UNSUPERVISED SENTENCE EMBEDDING USING DOCUMENT STRUCTURE-BASED CONTEXT

Anonymous authors
Paper under double-blind review

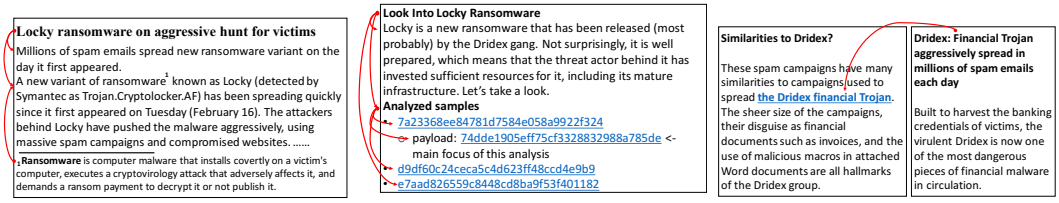
ABSTRACT

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 in the entire document based on a document structure. We identify a dependency structure of sentences using metadata or text styles. Furthermore, we propose a novel out-of-vocabulary word handling technique to model many domain-specific terms, which were mostly discarded by existing sentence embedding methods. We validate our model on several tasks showing 30% precision improvement in coreference resolution in a technical domain, and 7.5% accuracy increase in paraphrase detection compared to baselines.

1 INTRODUCTION

Distributed representations are ever more leveraged to understand text (Mikolov et al., 2013a;b; Levy & Goldberg, 2014; Pennington et al., 2014). Recently, Kiros et al. (2015) proposed a neural network model, SKIP-THOUGHT, that embeds a sentence without supervision by training the network to predict the next sentence for a given sentence. 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 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 unsupervised 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 not only the immediate context but more comprehensive 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 title of the document (1(a)). The first sentence of each item in a list structure can be influenced by the sentence introducing the list (1(b)). Moreover, html documents can contain hyperlinks to provide more information about a certain term (1(c)). With the contexts obtained from document structure, we can connect ransomware with payment (1(a)) and the four hashes with Locky (1(b)).



(a) Document Title and Footnote (b) Sections and List (c) Document Hyperlink

Figure 1: Examples of long distance dependencies between sentences

Our approach leveraging such structural elements has several advantages. First, it can learn from technical documents containing several subtopics that may cause sudden context changes. Some sentences have dependences to distant ones if a different perspective of the topic is introduced. Using

Table 1: Categorization of sentence embedding methods. * denotes unsupervised methods.

		Continuity	
		Continuous	Discontinuous
Range	Intra-sentence	Kusner et al. (2015); Kalchbrenner et al. (2014); Kim (2014); Wieting & Gimpel (2017); Conneau et al. (2017); Palangi et al. (2016); Le & Mikolov (2014)*	Socher et al. (2011); Ma et al. (2015); Tai et al. (2015)
	Inter-sentence	Kiros et al. (2015)*	Our work*

only small neighboring context results in insufficient input to the neural network. Using long distance dependencies, we can provide a broader context. Second, we can consider out-of-vocabulary (OOV) words using their information extracted from the structural context. The vocabulary in a neural network is always limited due to costly training time and memory use. An existing method discarding low frequency words results in losing important keywords in the technical domain.

We validate our model on several NLP tasks using a Wikipedia corpus. When trained with the Wikipedia corpus, our model produces much lower loss than SKIP-THOUGHT in the target sentence prediction task, confirming that training with only local context does not work well for such documents. We also compare the performance of the learned embedding on several NLP tasks including coreference resolution and paraphrase identification. For coreference resolution, our model shows roughly 30% improvement in precision over a state-of-the-art deep learning-based approach on cybersecurity domain, and produces 7.5% increase in accuracy compared with SKIP-THOUGHT for paraphrase identification.

The main contributions of the paper include:

- We propose a general-purpose sentence embedding method which leverages long distance sentence dependencies extracted from the document structure.
- We developed a rule-baed dependency annotator to automatically determine the document structure and extract all governing sentences for each sentence.
- We also present a new OOV handling technique based on the document structure.
- We have applied our methods to several NLP applications using cybersecurity datasets. The experiments show that our model consistently outperform existing methods.

2 RELATED WORK

Distributed representation of sentences, which is often called sentence embedding, has gained much attention recently, as word-level representations (Mikolov et al., 2013a;b; Levy & Goldberg, 2014; Pennington et al., 2014) 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, and can be categorized as shown in 1.

One direct extension of word embedding to sentences is combining words vectors in a continuous context window. Kusner et al. (2015) use a weighted average of the constituent word vectors. Wieting & Gimpel (2017), Conneau et al. (2017), and Palangi et al. (2016) use supervised approaches to train a long short-term memory (LSTM) network that merges word vectors. Kalchbrenner et al. (2014) and Kim (2014) use convolutional neural networks (CNN) over continuous context window to generate sentence representations. Le & Mikolov (2014) include a paragraph vector in the bag of word vectors, and apply a word embedding approaches (Mikolov et al., 2013a;b).

Recently, several researchers have proposed dependency-based embedding methods using a dependency parser to consider discontinuous intra-sentence relationships (Socher et al., 2011; Ma et al., 2015; Tai et al., 2015). Socher et al. (2011) uses recursive neural network to consider discontinuous

dependencies. Ma et al. (2015) proposes a dependency-based convolutional neural network which concatenate a word with its ancestors and siblings based on the dependency tree structure. Tai et al. (2015) proposes tree structured long short-term memory networks. These studies show that dependency-based (discontinuous) networks outperform their sequential (continuous) counterparts.

Unlike these approaches, considering only intra-sentence dependencies, SKIP-THOUGHT (Kiros et al., 2015) joins two recurrent neural networks, encoder and decoder. The encoder combines the words in a sentence into a sentence vector, and the decoder generates the next sentence. Our approach is similar to SKIP-THOUGHTsince both approaches are unsupervised and use inter-sentential dependencies. However, SKIP-THOUGHTconsiders only continuous dependency.

3 DOCUMENT STRUCTURED-BASED CONTEXT

Previous methods use intra-sentence dependencies such as dependency tree, or immediately neighboring sentences for sentence embedding. However, we identify more semantically related content to a target sentence 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: $s_{i,1}, \ldots, s_{i,|S_i|}$. For each *target sentence* $S_t \in D$, there can be a subset $G \subset D$ that S_t depends on (For simplicity, we use G to denote a S_t specific set). We call such a sentence in G a *governing sentence* of S_t, and say G_i governs S_t, or S_t depends on G_i. Each G_i is associated with S_t through one of the dependency types in $\mathcal{D}$ described below.

3.1 TITLES

The title of a document, especially a technical document, contains the gist of the document, and all other sentences support the title in a certain way. For instance, the title of the document can clarify the meaning of a definite noun in the sentence. Section titles play a similar role, but, mostly to 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 the sentences in the region. To identify titles in a document, we use various information from the metadata and the document content.

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 a title obtained from $\mathcal{D}_{TM}$ 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 the 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 $\cdots\cdots\cdots$ 8". The range of each section can be easily identified from the TOC. If the document is a HTML file, each line in the TOC tends to have a hyperlink to the corresponding section. For non-HTML documents, we can extract the page number from the TOC (*e.g.*, page 8) and locate the corresponding content if the document includes the page numbers.

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 header or in the footer in many pages, we take the text as a title and consider all the sentences appearing in these pages belong to the title.

Text Styles ($\mathcal{D}_{TS}$): Titles often have a distinctive text style including a different font style (*e.g.*, *italics* or **bold**), a bigger font size or capitalized words. They also tend to be shorter than typical sentences and have no period at the end. We apply this heuristic to detect titles when the document does not contain any tags or a TOC. 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. These list structures typically state the main concept first, and, then, the supporting points are described in a bulleted, numbered or in-text list as illustrated in Figure 2. In these lists, an item 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. Once list items are identified, we consider the sentence appearing prior to the list items as the introductory sentence and assume that it governs all the 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. We consider the sentence appearing prior to the list items as the introductory sentence and assume that it governs the list items.

The categories of the products State Farm offers are as follows:
- We have property and casualty insurance.
- We offer comprehensive types of life and health insurances.
- We have bank products.

Figure 2: A sample text with a bulleted list

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 in the embedding of the 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 (Kiros et al., 2015; Gan et al., 2017). 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 left sentence.

4 NEURAL NETWORK MODELS

Similarly to SKIP-THOUGHT (Kiros et al., 2015), 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. Furthermore, we handle (OOV) words based on their occurrences in the context. Our model has several encoders (one encoder for each $G_i \in G$), a decoder and an OOV handler as shown in Figure 3.

The input to each cell is a word, represented as a dense vector. In this work, we use the pre-trained vectors from the CBOW model (Mikolov et al., 2013b), and the word vectors can be optionally updated during the training step. 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 by averaging all word vectors, we introduce a new OOV handler in our model. The OOV handler maps all OOV words appearing in governing sentences to placeholders and extend the vocabulary with the OOV placeholders. More details about OOV handler is described in Section 5.

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

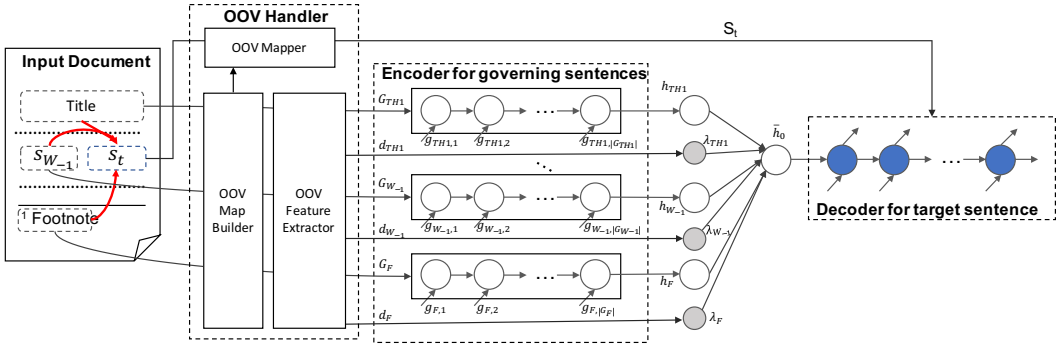


Figure 3: Our model architecture.

$G_i = (g_{i,1}, \ldots, g_{i,|G_i|})$ 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 S_i.

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

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 vector 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; $\mathcal{S}(\cdot)$ is the sigmoid function; u_{dep} and a_{dep} are dependency-specific weight parameters; $W_{\mathrm{dep}(i)}$ is the transformation matrix for the dependency type $\mathrm{dep}(i)$; 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:

$$o_t, \bar{h}_t = \mathrm{RC}(o_{t-1}, \bar{h}_{t-1}; \theta_D), \qquad y_t = \sigma(V o_t + c) \tag{2}$$

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; $\sigma(\cdot)$ 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 $\theta_E, W_{\mathrm{dep}(i)}, b, V, c, \theta_D$ and optionally $w(\cdot)$.

5 Out-of-Vocabulary (OOV) Mapping

Incorporating all the words from a large text collection in deep learning models is infeasible, since the amounts of memory use and training time will be too costly. Existing sentence embedding techniques reduce the vocabulary size mainly by using only high frequency words. Then, they create an arbitrary word vector by averaging all the word vectors in the vocabulary and map all the discarded words into the vector. However, this frequency-based filtering can loose many important words including domain-specific words and proper nouns and can produce unsatisfactory results for technical documents. Specifically, 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 input embeddings of G or S_t, we map all OOV words to the average vector of the embeddings of all other words in the sentence in the same way as other methods. However, when we produce the word labels (output logits) for S_t, we handle the OOV words as follows. We first select N most frequent words in the training corpus as an initial vocabulary V_0 in the same way as existing methods. Note that N (typically, tens of thousands) is much smaller than the size of a training corpus (typically, millions or billions). How can we model this many OOV words without deteriorating the

Algorithm 1: Building OOV map

Function BuildOOVMap $(G, \text{Vocabulary})$
Input : A governing sentence set $G = \{G_1, \ldots, G_{|G|}\}$ and a vocabulary V
Output: OOV Map
foreach $G_i \in G$ **do**
  $\text{OOVWords}_i \leftarrow \{w_j \in G_i | w_j \notin V, j = 1, \ldots, \eta\}$
  $\text{WtoPL}_i \leftarrow \{w_j \rightarrow O_i(j) | w_j \in \text{OOVWords}_i, j = 1, \ldots, |\text{OOVWords}_i|\}$
return $\bigcup_i \text{WtoPL}_i$

system performance? We note that only the OOV words appearing in governing sentences influence in model training, and many semantically important words tend to appear in the beginning or at the end of the governing sentences. Based on these observations, we select only the first and the last η OOV words in a governing sentences, but, other selection methods can be used. Further, to minimize the vocabulary size increase, we map individual OOV words to *OOV variables* based on the dependency type and the location of the OOV words. We denote OOV variables by $O_i(j)$, where i denotes the i-th dependency type and j is the index of the OOV word in the governing sentence.

The size of the augmented vocabulary is clear. Let $\mathcal{D}$ denote the a set of all possible dependency types. For each dependency $i \in \mathcal{D}$, we add $2 \times \eta$ OOV variables, resulting in $2 \times \eta \times |\mathcal{D}|$ new words, and, thus, the extended vocabulary $V = V_0 \cup \{2 \times \eta \times |\mathcal{D}| \text{ OOV variables}\}$). We conduct experiments with 10 dependency types as described in Section 3 and set η to 4 adding 80 unique OOV variables to the vocabulary. Note that the mapping between OOV words and OOV variables is many-to-many. For example, suppose "We discuss Keras first' is a target sentence S_t, and, "Slim and Keras are two tools you must know" is extracted as the document title by the dependency type $\mathcal{D}_{TS}$, "PLA's weekly review: Slim and Keras are two tools you must know" is extracted as the document title by $\mathcal{D}_{TM}$ for S_t, and, words 'Slim', 'Keras' and 'PLA' are OOV words. Then, we map the 'Slim' and 'Keras' from the first title to OOV variable $O_{TS}(1)$ and $O_{TS}(2)$ and 'PLA', 'Slim' and 'Keras' from the second title to $O_{TM}(1)$, $O_{TM}(2)$, and $O_{TM}(3)$ respectively. As a result, 'Keras' in S_t is mapped to $O_{TS}(1)$ and $O_{TM}(3)$.

Our OOV handler performs the following steps. First, we build an OOV map to convert OOV words to OOV variables and vice versa. Algorithm 1 summarizes the steps to build a map which converts the first η OOV words into OOV variables. To model the last η OOV words, we reverse the words in each G_i, and index them as $w_{-1}, w_{-2}, \ldots$, then pass them to *BuildOOVMap* to construct $O_i(-1), O_i(-2), \ldots, O_i(-\eta)$.

Once we have the OOV mapping and the augmented vocabulary, we can formulate an optimization goal taking into account the OOV words. 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 be OOV words. 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 # of OOV words whose initials are uppercased, # of OOV words that are uppercased, and # of OOV words with any of the letters are uppercased. Together with the linear regression parameters, $\mathcal{U}$ and g, the model learns the weights for different dependency types.

6 EXPERIMENTS

In this section, 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

Table 2: Comparison of target sentence prediction using the average loss per sentence.

Method	Avg. Loss
OURS-OOV	0.1548
OURS	0.1859
SKIP-THOUGHT-OOV	0.1907

Table 3: Comparison of paraphrase detection accuracy

Method	OURS	SKIP-THOUGHT
Accuracy	0.72	0.67

807,647 randomly selected documents from the 2009 Wikipedia dump, after removing the discussion and resource (*e.g.*, images) sections from the wiki pages. Since our approach leverages HTML tags to identify document structures, our model used the raw HTML files. For the baseline systems, we provided plain text version of the same articles. All models were trained for 300K steps with 64-sized batches and the Adagrad optimizer (Duchi et al., 2011). For the evaluation, we used up-to 8 governing sentences as the context for a target sentence. When a sentence has more than 8 governing sentences, we randomly choose 8 sentences. We set the maximum number of words in a sentence to be 30 and padded each sentence with special start and end of sentence symbols.

6.1 TARGET SENTENCE PREDICTION

Unlike most other approaches, our model and SKIP-THOUGHT (Kiros et al., 2015) can learn application-independent sentence representations without task-specific labels. Both models are trained to predict a target sentence given a set of context. For a quantitative evaluation between the two models, we compare the prediction losses by using the same loss function. We randomly chose 640,000 target sentences for evaluation and computed the average loss over the 640K sentences. Since SKIP-THOUGHT cannot handle OOV words, we excluded the loss values corresponding to the OOV words from both models and denoted them as OURS-OOV and SKIP-THOUGHT-OOV. However, OOV denotes our model including the loss values of OOV words. Table 2 shows the comparison of the three models. As we can see, both OURS-OOV and OURS outperform SKIP-THOUGHT-OOV resulting in 19% and 2.5% reduction in the loss values respecitively.

6.2 PARAPHRASE DETECTION

Further, we compare our model with SKIP-THOUGHT on a paraphrase detection task using the Microsoft Research Paraphrase corpus (Microsoft, 2016). The data consists of 5,801 sentence pairs extracted from news data and the assessment (if a sentence is a paraphrase or not), which were determined by three assessors using majority voting. We used 4,076 pairs for training and 1,725 pairs for testing. Since the data sets contains sentence pairs only and no structural context, we evaluate only the effectiveness of the trained encoder. To compare the qualities of sentence embeddings by the two models, we use the same logistic regression classifier with features based on embedded sentences as in (Kiros et al., 2015). 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 3), demonstrating the effectiveness of our encoder trained with the structural dependencies. Note that SKIP-THOUGHT trained on the Wikipedia corpus performs worse than a model trained on books or movie scripts due to more sophisticated and less sequential structure in Wikipedia documents.

6.3 COREFERENCE RESOLUTION

Traditionally, the coreference resolution problem is considered as a pairwise classification problem or clustering problem using handcrafted features (Clark & Manning, 2016). We show that our approach can be used to extract features for coreference resolution and perform coreference resolution without handcrafted features and supervision. We treat the coreference resolution problem as an inference problem given the context . We first identify a pronoun or an entity reference (*e.g.*, a definite noun phrase) and build a list of candidate referents that conform to the pronoun type and the entity type of the reference. Then, we replace the entity reference with each of the candidate referents and choose the one with the lowest loss value as the result, if the ratio of its loss to the original sentence loss value is less than a threshold value θ.

Table 4: Overall performance on coreference resolution

Method	Precision	Recall
OURS ($\theta = 0.99$)	0.47	0.12
OURS ($\theta = 1.00$)	0.35	0.17
DEEPCOREF	0.13	0.10

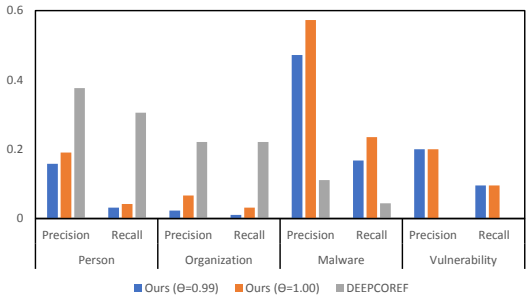


Figure 4: Performance per entity types

We compare our approach with the Stanford Deep Coreference Resolution tool (Clark & Manning, 2016) (denoted as DEEPCOREF henceafter), that leverages both handcrafted features and word embeddings to build distributed representations of coreference cluster pairs. We conducted experiments for several cybersecurity related entity types such as 'Malware' and 'Operating System' in addition to general entity types including 'Person' and 'Organization'. The evaluation dataset consists of 563 coreferences from 38 Wikipedia articles about malware programs that appeared after year 2009 and were not included in the training document set. For the evaluation, we set θ to 0.99 and 1.00.

Table 4 summarizes the results of the two systems. Our model achieves higher precision and recall than DEEPCOREF. Since DEEPCOREF was trained for a general domain, its overall performance on domain specific documents is very low. Figure 4 shows the two systems' performance on different entity types. As we can see, OURS works well for domain specific entities such as 'Malware' and 'Vulnerability', while DEEPCOREF shows higher precision for 'Person' and 'Organization'. The reason OURS performs worse for 'Person' and 'Organization' is the security documents do not provide much information to find dependencies as evidence to infer the coreferences.

6.4 DEPENDENCY IMPORTANCE

Note that u_{dep} in Equation 1 denotes the importance level of a dependency dep. In Table 5, we show the relative importance of the different dependencies compared to the sequential dependency ($\mathcal{D}_{W,-1}$), which is used in other methods. The values show their relative importance levels when the importance of $\mathcal{D}_{W,-1}$ is set to 1. As we can see, all levels of document and section titles, except the fourth level subsection title, play a much more significant role than the sequential dependency. The reason 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 5: The weights of different dependency types. * indicates non converging dependencies.

dep	$\mathcal{D}_{TM}$	$\mathcal{D}_{TH1}$	$\mathcal{D}_{TH2}$	$\mathcal{D}_{TH3}$	$\mathcal{D}_{TH4}$	$\mathcal{D}_{TH5}$	$\mathcal{D}_{TC}$
$\lvert u_{dep}/u_{\mathcal{D}_{W,-1}} \rvert$	1.00	2.30	2.30	2.30	0.24	1.40	2.94*
dep	$\mathcal{D}_{TR}$	$\mathcal{D}_{TS}$	$\mathcal{D}_{LT}$	$\mathcal{D}_{LF}$	$\mathcal{D}_H$	$\mathcal{D}_F$	
$\lvert u_{dep}/u_{\mathcal{D}_{W,-1}} \rvert$	1.23*	0.08*	2.67*	1.00	1.00	1.00	

7 CONCLUSION

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

REFERENCES

Kevin Clark and Christopher D. Manning. Improving coreference resolution by learning entity-level distributed representations. In *Association for Computational Linguistics (ACL)*, 2016. URL https://github.com/clarkkev/deep-coref.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *The Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Learning generic sentence representations using convolutional neural networks. 2017.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 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.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, pp. 1746–1751, 2014.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 3294–3302, 2015.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 957–966, 2015.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, (ICML)*, pp. 1188–1196, 2014.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 302–308, 2014.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 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, Beijing, China, July 2015. Association for Computational Linguistics.

Microsoft. Microsoft research paraphrase corpus. https://www.microsoft.com/en-us/download/details.aspx?id=52398, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR)*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 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, 2013b.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2011.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 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.

John Wieting and Kevin Gimpel. 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.