware' with 'payment' (Figure 1(a)).



(a) Document Title and Footnote (b) Section, List and Hyperlink

Fig. 1. Examples of long distance dependencies between sentences

Our approach, leveraging such structural elements, has several advantages. First, to provide enough context to understand a sentence, instead of using a global context of all sentences in the document, we leverage a concise set of context sentences to be considered using the structural dependencies. A larger context can produce more accurate representations of sentences. However, it is infeasible to train neural network models with a large number of context sentences. Second, we further prioritize the selected sentences based on their semantics and the dependency types. In this way, our model can better handle documents containing several subtopics that may cause sudden local context changes. Some sentences have dependencies to distant ones when a different perspective of the topic is introduced. Using only small neighboring sentences results in insufficient input to the neural network to understand such a sudden change. Using long distance dependencies, we can provide a broader context.

Additionally, we can leverage the structural information to better handle out-of-vocabulary (OOV) words. The vocabulary in a neural network is always limited due to costly training time and memory use. Existing methods discard low frequency words and map all OOV words to one or a few variables. This method can loose important keywords in a technical domain that continuously creates new terms. We introduce more fine-grained OOV variables using information extracted from the structural context.

We empirically show that the model actually learns to rely more on some of the dependencies. We validate our model on several NLP tasks using a Wikipedia corpus which shows that our model consistently outperforms the existing methods. Our model produces much lower loss for the target sentence prediction task and 5% increase in accuracy for paraphrase identification than Skip-Thought. The results confirm that training with only local context does not work well for such documents. We also compare the performance of the learned embedding for coreference resolution. For coreference resolution, our model shows 23%

Table 1. Categorization of sentence embedding methods. * denotes methods not requiring labeled data.

Range	Continuity	
	Continuous	Discontinuous
Intra-sentence	[9, 6, 7, 24, 2, 18]; [10]*	[22, 13, 23]
Inter-sentence	[8]*	Our work*

improvement in F1 over DeepCoref [1], a state-of-the-art deep learning-based approach.

The main contributions of the paper include:

- A general-purpose sentence embedding method which leverages long distance sentence dependencies extracted from the document structure.
- A rule-based dependency annotator to automatically determine the document structure and extract all governing sentences for each sentence.
- A new OOV handling technique based on the document structure.

2 Related work

Distributed representation of sentences, or sentence embedding, has gained much attention recently, as word-level representations [15, 16, 11, 19] are not sufficient for many sentence-level or document-level tasks, such as machine translation, sentiment analysis and coreference resolution. Recent approaches using neural networks consider some form of dependencies to train the network. Dependencies can be continuous (relating two adjacent words or sentences) or discontinuous (relating two distant words or sentences), and intra-sentence (dependency of words within a sentence) or inter-sentence (dependency between sentences). Many sentence embedding approaches leverage these dependencies of words to combine word embeddings as shown in Table 1.

One direct extension of word embedding to sentences is combining words vectors in a continuous context window. [9] uses a weighted average of the constituent word vectors. [24], [2], and [18] use supervised approaches to train a LSTM network that merges word vectors. [6] and [7] use convolutional neural networks (CNN) over continuous context window to generate sentence representations. [10] includes a paragraph vector in the bag of word vectors and apply a word embedding approaches [15, 16].

Recently, several researchers have proposed dependency-based embedding methods using a dependency parser to consider discontinuous intra-sentence relationships [22, 13, 23]. [22] uses recursive neural network to consider discontinuous dependencies. [13] proposes a dependency-based CNN which concatenate a word with its ancestors and siblings based on the dependency tree structure. [23] proposes tree structured LSTM networks. These studies show that dependency-based (discontinuous) networks outperform their sequential (continuous) counterparts.

Unlike these approaches considering only intra-sentence dependencies, Kiros et al., 2015 propose a new architecture SKIP-THOUGHT [8] joining two recurrent neural networks, encoder and decoder. The encoder combines the words in a sentence into a sentence vector, and the decoder generates the words in the next sentence. Our approach is similar to SKIP-THOUGHT since both approaches are unsupervised and use inter-sentential dependencies. However, SKIP-THOUGHT considers only continuous dependency.

Unlike our approach considering OOV in the output using the document structure, there are approaches that build an embedding of an OOV word on the fly that can be used as input to our system [20], [21], and [5]. Our OOV handling focuses more on mechanism to produce OOV words as the output of the network and leverage them in training, which is out of the scope of these previous papers. [12] proposes a word position-based approach to address the OOV problem for neural machine translation (NMT) systems. Their methods allow a neural machine translation (NMT) system to emit, for each unknown word in the target sentence, the position of the corresponding word in the source sentence. However, their methods are not applicable to sentence embedding, as they rely on an aligned corpus. Also, our approach considers not only word positions but also the dependency types to define OOV words.

3 Document Structured-based Context

Previous sentence embedding methods use intra-sentence dependencies such as a dependency tree, or immediately neighboring sentences for sentence embedding. However, we identify more semantically related content to define sentence dependencies based on the document structure as shown in Figure 1. In this section, we describe a range of such inter-sentence dependencies that can be utilized for sentence embedding and the techniques to automatically identify them.

We use the following notations to describe the extraction of document structure-based context for a given sentence. Suppose we have a document $D = \{S_1, \ldots, S_{|D|}\}$, which is a sequence of sentences. Each sentence S_i is a sequence of words, represented as $s_{i,1}, \ldots, s_{i,|S_i|}$. For each *target sentence* $S_t \in D$, S_t depends on a subset $G \subset D$[1]. We call such a sentence G_i in G a *governing sentence* of S_t, and say G_i governs S_t, or S_t depends on G_i, defined by one of the dependency types in $\mathcal{D}$ described below.

3.1 Titles

The title of a document, especially a technical document, contains the topic entity, the key claim, and/or the summary of the document, and all other sentences describe and elaborate the title. For instance, the title of the document (*e.g.*, WannaCry) can clarify the meaning of a definite noun phrase (*e.g.*, the ransomware) in the sentence. Section titles play a similar role, but, mostly to

[1] For simplicity, we use G to denote a S_t specific set.

the sentences within the section. We detect different levels of titles, starting from the document title to chapter, section and subsection titles. Then, we identify the region in the document which each title governs and incorporate the title in the embedding of all sentences in the region. To identify titles in a document, we use the various information from the metadata and the document content as follows.

Document Metadata ($\mathcal{D}_{TM}$): We extract a document title from the <title> tag in a HTML document or from the title field in *Word* or PDF document metadata. Since the document title influences all sentences in a document, we consider this title governs every sentence in D.

Heading Tag ($\mathcal{D}_{THn}$): The heading tags <h1> to <h6> in HTML documents are often used to show document or section titles. We consider all sentences between a heading tag and the next occurrence of the same level tag are considered under the influence of the title.

Table Of Contents ($\mathcal{D}_{TC}$): Many documents contain a table of contents (TOC) providing the overall structure of the document. To detect the titles based on the table of contents, we first recognize a phrase indicating TOC, such as "table of contents", "contents" or "index". Then, we parse the content following the cue phrase and check if it contains a typical TOC pattern such as "Chapter 1 – Introduction" or "Introduction ····· 8". The range of each section can be easily identified from the TOC. If the document is an HTML file, each line in the TOC tends to have a hyperlink to the section. For non-HTML documents, we can extract the page number from the TOC (*e.g.*, page 8) and locate the corresponding pages.

Header and Footer ($\mathcal{D}_{TR}$): Technical documents often contain the document or section titles in the headers or footers. Thus, if the same text is repeated in the headers or footers in many pages, we take the text as a title and consider all sentences appearing in these pages belong to the title.

Text Styles ($\mathcal{D}_{TS}$): Titles often have a distinctive text style. They tend to have no period at the end and use a larger font size, a higher number of *italic* or **bold** text, and a higher ratio of capitalized words compared to non-title sentences. We first build a text style model for sentences in the document body, capturing the three style attributes. If a sentence ends without a period and any dimension of its style model has higher value than that of the text style model, we consider the sentence as a title. Then, we split the document based on the detected titles and treat each slice as a section.

3.2 Lists

Authors often employ a list structure to describe several elements of a subject. This list structure typically has an introductory sentence stating the main concept followed by a bulleted, numbered or in-text list of items supporting the main concept as illustrated in Figure 1(b). An item in the list is conceptually more related to the introductory sentence than the other items in the list, but the distance can be long because of other items. We use the following methods

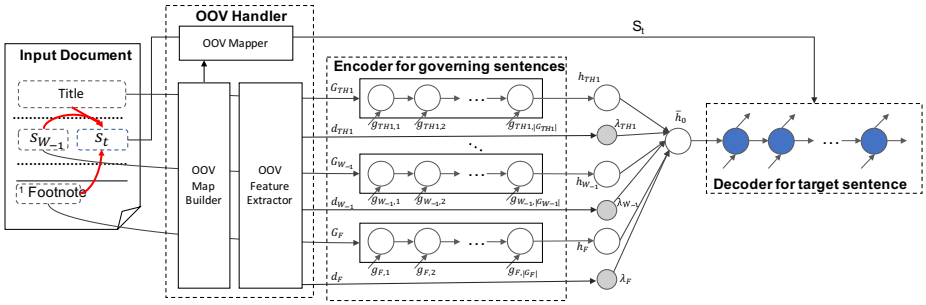


Fig. 2. Our model architecture.

to identify list items, consider the sentence appearing prior to the list items as the introductory sentence and assume that it governs all items in the list.

Formatted List ($\mathcal{D}_{LF}$): To extract numbered or bulleted lists, we use the list tags (*e.g.*, <ul>, <ol>, <li>) for HTML documents. For non-HTML documents, we detect a number sequence (*i.e.*, 1, 2, ...) or bullet symbols (*e.g.*, -, ·) repeating in multiple lines.

In-text List ($\mathcal{D}_{LT}$): We also identify in-text lists such as "First(ly), …. Second(ly), …. Last(ly), …" by identifying these cue words.

3.3 Links

Hyperlinks ($\mathcal{D}_H$): Some sentences contain hyperlinks or references to provide additional information or clarify the meaning of the sentence. We can enrich the representation of the sentence using the linked document. In this work, we use the title of the linked document to govern the target sentence. Alternatively, we can use the embedding of the linked document.

Footnotes and In-document Links ($\mathcal{D}_F$): Footnotes also provide additional information for the target sentence. In an HTML document, such information is usually expressed with in-document hyperlinks, which ends with "#dest". In this case, we identify a sentence marked with "#dest" and add a dependency between the two sentences.

3.4 Window-based Context ($\mathcal{D}_{Wn}$):

We also consider the traditional sequential dependency used in previous methods [8, 4]. Given a document $D = \{S_1, \ldots, S_{|D|}\}$, the target sentence S_t is considered to be governed by n sentences prior to ($n < 0$) or following ($n > 0$) S_t. In our implementation, we use only one prior sentence ($\mathcal{D}_{W_{-1}}$).

4 Neural Network Models

In this section, we describe our model architecture (Figure 2) in detail. Based on the dependencies extracted in Section 3, we build a sentence embedding model.

Similarly to Skip-Thought [8], we train our model to generate a target sentence S_t using a set of governing sentences G. However, Skip-Thought takes into account only the window-based context ($\mathcal{D}_{Wn}$), while our model considers diverse long distance context and their dependency types as described in Section 4.1. Additionally, unlike existing sentence embedding methods, which include only a small fraction of words (typically high frequency words) in the vocabulary and map all other words to one OOV word, we introduce a new OOV handler in our model in Section 4.2.

4.1 Inter-Sentential Dependency-based Encoder-Decoder

Our model has several encoders (one encoder for each governing sentence $G_i \in G$), a decoder and an OOV handler as shown in Figure 2. The input to each cell is a word, represented as a dense vector. We use the pre-trained vectors from the CBOW model [16], and the word vectors can be optionally updated during training.

We now formally describe the model given a target sentence S_t and a set G of its governing sentences. We first describe the encoders that digest each $G_i \in G$. Given the i-th governing sentence $G_i = (g_{i,1}, \ldots, g_{i,|G_i|})$, let $w(g_{i,t})$ be the word representation (pre-trained or randomly initialized) of word $g_{i,t}$. Then, the following equations define the encoder for G_i.

$$\begin{aligned} h_{i,t} &= \mathrm{RC}(w(g_{i,t}), h_{i,t-1}; \theta_E), \\ \lambda_i &= \mathsf{sigmoid}(\mathcal{U} d_i + g), \qquad h_i = h_{i,|G_i|}, \\ \bar{h}_0 &= \sum_i W_{\mathrm{dep}(i)} \left\{ \lambda_i (u_{\mathrm{dep}(i)} h_i + a_{\mathrm{dep}(i)}) \right. \\ &\qquad \left. + (1 - \lambda_i) h_i + b \right\} \end{aligned} \tag{1}$$

where RC is a recurrent neural network cell (e.g., LSTM or GRU) that updates the memory $h_{i,t}$; θ_E is the parameters for the encoder RC; λ_i is an OOV weight that decides how much we rely on out-of-vocabulary words; d_i denotes the OOV features for G_i; $\mathcal{U}$ and g are linear regression parameters; $\mathsf{sigmoid}(\cdot)$ is the sigmoid function; u_{dep} and a_{dep} are an OOV weight transformation; W and b are a transformation matrix and a bias; and $\bar{h}_0$ is the aggregated information of G and is passed to the decoder for target sentence generation.

Now, we define the decoder as follows:

$$\begin{aligned} o_t, \bar{h}_t &= \mathrm{RC}(o_{t-1}, \bar{h}_{t-1}; \theta_D), \\ y_t &= \mathsf{softmax}(V o_t + c) \end{aligned} \tag{2}$$

where RC is a recurrent neural network cell that updates the memory $\bar{h}_t$ and generates the output o_t; θ_D is a set of parameters for the decoder RC; $\mathsf{softmax}(\cdot)$ is the softmax function; and $V o_t + c$ transforms the output into the vocabulary space. That is, $V o_t + c$ generates logits for words in the vocabulary set and is used to predict the words in the target sentence.

To strike a balance between the model accuracy and the training time, we use K randomly chosen governing sentences from G for all target sentence. We use the cross entropy between y_t and o_t as the optimization function and update $\theta_E, W_{\text{dep}(i)}, b, V, c, \theta_D$ and optionally $w(\cdot)$.

4.2 Out-of-vocabulary (OOV) mapping

Incorporating all the words from a large text collection in deep learning models is infeasible, since the amount of memory use and training time will be too costly. Especially, in technical domains, new jargons are constantly added, and, their character level information is often not very useful (*e.g.*, WannaCry, 129.42.56.189).

Thus, we propose an OOV word handling method based on the diverse sentence relationships from Section 3. OOV word handling is desired in the following three places: (1) input embeddings to encode the governing sentences (G); (2) input embeddings to decode the target sentence (S_t); and (3) output logits to compute the loss with respect to S_t. For the first two cases, *i.e.*, generating the input embeddings of G and S_t for the encoder and the decoder, we use the average vector of all words in the vocabulary to represent all OOV words.

While there are several approaches to generate input embeddings for OOV words (Case 1 & 2), such as average of all word embeddings, character-based embedding, context-based embedding [20, 21, 5], there has been little work for building a model generating OOV words in the output and use them in the training loss (Case 3). Existing sentence embedding techniques reduce the vocabulary size mainly by using only high frequency words and by collapsing all other words to one special word (e.g., $<unk>$). However, this single OOV symbol for all OOV words treats both very important OOV word (*e.g.*, topic entities, domain-specific words and proper nouns) and other words alike, resulting in unsatisfactory results for technical documents.

Instead of replacing all OOV words by a single variable, we consider the dependency and the position of OOV words to build a set of *OOV variables*. Given the training corpus with the entire vocabulary V_M with size M, we first select N most frequent words in the training corpus as an initial vocabulary V_N (typically, $N \ll M$, *i.e.*, tens of thousands vs. millions or billions). Then, we build an OOV map that reduces the OOV words ($V_M - V_N$) into a smaller vocabulary V_{OOV} of *OOV variables*, $\{O_i(j)\}$, where $O_i(j)$ represents j-th OOV word given a governing sentence G_i (*e.g.*, an OOV variable may indicate the actor in the previous sentence). In particular, we use OOV variables to represent the first and the last η OOV words in sentences with each dependency, observing that many semantically important words tend to appear at the beginning or the end of the governing sentences. We denote the j-th last OOV word by $O_i(-j)$. This idea of encoding OOV words based on their positions in a sentence is similar to the machine translation approach by [12]. However, we encode OOV words using the dependency type of the sentence as well as their position in the sentence.

After we replace OOV words using the OOV mapping, we have the augmented vocabulary $V_N \cup V_{OOV}$ with a manageable size. The optimization goal

of each RNN cell without OOV words is to predict the next word with one correct answer. In contrast, our model allows multiple correct answers, since an OOV word can be mapped to multiple OOV variables. We use the cross entropy with soft labels as the optimization loss function. The weight of each label is determined by the inverse-square law, *i.e.*, the weight is inversely proportional to the square of the number of words associated with the label. This weighting scheme gives a higher weight to less ambiguous dependency.

One additional component we add related to OOV words is a weight function for the governing sentences based on occurrences of proper nouns (λ_i in Equation 1). Instead of equally weighing all governing sentences, we can give a higher weight to sentences with proper nouns, which are more likely to have OOV words, to leverage the contextual information of such OOV words in other sentences to understand the OOV words in the target sentence. Thus, we introduce a feature vector representing the number of OOV proper nouns in the i-th governing sentence (d_i in Equation 1). Currently, the features include the number of OOV words whose initials are uppercased, the number of OOV words that are uppercased, and the number of OOV words with at least one upper-case letter. Together with the linear regression parameters, $\mathcal{U}$ and g, the model learns the weights for different dependency types.

5 Experiments

We empirically evaluate our approach on various NLP tasks and compare the results with other existing methods. We trained the proposed model (OURS) and the baseline systems on 807,647 randomly selected documents from the 2009 Wikipedia dump, which is the last Wikipedia dump released in *HTML* format. Since our approach leverages HTML tags to identify document structures, our model use the raw HTML files. For the baseline systems, we provide plain text version of the same articles. All models were trained for 300K steps with 64-sized batches and the Adagrad optimizer [3]. For the evaluation, we use GRU cells for RC in Equation 2. For each target sentence, if there are more than 8 governing sentences, we randomly choose 8 of them as the context ($K = 8$). We set the maximum number of words in a sentence to be 30 and pad each sentence with special start and end of sentence symbols. We set η to 4, resulting in $|V_{OOV}| = 80$.

5.1 Dependency importance

In this experiment, we show the relative importance of long distance sentence relations compared to sequential relations. Note that W_{dep} in Equation 1 implies the importance level of a dependency *dep*. In Table 2, we show the relative importance of the different dependencies compared to the sequential dependency ($\mathcal{D}_{W_{-1}}$), which is used in other methods. As we can see, all levels of document and section titles, except the fourth level subsection title, play a more significant role than the sequential dependency. The reason that the title from the metadata,

($\mathcal{D}_{TM}$), does not have a high weight as the title from the heading 1 tag ($\mathcal{D}_{TH1}$) is that the metadata contains extra text, "- Wikipedia", in the title for Wikipedia articles (*e.g.*, "George W. Bush - Wikipedia" instead of "George W. Bush"). Further, hyperlinks ($\mathcal{D}_H$), in-document links ($\mathcal{D}_F$) and formatted lists ($\mathcal{D}_{LF}$) are all shown to have a similar influence as the sequence dependency. The remaining dependencies, $\mathcal{D}_{TC}$, $\mathcal{D}_{TR}$, $\mathcal{D}_{TS}$, and $\mathcal{D}_{LT}$ are scarcely found in the Wikipedia corpus, and thus, did not converge or were not updated.

Table 2. Weights $\|W_{dep}\|_2/\|W_{\mathcal{D}_{W,-1}}\|_2$ of dependencies.

$\mathcal{D}_{TH1}$	$\mathcal{D}_{TH2}$	$\mathcal{D}_{TH3}$	$\mathcal{D}_{LF}$	$\mathcal{D}_{TM}$	$\mathcal{D}_F$	$\mathcal{D}_{TH4}$	$\mathcal{D}_{TH5}$	$\mathcal{D}_H$
2.30	2.30	2.30	1.00	1.00	1.00	0.24	1.40	1.00

5.2 Target sentence prediction

Unlike most other approaches, our model and Skip-Thought [8] can learn application-independent sentence representations without task-specific labels. Both models are trained to predict a target sentence given a context. The prediction is a sequence of vectors representing probabilities of words in the target sentence. For a quantitative evaluation between the two models, we compare their prediction losses by using cross entropy loss. We randomly chose 640,000 target sentences for evaluation and computed the average loss over the 640K sentences.

We compare Skip-Thought with two versions of our model. Ours denotes our model using the document structure-based dependencies and the OOV handler. Ours-DEP denotes our model with the OOV handler but using only local context like Skip-Thought to show the impact of the OOV handler. Table 3 shows the comparison of the three models. The values in the table are the average loss per sentence. We measure the average loss value excluding OOV words for Skip-Thought, as it cannot handle OOV words. However, for our models, we measure the loss values with (All Words) and without OOV words (Voc. Words). As we can see, both Ours−DEP and Ours significantly outperform Skip-Thought resulting in 25.8% and 26.9% reduction in the loss values respectively.

5.3 Paraphrase detection

Further, we compare our model with Skip-Thought on a paraphrase detection task using the Microsoft Research Paraphrase corpus [14]. The data set consists of 5,801 sentence pairs extracted from news data and their boolean assessments (if a sentence pair is paraphrase or not), which were determined by three assessors using majority voting. The goal is correctly classifying the boolean assessments, and the accuracy (# correct pairs / # all pairs) is measured. We used 4,076 pairs

Table 3. Comparison of our models and SKIP-THOUGHT for target sentence prediction

Method	All Words	Voc. Words
OURS	0.1456	0.1394
OURS-DEP	0.1467	0.1415
SKIP-THOUGHT	N/A	0.1907

Table 4. Comparison of paraphrase detection accuracy

Method	Accuracy
OURS	0.72
SKIP-THOUGHT	0.67

for training and 1,725 pairs for testing. Since the data sets contain sentence pairs only and no structural context, we evaluate only the effectiveness of the trained encoder. To compare the quality of sentence embeddings by the two models, we use the same logistic regression classifier with features based on embedded sentences as in [8]. Given a pair of sentences S_1 and S_2, the features are the two embeddings of S_1 and S_2, their entry-wise absolute difference, and their entry-wise products. Our model shows a 5% points higher accuracy than SKIP-THOUGHT in paraphrase detection (Table 4), demonstrating the effectiveness of our encoder trained with the structural dependencies. Note that SKIP-THOUGHT trained with Wikipedia corpus performs worse than a model trained on books or movie scripts due to more complex and less sequential structure in Wikipedia documents.

5.4 Coreference resolution

While our system is not designed for coreference resolution, the rich sentence embedding can be used for unsupervised coreference resolution, unlike the methods relying on annotated corpus [1]. Although building a dedicated coreference resolution method for a given domain can produce better results, we claim that our embedding approach can extract a good starting set of features. We first detect entity mentions, and, for a pronoun or a generic entity mentions (*e.g.*, a definite noun phrase), we select a list of candidate referents that conform to the mention type of the entity reference. Then, we replace the entity reference with each of the candidate referents and compute the loss of the new sentence. Finally, we choose the referent with the lowest loss value as the result, if the loss is less than the original sentence loss value. We extend our model to use sequential dependencies of $\mathcal{D}_{W_{-3}}, \ldots, \mathcal{D}_{W_1}$ (Section 3.4), and further train it with a 700K unlabeled cybersecurity corpus collected from security blogs and websites.

We compare our approach with the Stanford Deep Coreference Resolution tool (DEEPCOREF) [1] on a set of cybersecurity-related documents. The evaluation data consists of 628 entity coreferences extracted from 38 Wikipedia articles about malware programs which were not included in the training document set.

Table 5. Overall performance on coreference resolution

Method	Prec.	Recall	F1
OURS+*SER*	0.77	0.20	0.32
DEEPCOREF+*NER*	0.13	0.10	0.11
DEEPCOREF+*SER*	0.66	0.05	0.09

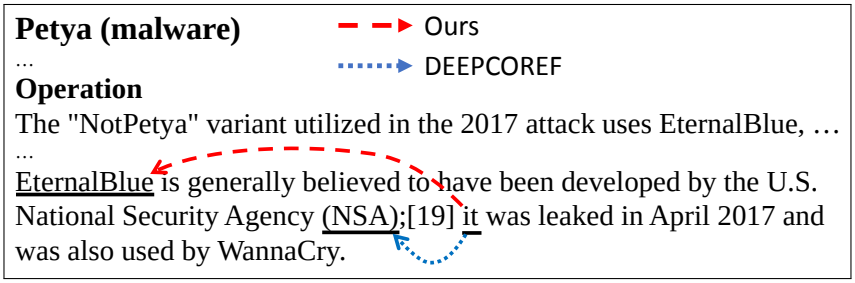


Fig. 3. Example coreference resolution

We conducted experiments for several cybersecurity related entity types such as 'Malware' and 'Vulnerability' and general entity types such as 'Person' and 'Organization'.

Since DEEPCOREF was designed mostly for general entity types and may not be able to identify security entity types, we apply the system both with its own named entity recognizer as the candidate generator (DEEPCOREF+*NER*) and with our candidate generator designed for security entities (DEEPCOREF+*SER*). Table 5 shows MUC precision, recall, and F1-score [17]. Our model achieves higher precision and recall than both versions of DEEPCOREF. Note that DEEPCOREF+*NER* produces very low precision compared to the other models. While DEEPCOREF+*SER* shows higher precision, it still performs worse than OURS+*SER* due to the lack of features for security terms. Figure 4 shows the performance for different entity types. As we can see, while DEEPCOREF+*SER* shows higher F1 score than DEEPCOREF+*NER* for the security entity types, it still shows lower F1 score than OURS+*SER* due to semantics unseen during the training. That is, for person and organization, syntactic features used by DEEPCOREF are important. However, when there is no such features available (*i.e.*, malware and vulnerability), the semantic relationship among sentences is more important. Figure 3 shows an example case where DEEPCOREF identifies the closer candidate as coreferent rather than examining semantics.

6 Conclusion and Future Work

In this paper, we presented a novel sentence embedding technique exploiting diverse types of structural contexts and domain-specific OOV words. Our method

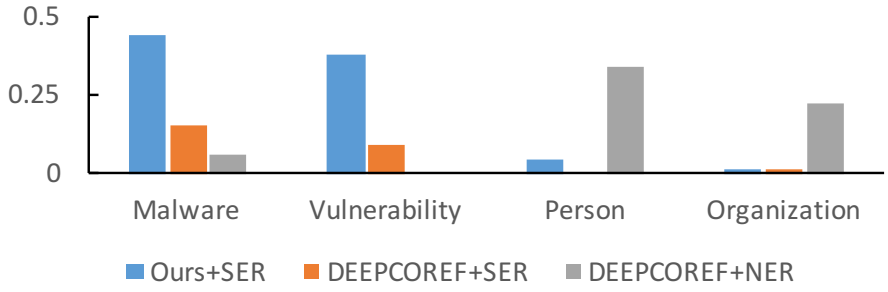


Fig. 4. F1-score per entity types

is unsupervised and application-independent, and it can be applied to various NLP applications. We evaluated the method on several NLP tasks including coreference resolution, paraphrase detection and sentence prediction. The results show that our model consistently outperforms the existing approaches confirming that considering the structural context generates better quality sentence representations.

There are a few possible directions of future work. The proposed approach relies on rule-based dependency annotation. Devising a supervised dependency annotator can be an interesting direction to adapt to other domains with slightly different rules or document format (*e.g.*, XLS). There are also unsupervised neural dependency parsers for intra-sentence dependencies. Studying an inter-sentence counterpart would be very useful for our framework. In our implementation, we used only document titles of the hyperlinked documents. But, linking documents to understand a new document and better exploiting related or prerequisite documents can be an important research direction.

References

1. Clark, K., Manning, C.D.: Deep reinforcement learning for mention-ranking coreference models. In: Empirical Methods on Natural Language Processing (EMNLP) (2016)
2. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: The Conference on Empirical Methods on Natural Language Processing (EMNLP) (2017)
3. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**(Jul), 2121–2159 (2011)
4. Gan, Z., Pu, Y., Henao, R., Li, C., He, X., Carin, L.: Learning generic sentence representations using convolutional neural networks. In: Proceedings of the 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP) (2017)

5. Horn, F.: Context encoders as a simple but powerful extension of word2vec. In: Proceedings of the 2nd Workshop on Representation Learning for NLP. pp. 10–14 (2017)
6. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL). pp. 655–665 (2014)
7. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, (EMNLP). pp. 1746–1751 (2014)
8. Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS). pp. 3294–3302 (2015)
9. Kusner, M.J., Sun, Y., Kolkin, N.I., Weinberger, K.Q.: From word embeddings to document distances. In: Proceedings of the 32nd International Conference on Machine Learning (ICML). pp. 957–966 (2015)
10. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, (ICML). pp. 1188–1196 (2014)
11. Levy, O., Goldberg, Y.: Dependency-based word embeddings. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL). pp. 302–308 (2014)
12. Luong, M.T., Sutskever, I., Le, Q.V., Vinyals, O., Zaremba, W.: Addressing the rare word problem in neural machine translation. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP). pp. 11–19 (2015)
13. Ma, M., Huang, L., Xiang, B., Zhou, B.: Dependency-based convolutional neural networks for sentence embedding. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL). pp. 174–179. Association for Computational Linguistics, Beijing, China (July 2015)
14. Microsoft: Microsoft research paraphrase corpus. `https://www.microsoft.com/en-us/download/details.aspx?id=52398` (2016)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. International Conference on Learning Representations (ICLR) (2013)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS). pp. 3111–3119 (2013)
17. Moosavi, N.S., Strube, M.: Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 632–642 (2016)
18. Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R.: Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. Proceedings of IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) **24**(4), 694–707 (2016)
19. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)

20. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 2227–2237. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-1202, http://aclweb.org/anthology/N18-1202
21. Santos, C.D., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14). pp. 1818–1826 (2014)
22. Socher, R., Lin, C.C., Ng, A.Y., Manning, C.D.: Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of the 26th International Conference on Machine Learning (ICML) (2011)
23. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL). pp. 1556–1566 (2015)
24. Wieting, J., Gimpel, K.: Revisiting recurrent networks for paraphrastic sentence embeddings. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. pp. 2078–2088. Association for Computational Linguistics (2017)