A discourse-aware neural network-based text model for document-level text classification

Kangwook Lee
School of Computing, KAIST, South Korea

Sanggyu Han
School of Computing, KAIST, South Korea

Sung-Hyon Myaeng
School of Computing, KAIST, South Korea

Abstract
Capturing semantics scattered across entire text is one of the important issues for Natural Language Processing (NLP) tasks. It would be particularly critical with long text embodying a flow of themes. This article proposes a new text modelling method that can handle thematic flows of text with Deep Neural Networks (DNNs) in such a way that discourse information and distributed representations of text are incorporate. Unlike previous DNN-based document models, the proposed model enables discourse-aware analysis of text and composition of sentence-level distributed representations guided by the discourse structure. More specifically, our method identifies Elementary Discourse Units (EDUs) and their discourse relations in a given document by applying Rhetorical Structure Theory (RST)-based discourse analysis. The result is fed into a tree-structured neural network that reflects the discourse information including the structure of the document and the discourse roles and relation types. We evaluate the document model for two document-level text classification tasks, sentiment analysis and sarcasm detection, with comparisons against the reference systems that also utilise discourse information. In addition, we conduct additional experiments to evaluate the impact of neural network types and adopted discourse factors on modelling documents vis-à-vis the two classification tasks. Furthermore, we investigate the effects of various learning methods, input units on the quality of the proposed discourse-aware document model.

Keywords
Deep learning; discourse analysis; neural network; sarcasm detection; sentiment analysis; text classification; text model

1. Introduction
Document-level Natural Language Processing (NLP) tasks are often best handled by methods that require representations of documents. Previously, the ‘text-as-a-whole’ approach has been dominant, such as having bag-of-words representations for retrieving and classifying text and using Latent Dirichlet Allocation (LDA) for capturing and representing latent semantics of documents. However, semantic information that comes from a sequence of smaller text chunks, which is crucial for identifying the main theme of text, has not been exploited much for NLP tasks. Take, for example, a short paragraph ‘The car looks nice, and even its price is reasonable. But I don’t like it’, which can be divided into three segments (The car looks nice,) (and even its price is reasonable.) (But I don’t like it.). The first and second segments can be merged into a larger segment since both convey information of the car. Next, the merged segment and the third segment...
can then form a larger segment that embraces the entire paragraph and express the author’s contrasting sentiment on the car, changing the overall meaning and the sentiment polarity of the paragraph.

Meanwhile, some NLP tasks have been successfully tackled with text representations using Deep Neural Networks (DNNs). Two major approaches to constructing document-level representations of a DNN are context-based learning and composition-based learning [1,2]. A context-based learning trains a distributed representation with word co-occurrence information. This approach has been shown to perform well on various text classification tasks and hence deemed to adequately abstract meanings of words. Although its ability lies in computing the overall semantics of text just based on words, it is not clear whether its effectiveness can be extended to a situation where flows of themes in a document may be critical for understanding the semantics. The composition-based learning method, on the other hand, combines the representations of smaller segments (e.g. words and sentences) for making a representation of a larger text. This learning method in general is more amenable than context-based learning for modelling the thematic flow of the text. However, the past approaches have rarely incorporated thematic information explicitly that runs across multiple sentences, that is, discourse, although such information has played an important role in previous approaches, not based on DNN, for NLP tasks [3,4].

In this article, we propose a new method for computing a distributed representation of a document using its discourse information as a way of constructing a document model that takes into account the thematic flow of the text. According to Rhetorical Structure Theory (RST) [5], the semantics of text is formed from smaller segments of the text and their discourse relations. The discourse relations are paratactic or hypotactic relations that hold across two text spans, that is, Elementary Discourse Units (EDUs), each of which is considered as either a nucleus or a satellite for its discourse role in the relation. Each discourse relation has a discourse relation type which can be defined by its purpose in the discourse. Thus, we tackle the problem of incorporating discourse information, namely, discourse structure, discourse role and discourse relation type, in modelling text with DNN. While this new DNN-based method has a potential to serve as an enhanced text modelling technique based on the RST, this article focuses on its application for text classification tasks that require an understanding of the underlying thematic flow of the text.

Considering discourse information in modelling text with DNNs is challenging because most neural network models are not suitable for incorporating the discourse information; they do not have the capability of adopting linguistic information that can be acquired by an external source. Thus, a specialised neural network is necessary for learning a document-level distributed representation with RST-based discourse analysis results — discourse structure, discourse role and discourse relation type. For the purpose of integration, we design a discourse-aware tree-structured neural network that incorporates discourse information obtained from a discourse parser. The proposed tree-structured neural network has two variants depending on which neural network the model utilises Recursive Neural Network (RecNN) [6] and tree Long Short-Term Memory (tree LSTM) [7], an extended version of LSTM [8].

Tree-structured neural networks can suffer from the inherent problem of vanishing gradients in the training process especially when the hierarchy of the tree-structured neural network grows deep for a long text. Such a long path in the backpropagation process may lead to a serious degradation of performance caused by a lack of annotations [9]. In order to alleviate the problem, we compute sentence embeddings with a separate context-based learning method and use the embeddings as the atomic inputs (i.e. EDU embeddings) for the proposed neural network. By doing so, we can shorten the hierarchy of the tree-structured network and avoid the need for annotations at many levels, such as words, phrases and clauses, which are usually essential to guarantee the performance. In detail, the proposed tree-structured neural networks adopt discourse roles and discourse relation types in the composition process to build a distributed representation of a document. To incorporate the discourse role in the modelling process, the proposed network reorders children nodes (in the RecNN-based method) or assigns different weight matrices to children nodes (in the tree LSTM-based method) based on the discourse roles of the children. Moreover, to incorporate the discourse relation type information of a pair of discourse units, we adopt special embeddings, referred to as discourse relation type embeddings. We employ a discourse parser, Discourse Parsing from Linear Projection (DPLP) [10], which detects segmented text spans, that is, EDUs, in a document and classifies discourse relations among the EDUs. For the separate process of training EDU embeddings, we utilise Paragraph Vector [11], which is one of the most well-known methods for learning embeddings of text with arbitrary lengths.

The ability to represent the thematic structure of text is invaluable for various NLP applications that can benefit from understanding the flow of semantics in text. In order to demonstrate such a potential of the proposed method, we apply it to two practical document-level text classification tasks, sentiment analysis and sarcasm detection, for which understanding semantic flows of documents is of utmost importance. The tasks have also been studied quite extensively with appropriate text collections [12–14], resulting in well-known state-of-the-art performance, which allows for meaningful comparisons with experiments. Note that we only assume that the target document can only have a single-class label in the classification tasks; extending the proposed method to the multi-label classification problem requires major changes.
on the design of the loss function and the cell computation mechanism, which should be studied in a separate work. There are two objectives for our experiments. One is to demonstrate the potential of the proposed method for the two application tasks, comparing with reference systems that utilise discourse information. The other is to explore the effects of modelling factors, including types of neural networks, adopted discourse information, input units and composition methods, on the quality of the document model.

The main contributions of this study can be summarised as follows:

- We propose novel tree-structured neural networks for modelling a document with full discourse information, including discourse structure discourse role, and discourse relation types, of the document.
- By applying a separate training process for EDU embeddings, we ease the vanishing gradient problem of the composition-based learning for long text while reducing necessary memory size in the training process of tree-structured neural networks.
- Through a thorough comparison of the proposed method against baselines and reference systems, we cast light on the effectiveness of incorporating discourse information in DNN for text modelling.
- We investigate effects of modelling aspects, such as types of neural networks and adopted discourse factors, in improving the quality of the distributed representation built with the discourse-aware tree-structured neural networks.

We open all the implementations proposed in this article to the public via an online repository.¹

2. Related work

DNN has been successfully applied to some NLP tasks such as classification by modelling texts of various granularities. We briefly review the characteristics of two major approaches to text modelling, context-based learning and composition-based learning, as they are both incorporated into our work.

The context-based learning approach dates back to when the traditional learning algorithms for distributed representations were proposed [15]. Often referred to as embeddings, they capture semantics of pieces of text in such a way that the text segments can be correctly predicted with their contexts. It has been widely adopted as a language model for words [16–18]. The distributed representations of words store the semantics in real-valued dense vectors that support tasks such as similarity-based word clustering and word sense disambiguation. In order to learn distributed representations of larger units of text, Le and Mikolov [11] proposed Paragraph Vector based on Word2Vec [17], one of the well-known methods for computing word embeddings. A follow-up study applies Paragraph Vector to document-level NLP tasks and analyses its effect [1].

A drawback of these approaches, however, is that they need a relatively large amount of training data and well-tuned hyperparameters to avoid the overfitting problem. For example, the effectiveness of context-based learning can diminish rather rapidly with rare tokens for which securing enough examples for learning context would be difficult. Furthermore, it does not take into account the order of words nor thematic flow of text such as discourse. This may lead to a significant failure on some text analysis tasks since it cannot capture the semantic difference caused by the distinct compositions of the text; for example, the meanings of two sentences ‘I sing the song because I like it’ and ‘I like the song because I sing it’, which consist of exactly same words, are different because of the ways the words are composed.

The other approach to modelling text with DNN draws on compositional semantics. Convolutional Neural Network (CNN), which can automatically capture semantics of text by weaving word embeddings via multiple convolution layers, has been widely used for modelling text [19–22]. In CNN, the vectors at the bottom layer pass through upper layers, and a single vector that represents the semantics of the text is approximated by a convolution operation. Stepwise methods such as recurrent neural networks and LSTM networks have been shown to be effective for semantic composition [23]. Lin et al. [2] proposed a language model based on a hierarchical recurrent neural network, which gradually learns from word-level embeddings to a document-level embedding through sentence-level embeddings. Socher et al. [24] introduced several RecNNs for sentence-level semantic composition using the parse trees of the sentences. Adopting latest DNN techniques to composition-based learning, Zhu et al. [9] proposed an extension of LSTM for structural neural networks, in which a node can reflect the history memories of multiple child nodes and multiple descendant nodes in a recursive process at the sentence level. A similar approach, tree LSTM [7], was introduced to enable LSTM networks to adopt the structural information. Moreover, Chen et al. [25] proposed a gated RecNNs to model a sentence without any external structure. Instead, it employs a Full Binary Tree (FBT) structure to control the combinations in recursive structure.

One limitation of these composition-based approaches is that they are supervised learning methods and therefore need annotations in the training process. The tree-structured neural networks such as RecNN and tree LSTM in particular need

¹ Journal of Information Science, 2017, pp. 1–21 © The Author(s), DOI: 10.1177/0165551517743644
annotations for every junction in the composition process. More importantly, none of the DNN-based approach has been able to accommodate inter-sentential semantics, perhaps because such a deep hierarchy from words to a document is prone to causing a vanishing gradient problem and ruinous memory in the training process of tree-structured neural networks. Since the hierarchy of a tree-structured neural network can grow exponentially as the length of input text becomes longer, the length of the text that can be processed with tree-structured neural networks may be limited.

3. Incorporating discourse information in a tree-structured neural network

As a way to capture deeper semantics or a flow of themes of a document, we introduce a new DNN-based document modelling method that incorporates the result of discourse analysis. Figure 1 shows the overall scheme, where the distributed representation of a document is computed based on the proposed tree-structured neural networks, in the form of either RecNN or tree LSTM, with the EDU embeddings and their discourse structures, roles and relation types.

Computing EDU embeddings, as opposed to using a deep hierarchy from words to a document, is an important design decision. By training EDU embeddings separately and feeding them to the proposed tree-structured neural network, it became possible to minimise the vanishing gradient problem and necessary memory space in the training process. This two-step scheme also makes it possible to train document representations with only document-level annotations by resolving the long-term dependency problem.

To minimise the vanishing gradient problem and necessary memory space for the training process, we propose a two-step approach that separately trains EDU embeddings and feeds them to the proposed tree-structured neural network that incorporates discourse information. In addition, this two-step scheme makes it possible to train document representations with only document-level annotations because the signals from documents do not have to propagate all the way down to words, thereby alleviating the long-term dependency problem.

The most significant aspect of the proposed architectures lies in the way the full discourse information is incorporated into the tree-structured neural networks. While the structure of the neural network is congruent to the discourse parse tree, the discourse roles of the descendants determine the order by which the descendants’ distributed representations are concatenated in the RecNN. In the tree LSTM, on the other hand, separate weight matrices are assigned to the descendants so that the cell computations, that is, the computations in nodes, are determined by whether the descendant is a nucleus or satellite. In order to incorporate different discourse relation types, furthermore, we introduce relation embeddings as parameters so that they take part in representing the semantics of EDU pairs. In short, the architectures of the RecNN and tree LSTM are chosen to reflect the discourse structure, whereas discourse roles and relation types enrich the semantic representations of the EDUs by reflecting their flows in terms of directionality and the nature of the connections. Technical details of the actual computations are found in Section 3.3.

![Figure 1](https://example.com/figure1.png)

Figure 1. An overview of the proposed tree-structured neural networks for modelling a document. It utilises three discourse factors, that is, discourse structure, discourse roles and discourse relation types, while the atomic text segments for discourse, that is, EDUs, are separately trained by Paragraph Vector.
3.1. Discourse parsing

Understanding the discourse structure of a document is analogous to parsing a sentence. Similar to a hierarchical structure of a sentence constructed with the syntactic relations among the words, the thematic discourse relations among the text segments are represented. As an effort to discover hierarchical discourse relations in text, Mann and Thompson [5] proposed RST, a compositional model of discourse where EDUs are gradually combined into larger discourse units until covering the entire document. The discourse relation between a pair of EDUs can be categorised by the discourse relation types such as Elaboration, Conjunction and Concession. At the same time, the EDUs are labelled as either a nucleus or a satellite to reflect their discourse roles in the relation. In the rest of this article, we particularly refer to the discourse unit formed by combining a pair of discourse units as Intermediate Discourse Unit (IDU) while EDU refers to only the atomic discourse unit. In addition, we will refer to the union of EDU and IDU as Discourse Unit (DU) for convenience.

There are handful RST-based discourse parsers, and we employ the DPLP tool [10] because it shows the best performance in identifying discourse relations, one of the most difficult subtasks in discourse parsing. To boost the performance of discourse analysis, we remove markup tags, punctuations and unicode control characters from the text. Figure 2 shows an example of a discourse parsing result. When EDU 1A and EDU 1B are combined with an Elaboration relation, 1B plays a role as a satellite in the relation with its nucleus 1A. They form a larger discourse unit for a Background relation with another IDU made of 1C and 1D.

The first work of discourse relations in the literature defined a set of 32 discourse relation types [5], which was extended to a set of 78 discourse relation types not only for improving granularity of discourse relation types but also for handling discourse roles and discourse relation types simultaneously [26]. Several simplified versions of the discourse relation types were proposed later [27,28]. Table 1 shows a discourse relation type hierarchy gleaned from previous studies [26–28].

In this study, we opt for the extended version containing 78 discourse relation types [26] because the DPLP parser analyses a document’s discourse with the extended version. The final set includes three special discourse relation types (i.e. Textual-Organisation, Span and Same-Unit) to impose the structure of the parse tree and another special discourse relation type, Sole-EDU, to represent a document consisting of single EDU.

3.2. Training EDU embeddings

After identifying EDUs of a document with the discourse parser, we compute distributed representations of the EDUs with the Paragraph Vector approach [11], which learns the distributed representations from unlabelled data. Note that, in this article, we set the smallest EDUs to sentences instead of smaller text units such as phrases and clauses. The main reason is to minimise the long-term dependencies caused by a deep parse tree that leads to degradation of composition-based learning [9]. Another reason is to alleviate the negative effect of the errors in segmenting text into EDUs (18.4% error rate [10]). Since the length of EDUs based on the definition of RST is always below sentences, the errors on EDU segmentation could be lightened via this simple procedure.

EDU embeddings are trained by Paragraph Vector, which learns a distributed representation of a multi-word text segment with a method similar to Word2Vec [17]. The training process of Word2Vec is designed to predict the next word’s embedding with embeddings of previous words in the context window (Continuous Bag-of-Words (CBOW)) and to predict words in the context window with a given word’s embedding (skip-gram). Although the word embeddings are initialised randomly at the start of the training process, they can finally encode semantics as an indirect result of the prediction task. Similar to this intuition, an embedding of a text segment can be trained. Extending the CBOW and skip-gram methods developed for Word2Vec, Paragraph Vector handles text of varying lengths with Distributed Memory (DM) and Distributed Bag-of-Words (DBOW) methods as explained below. Figure 3 shows the basic concepts of DM and DBOW.
In DM, every EDU is mapped to a unique embedding, denoted as \( p \), and every word is also mapped to a unique vector, denoted as \( w \). Given a sequence of training words in the target EDU \( w_1, w_2, \ldots, w_T \) and the size of the context window \( k \), the objective function of DM can be formulated as follows

\[
\frac{1}{T} \sum_{t=k}^{T-k} \log p(w_t | p, w_{t-k}, \ldots, w_{t+k})
\]

The EDU and word embeddings are averaged or concatenated to predict the next word in a context. Note that we use averaging in the current implementation. The context is sampled from a sliding window over the EDU. For the example of DM in Figure 3, the distributed representation of an EDU, \( p \), is trained by setting the objective of minimising the error in predicting the embedding of the word, \( w_3 \), with the embeddings of its context words' embeddings, \( w_1, w_2, w_4 \) and \( w_5 \), as well as \( p \). In this manner, the distributed representation \( p \) acts as a memory that remembers what is missing from the current context. The EDU embeddings are shared across all contexts generated from the same EDU but not across other EDUs, while the word embeddings are shared across all the EDUs. Both the EDU embeddings and word embeddings are trained using stochastic gradient descent and the gradient is obtained via backpropagation. To compute embeddings of EDUs in the test process, the same process is adopted while parameters of the model are fixed.

In DBOW, the EDU embedding is trained by predicting words randomly sampled from the EDU instead of predicting the next word with the context. At each iteration of stochastic gradient descent, random words in a text window sampled
from a given EDU are selected to form a classification task to predict the included words in the EDU. The objective function of DBOW can be formulated as follows

$$\frac{1}{T} \sum_{t=k}^{T-k} \log p(w_{t-k}, \ldots, w_t, \ldots, w_{t+k} | p)$$

(2)

For the DBOW example in Figure 3, the EDU embedding, \( p \), is trained to minimise the error in predicting the included words, \( w_1, w_2, w_3, w_4 \) and \( w_5 \). Like the DM method, DBOW method also computes embeddings of EDUs in the test process with fixed parameters. For evaluation, we pick the method, that is, DM or DBOW, and hyperparameters with preliminary experiments for each evaluation data set. The details of the preliminary experiments are described in Sections 4.1.2 and 4.2.2.

3.3. Document modelling with a discourse-aware neural network

This section describes the core of the proposed tree-structured neural networks for modelling documents. The three discourse factors, that is, discourse structure, discourse roles and discourse relation types, in a given document are discovered by the discourse parser based on RST. To take account of the discourse structure, we employ two classic tree-structured neural networks, RecNN [6] and tree LSTM [7], which can naturally accommodate the structural information of text. They can easily incorporate the tree-like discourse structure where leaf nodes correspond to EDUs, while the root node is recursively constructed from the underlying DUs. In this manner, the root is supposed to represent the RST-based compositional semantics of the entire document.

The tree-structured neural networks are extended with new ideas of incorporating discourse roles and discourse relation types. By reordering children nodes or assigning separate weight matrices to children nodes based on the discourse roles of the children nodes, we enable the tree-structured neural networks to capture the discourse role information. At the same time, we employ special embeddings for denoting discourse relation types and attach them to DU embeddings to involve the discourse relation type information. With these ideas, the full discourse information is well integrated into tree-structured neural networks.

3.3.1. Tree-structured neural networks for adopting discourse structure

We employ two types of tree-structured neural networks, RecNN and tree LSTM, for adopting discourse structure. RecNN is the oldest approach for adopting an external structure in the composition process. Although a number of advanced models of RecNN have been introduced [24,29], we employ a basic one to keep the composition process simple so that the effect of adopting the discourse structure can be clearly shown. Tree LSTM is a way of incorporating the structural information to LSTM instead of using the sequential order based on the appearance. We give detailed descriptions of the two below.

RecNN calculates the hidden state of a parent node \( p \), denoted as \( h_p \), with the hidden states of its two children \( c_1 \) and \( c_2 \) in the discourse structure, represented by the vectors \( h_{c_1} \) and \( h_{c_2} \). This process can be formulated as follows

$$h_p = \tanh(W \cdot [h_{c_1}, h_{c_2}] + b)$$

(3)

where \( W \) is a \( K \times 2K \) weight matrix and \( b \) is a \( K \times 1 \) bias vector with \( K \) being the dimension of DU embeddings. In a typical RecNN, the order of concatenating children follows the order of appearance.

Tree LSTM is a generalisation of the standard LSTM architecture and can easily represent tree-structured network topologies. While the standard LSTM composes its hidden state from the input at the current time step and the hidden state of the LSTM unit in the previous time step, the tree LSTM composes its state from an input embedding and the hidden states of the child nodes. Figure 4 shows an example of the composition process of tree LSTM.

Tree LSTM includes two variants distinguished by the way cells are computed: child-sum tree LSTM and N-ary tree LSTM. The major differences between these two variants are as follows: (1) child-sum tree LSTM is flexible to adopt any structural information (e.g. the number of children nodes can differ from their parents), while N-ary tree LSTM can take only a fixed number, \( N \), of children nodes and (2) N-ary tree LSTM can treat children nodes differently in the cell computation, while child-sum tree LSTM considers that all the children are the same. Because of these differences, the variants have different abilities to adopt discourse information. For instance, although both variants can be built on the discourse structure, the child-sum tree LSTM totally ignores discourse roles in the cell computation and treats the descendants the same. The N-ary tree LSTM, on the other hand, treats the descendants differently so that the descendants can be emphasised (or minimised) according to their discourse roles in the composition process. The detailed effects of using different tree LSTM variants will be investigated in Section 4.3.1.
For details of two variants of tree LSTM, let us start with child-sum tree LSTM. In child-sum tree LSTM, each node has three gates, that is, input gate, forget gate and output gate, which control the flow of computation in a node. Especially, the forget gate is a little different from that of a typical LSTM so that it can adopt children nodes’ information selectively. When the collection of child nodes of node $j$ is $C(j)$, the equation of child-sum tree LSTM is expressed as follows:

$$h_p = \sum_{k \in C(j)} h_k$$  \hspace{1cm} (4)$$

$$i_j = \sigma(W^{(i)}x_j + U^{(i)}h_p + b^{(i)})$$  \hspace{1cm} (5)$$

$$f_{jk} = \sigma(W^{(f)}x_j + U^{(f)}h_k + b^{(f)})$$  \hspace{1cm} (6)$$

$$o_j = \sigma(W^{(o)}x_j + U^{(o)}h_p + b^{(o)})$$  \hspace{1cm} (7)$$

$$u_j = \tanh(W^{(u)}x_j + U^{(u)}h_p + b^{(u)})$$  \hspace{1cm} (8)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k$$  \hspace{1cm} (9)$$

$$h_p = o_j \odot \tanh(c_j)$$  \hspace{1cm} (10)$$

where $i_j$ and $o_j$ denote the input gate and the output gate of node $j$, respectively, while $f_{jk}$ is the forget gate for a child node $k$ of node $j$. $h_p$, $u_j$ and $c_j$, respectively, represent the hidden output, the candidate value of cell state and the cell state of node $j$. Since the hidden output of child-sum tree LSTM depends on the sum of hidden states of child nodes, it is suitable even when the number of children is flexible.

N-ary tree LSTM can distinguish children nodes in the cell computation by assigning different weight matrices to children nodes, while the number of children nodes is fixed to $N$. In a typical N-ary tree LSTM, the children nodes can be ordered from 1 to $N$ according to the appearance order. N-ary tree LSTM can be formulated as follows:

$$i_j = \sigma\left(W^{(i)}x_j + \sum_{l=1}^{N} U^{(i)}_{j} h_{jl} + h^{(i)}\right)$$  \hspace{1cm} (11)$$
where the hidden output and the memory cell value of the \( j \)th child of node \( j \) are denoted as \( h_{jl} \) and \( c_{jl} \), respectively.

### 3.3.2. Incorporating discourse roles in tree-structured neural networks

For most discourse relations, the participating DUs play the role of either a nucleus or a satellite. To take account of discourse roles in the cell computation of tree-structured neural networks, we design a way of specialising the neural networks so that they can assign separate weight matrices to the child nodes according to their discourse roles. Note that we only design and apply this specialisation on RecNN and N-ary tree LSTM since child-sum tree LSTM is not capable of distinguishing children nodes in the cell computation.

In a RecNN, we rearrange the order of the concatenation of children nodes, which results in the fact that each child utilises the separate region of the weight matrix in the cell computation. For this, we modify the original equation of RecNN, equation (3), as follows

\[
h_p = \tanh(W \cdot \text{Concat}(h_{c_1}, h_{c_2}) + b)
\]  

where \( \text{Concat}(h_{c_1}, h_{c_2}) \) rearranges the order of the concatenation of the two child embeddings based on their discourse roles as defined below

\[
\text{Concat}(h_{c_1}, h_{c_2}) = \begin{cases} 
[h_{c_n}, h_{c_s}] & \text{if children play roles as nucleus and satellite, respectively} \\
[h_{c_1}, h_{c_2}] & \text{if both children play roles as nucleus}
\end{cases}
\]  

where \( c_n \) and \( c_s \) are the child nodes playing the nucleus role and the satellite role, respectively. The number on the child nodes, such as \( c_1 \) and \( c_2 \), means the appearance order: \( c_1 \) appears earlier than \( c_2 \).

Similarly, we propose a specialised N-ary tree LSTM designating weight matrices based on the discourse roles because all the discourse relations in RST formed with a pair of children nodes. The specialised N-ary tree LSTM for adopting discourse roles is expressed as follows

\[
i_j = \sigma(W^{(i)}x_j + U^{(i)}_N h_{N} + U^{(i)}_S h_{S} + b^{(i)})
\]

\[
f_{jk} = \sigma(W^{(f)}x_j + U^{(f)}_N h_{N} + U^{(f)}_S h_{S} + b^{(f)})
\]

\[
o_j = \sigma(W^{(o)}x_j + U^{(o)}_N h_{N} + U^{(o)}_S h_{S} + b^{(o)})
\]

\[
u_j = \tanh(W^{(u)}x_j + U^{(u)}_N h_{N} + U^{(u)}_S h_{S} + b^{(u)})
\]

\[
c_j = i_j \odot u_j + f_{jN} \odot c_{jN} + f_{jS} \odot c_{jS}
\]

\[
h_p = o_j \odot \tanh(c_j)
\]
3.3.3. Discourse relation type embeddings in tree-structured neural networks. As a way of incorporating the identified discourse relation types in a parsing result into tree-structured neural networks, we devise a method of utilising new embeddings in the composition process, separately from the DU embeddings. Once we utilise 82 discourse relation types (78 discourse relation types + 4 special purpose relation types) as mentioned in Section 3.1, the discourse relation type embedding for a specific relation type is concatenated to the DU embedding, that is, EDU embedding or IDU embedding computed by composing the embeddings of the descendants. The resulting embedding finally reflects the semantics of the way two chunks of text are combined. Figure 5 shows the concatenation of the DU embedding and the discourse relation type embedding in a node.

To adopt the concatenation in the cell computation, all the weight matrices of the proposed tree-structured neural networks are extended to adopt the concatenated embedding. For example, the weight matrix of RecNN is extended from $\mathcal{K} \times 2\mathcal{K}$ to $\mathcal{K} \times 2(\mathcal{K} + L)$ and the weight matrices of tree LSTM, $W$ and $U$, are extended from $\mathcal{K} \times \mathcal{K}$ to $\mathcal{K} \times (\mathcal{K} + L)$ where $\mathcal{K}$ and $L$ denote the dimension of DU embeddings and the dimension of discourse relation type embeddings, respectively. These discourse relation type embeddings are initialised and trained with other parameters for the document model, such as $W$ and $b$, via the backpropagation process of the proposed tree-structured neural networks.

3.3.4. Training details. For the two classification tasks, which we use to evaluate our models, a simple softmax layer is added to predict the conditional probability for each class with the document embedding composed via the proposed tree-structured neural networks. The softmax layer is defined as follows

$$\hat{p}(d) = \text{softmax}(W_{\text{softmax}} \cdot h_d + b_{\text{softmax}})$$

where $p(d)$ is a probability distribution of class labels for document $d$, $W_{\text{softmax}}$ is a $C \times K$ weight matrix and $b_{\text{softmax}}$ is a $C \times 1$ bias vector where $C$ and $K$ are the number of class labels in the target classification task and the dimension of DU embeddings, respectively. For a binary sentiment classification task, for example, $C$ would be two since there are positive and negative classes.

To train the parameters of the proposed tree-structured neural networks, we design a loss function using the cross-entropy error between the gold standard class distribution and the predicted class distribution of a document

$$\text{loss} = - \sum_{d \in TS} \sum_{i=1}^{C} p^g_i(d) \cdot \log(p_i(d)) + \lambda \|\theta\|^2$$

where $TS$ and $d$ represent a training set and a document, respectively. $p^g_i(d)$ denotes the conditional probability that the gold standard class of the document $d$ falls in the class label $i$, while $p_i(d)$ denotes the conditional probability that the predicted class belongs to the class. $\lambda$ is a coefficient for L2 regularisation.

Through BackPropagation Through Structure (BPTS), we take the derivative of the loss function with respect to the whole set of parameters $\theta$ and update the parameters with Adaptive Moment Estimation (Adam) [30]. Adam computes adaptive learning rates with one datum at each backpropagation step and updates the parameters as follows

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$
where $\eta$ is the learning rate and $\epsilon$ is a smoothing term that avoids division by zero. $m_t$ and $v_t$ are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients, respectively.

### 4. Evaluation

It is not straightforward to devise a general criterion for evaluating the quality of document models. Because of this reason, many studies on modelling documents were geared towards specific NLP tasks such as classification or text search and evaluated accordingly. As our motivation for this research is to model documents with their discourse information, we thought that any task that would benefit from analysing the argument structures could serve the purpose of evaluation. Another point to consider for designing the experiment for evaluation is the availability of a public data set that facilitates a direct comparison with previous approaches. Considering the aforementioned reasons, we chose two document-level text classification tasks: sentiment analysis and sarcasm detection, each of which calls for a sophisticated understanding about the document’s thematic flow. The classification performance is measured with accuracy \( \text{Accuracy} = \frac{\# \text{correct predictions}}{\# \text{predictions}} \)\(^2\). Since the previous works on both of the classification tasks were measured only with accuracy \([32-34]\), we have no choice but to measure the performance with accuracy.

\[
\text{Accuracy} = \frac{\# \text{correct predictions}}{\# \text{predictions}}
\]

For a further understanding of our approach internally, for example, relative difficulty of the classes, we also use precision, recall and F1 score \([31]\)

\[
\text{Precision} = \frac{\# \text{true-positive instances}}{\# \text{true-positive instances} + \# \text{false-positive instances}}
\]

\[
\text{Recall} = \frac{\# \text{true-positive instances}}{\# \text{true-positive instances} + \# \text{false-negative instances}}
\]

\[
F_1 \text{ score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Since our proposed method highlights on the use of full discourse information derived from RST, which includes discourse structure, discourse roles and discourse relation types, our experiments should be designed in such a way that the results of the two tasks must shed light on the value of the discourse analyses in modelling documents. As such, our first experiment is to compare the proposed modelling method against the state-of-the-art reference systems for the two tasks. The reference system for sentiment analysis also utilises discourse information but with external resources, whereas the system for sarcasm detection achieves the best performance so far by even using human-created scores. The main proposed model being compared is referred to as Discourse-aware Neural Network (DaNN), which is an N-ary tree LSTM built on the discourse structure. It assigns different weight matrices to child nodes for their roles and includes additional embeddings for discourse relation types. Among the variations we introduced, it is the only one that can accommodate the full discourse information. Other proposed models, RecNN and child-sum tree LSTM, are unable to take discourse roles properly.

The second part of our experiments, as in Section 4.3, is to investigate the effects of computation cells of neural networks (RecNN, child-sum tree LSTM and N-ary tree LSTM) and discourse information (discourse structure, discourse role and discourse relation type). Furthermore, the effects of composition methods and input text units on the qualities of the document models are examined in Section 4.4. For this, we conduct a comparative experiment for various composition methods (average, sequential LSTM, end-to-end tree LSTM and end-to-end Paragraph Vector) with different input text units (word, sentence and whole document).

### 4.1. Sentiment analysis

Sentiment analysis would need an understanding of a particular aspect of semantics in text, for example, positive or negative sentiment. An intuitive way of determining the overall sentiment of a given document is to compute the sum of polarity values of the sentences constituting the document, and this method is suitable for ordinary sentiment analyses. However, we conjecture that the overall sentiment may vary depending on the way the sentences are intertwined, like the example in Section 1. If that is the case, the sentences would contribute unequally to the overall sentiment, depending on their roles in the entire discourse.

---

\[^2\] Accuracy = \#correct predictions / \#predictions
4.1.1. Data and task explanations. We utilise two well-known data sets: the Cornell movie review data set [32] and the Stanford large movie review data set [33], for sentiment analysis. They are annotated with the binary sentiment labels, positive and negative. The statistics of the two data sets are shown in Table 2.

The Cornell movie review data set consists of 1000 positive and 1000 negative reviews. Given the size of the data set, we use 10-fold cross-validation by splitting the data set at a ratio of 8:1:1 for train, development and test sets, respectively. The Stanford large movie review data set consists of 12,500 positive and 12,500 negative reviews as a train set and another 12,500 positive and 12,500 negative reviews as a test set. For the experiment, we divide 10% of the train set as a development set. Both data sets consist of movie reviews collected in IMDb.com.2

4.1.2. Experimental settings. First, to obtain EDU embeddings, we use the Gensim [35] library to utilise Paragraph Vector [11] described in Section 3.2. At the start of learning, we initialise word embeddings with Glove embeddings [18] to reduce the limitations resulting from the small amount of training instances in an unsupervised manner. To find optimal hyperparameters of Paragraph Vector for each data set, we conducted a sentiment analysis task with support vector machine (SVM) in the scikit-learn library,3 which receives document embeddings as input. In this step, we compute an embedding of a document by averaging its EDU embeddings, and all the hyperparameters are tuned with the development set. The hyperparameters of Paragraph Vector on learning EDU embeddings are tested within the values described in Table 3.

Note that only one of the hierarchical sampling or negative sampling methods can be selected as the sampling method at a time. Based on the experiment results, we finally select the values shown in Table 4 as the settings in learning EDU embeddings for each data set.

---

**Table 2.** Statistics of the Cornell movie review data set and the Stanford large movie review data set

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of documents</th>
<th>Number of EDUs</th>
<th>Number of unique EDUs</th>
<th>Number of words</th>
<th>Number of unique words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cornell</td>
<td>2000</td>
<td>162,482</td>
<td>142,422</td>
<td>1,402,412</td>
<td>48,018</td>
</tr>
<tr>
<td>Stanford</td>
<td>50,000</td>
<td>600,920</td>
<td>579,881</td>
<td>13,229,985</td>
<td>142,870</td>
</tr>
</tbody>
</table>

EDUs: Elementary Discourse Units.

**Table 3.** The tested values to find the optimum for each hyperparameter on learning EDU embeddings

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Tested values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning method</td>
<td>[DM, DBOW]</td>
</tr>
<tr>
<td>Dimension of EDU embedding</td>
<td>[50, 100, 200, 300]</td>
</tr>
<tr>
<td>Size of context window</td>
<td>[0, 5, 10, 15, 20]</td>
</tr>
<tr>
<td>Hierarchical sampling</td>
<td>[Yes, no]</td>
</tr>
<tr>
<td>Size of negative sampling</td>
<td>[0 (no), 5, 10, 15, 20]</td>
</tr>
</tbody>
</table>

EDU: Elementary Discourse Unit; DM: Distributed Memory; DBOW: Distributed Bag-of-Words.

**Table 4.** The selected values of hyperparameters in learning EDU embeddings for the Cornell and the Stanford data sets

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Cornell</th>
<th>Stanford</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning method</td>
<td>DM</td>
<td>DM</td>
</tr>
<tr>
<td>Dimension of EDU embedding</td>
<td>100</td>
<td>300</td>
</tr>
<tr>
<td>Size of context window</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Hierarchical sampling</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Size of negative sampling</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

EDU: Elementary Discourse Unit; DM: Distributed Memory.
For the proposed tree-structured neural network, we randomly initialise all parameters from a uniform distribution $U(0.01, 0.01)$. All the hyperparameters are heuristically chosen by the hyperparameter optimisation process with the development set. The tested values of hyperparameters of the proposed tree-structured neural networks are described in Table 5.

After the hyperparameter tuning process, we finally set the values for the tree-structured neural networks as shown in Table 6. The maximum epoch of the training process is set to 25 for both data sets.

### 4.1.3. Results

Although only few previous works exist for comparison, we provide a comparison with the state-of-the-art technique by Bhatia et al. [34], which also utilises discourse information and borrows the propagation-based training process of RecNN. While their approach utilises the discourse structure as the structure for the backpropagation, the approach adopts discourse roles and discourse relation types by applying separate weight matrices depending on discourse roles and discourse relation types of the DUs. The key significant difference of this approach from the proposed method is that it computes polarity scores of EDUs based on a pre-constructed sentiment lexicon (hence supervised) rather than using distributed representations constructed from free text (hence unsupervised). In other words, it borrows RecNN merely as a tool for combining polarity scores of EDUs, whereas the proposed method is an end-to-end approach using discourse information without requiring the availability of a pre-constructed sentiment lexicon that often determines the performance of sentiment analysis. For the comparison, we cite the result from their paper. Table 7 shows the result.

For the proposed tree-structured neural network, we randomly initialise all parameters from a uniform distribution $U(-0.01, 0.01)$. All the hyperparameters are heuristically chosen by the hyperparameter optimisation process with the development set. The tested values of hyperparameters of the proposed tree-structured neural networks are described in Table 5.

After the hyperparameter tuning process, we finally set the values for the tree-structured neural networks as shown in Table 6. The maximum epoch of the training process is set to 25 for both data sets.

**Table 5.** The tested values to find the optimal value for each hyperparameter of the proposed neural networks

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Tested values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>$[10^{-4.5} \times 10^{-5}, 1, 2, 10^{-5}, 5, 10^{-6}, 2 \times 10^{-6}, 10^{-7}]$</td>
</tr>
<tr>
<td>Dropout ratio</td>
<td>$[0, 0.1, 0.2, 0.3, 0.4, 0.5]$</td>
</tr>
<tr>
<td>Coefficient for L2 regularisation</td>
<td>$[0, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}, 10^{-11}]$</td>
</tr>
</tbody>
</table>

**Table 6.** The selected values of hyperparameters of the proposed neural networks for the Cornell and the Stanford data sets

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Cornell</th>
<th>Stanford</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>$5 \times 10^{-5}$</td>
<td>$10^{-5}$</td>
</tr>
<tr>
<td>Dropout ratio</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Coefficient for L2 regularisation</td>
<td>$10^{-10}$</td>
<td>$10^{-10}$</td>
</tr>
</tbody>
</table>

**Table 7.** The accuracy with the reference system on the document-level sentiment analysis task

<table>
<thead>
<tr>
<th>Method</th>
<th>Cornell</th>
<th>Stanford</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bhatia et al. [34]$^a$</td>
<td>0.841</td>
<td>0.856</td>
</tr>
<tr>
<td>DaNN</td>
<td>0.8755</td>
<td>0.8906</td>
</tr>
</tbody>
</table>

DaNN: Discourse-aware Neural Network.

$^a$The results of Bhatia et al.’s approach are cited from their paper.

For the proposed tree-structured neural network, we randomly initialise all parameters from a uniform distribution $U(-0.01, 0.01)$. All the hyperparameters are heuristically chosen by the hyperparameter optimisation process with the development set. The tested values of hyperparameters of the proposed tree-structured neural networks are described in Table 5.

After the hyperparameter tuning process, we finally set the values for the tree-structured neural networks as shown in Table 6. The maximum epoch of the training process is set to 25 for both data sets.

4.1.3. **Results.** Although only few previous works exist for comparison, we provide a comparison with the state-of-the-art technique by Bhatia et al. [34], which also utilises discourse information and borrows the propagation-based training process of RecNN. While their approach utilises the discourse structure as the structure for the backpropagation, the approach adopts discourse roles and discourse relation types by applying separate weight matrices depending on discourse roles and discourse relation types of the DUs. The key significant difference of this approach from the proposed method is that it computes polarity scores of EDUs based on a pre-constructed sentiment lexicon (hence supervised) rather than using distributed representations constructed from free text (hence unsupervised). In other words, it borrows RecNN merely as a tool for combining polarity scores of EDUs, whereas the proposed method is an end-to-end approach using discourse information without requiring the availability of a pre-constructed sentiment lexicon that often determines the performance of sentiment analysis. For the comparison, we cite the result from their paper. Table 7 shows the result.

This result is encouraging and significant from several perspectives. First, it demonstrates the proposed document modelling method is effective for the sentiment analysis task even beyond the state-of-the-art performance. It is particularly meaningful because the approach by Bhatia et al. requires additional knowledge — sentiment scores (scalar values) of the EDUs, computed by a separate classifier as well as a sentiment lexicon, whereas the proposed approach is an end-to-end DNN-based method relying solely on the available text. Second, the proposed method adopting the two-step approach is much less costly in preparing annotated data for learning because it only requires polarity labels for documents, whereas Bhatia et al.’s system needs not only annotations for documents but also predicted labels of the DUs to
minimise the vanishing gradient and the long-term dependency problems caused by the deep network hierarchy. Finally, unlike Bhatia et al.'s work specifically developed for sentiment analysis, the proposed method is for computing a multi-purpose document model using discourse information but can be applied successfully to sentiment analysis as well as others.

The dependency of the proposed method on the discourse parser is unavoidable unless we take the discourse parsing into an end-to-end-system for classification. It is worth noting that DaNN has a great potential for further improvement as the underlying techniques used in the current implementation are not sufficiently mature, leaving space for improvement. For instance, the performance of the DPLP parser is not ripe despite that is the best discourse parser in existence. They have relatively low F1 scores: 81.6% on EDU segmentation, 70.95% on discourse role classification and 61.75% on discourse relation type classification [10]. Moreover, DaNN’s capability of updating parameters at each node would help improving the performance if additional annotations at EDU-level labels and IDU-level labels are available. In sum, the experimental result shows that the proposed multipurpose document modelling method can be used successfully for sentiment analysis to outperform the state-of-the-art sentiment analysis system that utilises discourse information. Its performance would become even better if we reduce the errors of the underlying techniques and/or provide finer level annotations.

We also measure the performance of DaNN with precision, recall and F1 score on each polarity. Table 8 shows the results.

Table 8. The precision, recall and F1 score for each class on the document-level sentiment analysis task

<table>
<thead>
<tr>
<th>Method</th>
<th>Cornell</th>
<th></th>
<th></th>
<th>Stanford</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1 score</td>
<td>Precision</td>
<td>Recall</td>
<td>F1 score</td>
</tr>
<tr>
<td>DaNN (positive)</td>
<td>0.8653</td>
<td>0.8871</td>
<td>0.8761</td>
<td>0.8823</td>
<td>0.9013</td>
<td>0.8917</td>
</tr>
<tr>
<td>DaNN (negative)</td>
<td>0.8861</td>
<td>0.8641</td>
<td>0.8749</td>
<td>0.8993</td>
<td>0.8800</td>
<td>0.8895</td>
</tr>
</tbody>
</table>

DaNN: Discourse-aware Neural Network.

4.2. Sarcasm detection

We consider document-level sarcasm detection a useful task for evaluating the effect of the new document modelling method. Compared with sentiment analysis, sarcasm detection would require a more sophisticated understanding of the thematic flow since sarcasm is often recognised by the way sentences are linked and contrasted. Given that the proposed method is meant to capture a flow of themes with discourse information, sarcasm detection would be a natural choice for testing the new document modelling method.

4.2.1. Data and task explanations. Although several data sets for sarcasm detection studies are available, the majority deals with short text, such as tweets, which lacks discourse information. Thus, we employ the Sarcasm data set introduced by Filatova [36] for our experiments. This data set consists of 437 sarcastic reviews and 817 regular reviews from Amazon. Table 9 shows some statistics of the data set. Note that a document contains more than 30 EDUs on average.

Table 9. Statistics of the Sarcasm data set used in the experiment

<table>
<thead>
<tr>
<th>Number of documents</th>
<th>Number of EDUs</th>
<th>Number of unique EDUs</th>
<th>Number of words</th>
<th>Number of unique words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1254</td>
<td>42,036</td>
<td>38,160</td>
<td>322,378</td>
<td>21,023</td>
</tr>
</tbody>
</table>

EDU: Elementary Discourse Units.

4.2.2. Experimental settings. Like the sentiment analysis task, the relatively small size of the data set makes it necessary to adopt 10-fold cross-validation for the evaluation. Similarly, we ran a preliminary experiment for finding optimal
We also conducted an experiment for tuning hyperparameters of the proposed tree-structured neural networks with the same tested values in the sentiment analysis experiment, as described in Table 5 and set the hyperparameters with the values shown in Table 11.

4.2.3. Results. We quote the performance of the state-of-art sarcasm detection method by Buschmeier et al. [37], which does not use any discourse information, because we were not able to identify a reference system that utilises discourse information. It attempts to take a mixture of various features for detecting sarcastic documents, including nine text features, for example, punctuation and emoticon, and two human-created scores. One of human-created scores is the star rating of the product review, while the other is the difference between the star rating and the computed sentiment score of the document. That is, this method makes an explicit use of meta-features unavailable in the text as well as various textual features. In order to demonstrate that the proposed document modelling method indeed helps in the sarcasm detection task and tease out the value of using discourse information extracted from the text, we show two cases in Table 12: one with textual features only and the other with both textual features and the human-created scores. We obtained the values with the provided source code.

The result shows that when without the human-created scores, the proposed method outperforms the state-of-the-art method (the first line in Table 12), which uses nine separately extracted textual features [37]. This is an indication that the distributed representations of the documents as proposed in our work capture the clues for sarcasm, including those originated from discourse information (see Section 4.3.2 for more details of the effect). The result also shows that the proposed method does not beat the score of Buschmeier et al.’s best method (second line in Table 12), which relies on two human-created scores as well as the textual features. This comparison confirms that indirect human supervision in the form of the star ratings provides invaluable information that could not be captured with only word-level or discourse-level text analysis. However, it does not disqualify the proposed method as a new approach to sarcasm detection.

### Table 10. The selected values of hyperparameters in learning EDU embeddings for the Sarcasm data set

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Sarcasm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning method</td>
<td>DM</td>
</tr>
<tr>
<td>Dimension of EDU embedding</td>
<td>50</td>
</tr>
<tr>
<td>Size of context window</td>
<td>15</td>
</tr>
<tr>
<td>Hierarchical sampling</td>
<td>No</td>
</tr>
<tr>
<td>Size of negative sampling</td>
<td>20</td>
</tr>
</tbody>
</table>

EDU: Elementary Discourse Unit; DM: Distributed Memory.

### Table 11. The selected values of hyperparameters of the proposed neural networks for the Sarcasm data set

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Sarcasm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Dropout ratio</td>
<td>0.4</td>
</tr>
<tr>
<td>Coefficient for L2 regularisation</td>
<td>$10^{-10}$</td>
</tr>
</tbody>
</table>

### Table 12. The accuracy with the reference system on the document-level sarcasm detection task

<table>
<thead>
<tr>
<th>Method</th>
<th>Sarcasm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buschmeier et al. [37] (binary classifier with text features only)</td>
<td>0.7900</td>
</tr>
<tr>
<td>Buschmeier et al. [37] (binary classifier with text features + human-created scores)</td>
<td>0.8370</td>
</tr>
<tr>
<td>DaNN</td>
<td>0.8101</td>
</tr>
</tbody>
</table>

DaNN: Discourse-aware Neural Network.
that the human-created scores are neither always available in reality nor can be created easily based on the reviews. From this perspective, the result is still very encouraging because the proposed method simply attempts to capture the semantics of a document as faithfully as possible, including discourse information, without an effort to customise it for the task at hand and yet achieves a remarkable performance.

We also measure the performance of DaNN with precision, recall and F1 score on each class. Table 13 shows the results.

From the result, DaNN is weighted in favour of the regular documents. One possible reason for this result is the biased ratio between sarcasm and regular documents; the regular documents are nearly twice more than the sarcasm documents. Under this circumstance, the parameters of DaNN are naturally trained with a bias to support the major class, that is, regular. To avoid this bias problem caused by the imbalanced data set, a special training technique, such as transfer learning, may be needed, which would be considered in a separate study.

4.3. Effects of modelling factors of DaNN variants

4.3.1. Effects of types of neural networks. We conduct an experiment to explore how the computation cells of neural networks influence on the document modelling performance. For this, we build three variants of DaNN with different computation cells: RecNN, child-sum tree LSTM and N-ary tree LSTM. The variants described in Section 3.3 are designed to consider only the discourse structure information for a simple and fair comparison. Since the child-sum tree LSTM even does not have capability of distinguishing children nodes according to their discourse roles, for example, it would not be possible to compare them for accommodating discourse roles. Table 14 shows the experimental result.

The two tree LSTM–based variants clearly outperform the RecNN-based variant in all the data sets, probably due to the function of the selective memory in the tree LSTM. The tree LSTM–based variants can adopt an input, keep an intermediate memory and activate an output selectively in the training process. It appears that this general advantage of LSTM over RecNN is proven to work well with the EDU embeddings. The two tree LSTM–based variants are similar in their performance with a slight superiority of the N-ary tree LSTM–based variant on the Cornell and the Sarcasm data sets. We prefer N-ary tree LSTM to the child-sum tree LSTM, not because of the slight difference but because of the inability of the child-sum tree LSTM in distinguishing children nodes in the cell computation and thus reflecting discourse roles.

4.3.2. Effects of discourse factors. To see the contributions of discourse roles and relation types in addition to the availability of discourse structure, we run an ablation experiment with the two DNNs: RecNN and N-ary tree LSTM. As shown in Table 15, we first eliminate the effect of discourse roles (with ‘- discourse roles’ on the second row) or the performance of taking both discourse structure and discourse relation types. We also eliminate the effect of discourse relation types (with ‘- discourse relation types’ on the third row) and then eliminate both (with ‘- discourse roles and discourse relation types’ on the fourth row) so that we see the result of just considering discourse structure only.

Table 13. The precision, recall and F1 score for each class on the document-level sarcasm detection task

<table>
<thead>
<tr>
<th>Method</th>
<th>Sarcasm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
</tr>
<tr>
<td>DaNN (sarcasm)</td>
<td>0.7558</td>
</tr>
<tr>
<td>DaNN (regular)</td>
<td>0.8347</td>
</tr>
</tbody>
</table>

DaNN: Discourse-aware Neural Network.

Table 14. The accuracies of three variants of the proposed DaNN on the development set

<table>
<thead>
<tr>
<th>Method</th>
<th>Cornell</th>
<th>Stanford</th>
<th>Sarcasm</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecNN</td>
<td>0.8290</td>
<td>0.8500</td>
<td>0.7982</td>
</tr>
<tr>
<td>Child-sum tree LSTM</td>
<td>0.8845</td>
<td>0.8932</td>
<td>0.8206</td>
</tr>
<tr>
<td>N-ary tree LSTM</td>
<td>0.8880</td>
<td>0.8884</td>
<td>0.8254</td>
</tr>
</tbody>
</table>

DaNN: Discourse-aware Neural Network; RecNN: Recursive Neural Network; LSTM: Long Short-Term Memory.

In this experiment, all the methods consider only the discourse structure for a fair comparison.
As a way of validating the way discourse roles are incorporated in the N-ary tree LSTM, we adopt the method used in Hogenboom et al. [4] as a variant. They proposed a way of using discourse roles for the purpose of selecting text inputs for sentiment analysis. Since the method is based on a typical lexicon-based approach, we just borrow and apply their idea to the N-ary tree LSTM. In their approach, all the satellite nodes at the leaf level, that is, EDU level, are ignored so that an N-ary tree LSTM is built only with nucleus EDU nodes. Note that discourse roles are not considered at all for intermediate levels, that is, IDU levels, because the lost information from this process would become too large if we ignore all the satellite nodes and their descendants in the training process. Table 15 shows the results under the two architectures: RecNN and N-ary tree LSTM.

It is clear that the discourse information has stronger positive effects with RecNN. Given the performance drop was the largest with ‘- discourse roles and discourse relation types’, both discourse roles and discourse relation types have their shares in improving the classification tasks with an exception of discourse relation types for the sarcasm detection task (adding discourse relation types while eliminating discourse roles was the worst). For the sentiment analysis task, discourse relation types are more important than discourse roles in improving the result.

On the other hand, the differences resulting from eliminating the discourse information are relatively small with the N-ary tree LSTM. Although small, the discourse relation types have a positive impact for the sentiment analysis task but a negative one for the sarcasm detection task. The discourse roles have an opposite impact; they are positive in the sarcasm detection task but negative in the sentiment analysis task. It appears that the discourse structure is so well incorporated with the N-ary tree LSTM that the additional information, that is, the discourse roles and discourse relation types may have an adversary effect, and more accurate extraction would be more beneficial to the tasks.

An interesting point is that Hogenboom et al.’s approach of incorporating discourse roles in modelling documents shows poor performance. We suspect that this poor result comes from the information loss caused by their policy of ignoring satellite nodes at leaf levels. While this approach of using just nucleus parts may work for a lexicon-based approach, it fails in a composition-based DNN approach.

In conclusion, which adopted full discourse, including discourse structure, discourse role and discourse relation type, shows the best performance among the results, adopting discourse role and discourse relation type information simultaneously may help to improve the discourse-aware tree-structured neural network’s performance. The progressive improvements between the variants confirm the value of differentiating the nucleus and satellite roles and utilising the discourse relation types in integrating text semantics with tree-structured neural networks. Left as future research is a detailed analysis and investigation on the types of documents that would benefit most by discourse roles and discourse relation types in each of the tasks.

### 4.4. Effects of input text units and composition methods

#### 4.4.1. Experimental settings

The main goal of this experiment is to understand the effects of different composition methods and different ways distributed representations are computed. As such, we compare DaNN against other various DNN-based baselines. In a way, this shows how we arrived at DaNN in retrospect or what alternatives we have examined. The baselines vary in both their input text units and composition methods, and the results help understanding the internal workings of the proposed model. The comparison results are summarised in Table 16.
AvgWordVec computes a document’s embedding with an average of embeddings of the words included in the document. Given that a document embedding can be considered as its features, the SVM classifier in the scikit-learn library (see Note 3) is employed for the classification tasks. Note that the word embeddings are acquired as by-products in the process of learning EDU embeddings. More specifically, the word embeddings are trained with the CBOW method of Word2Vec [17] when the embeddings of EDUs are trained with the DM method of Paragraph Vector [11]. As such, all the hyperparameters used for the word embeddings are the same as those of EDU embeddings.

LSTMWordVec uses a single-layer sequential LSTM as its composition method, implemented with Theano [38] and Keras. The dimension of the hidden layer is set to 200 and a softmax layer is attached at the top for the classification tasks. The LSTM is trained with a loss function based on the binary-cross-entropy error between the gold standard and the predicted labels, and the learning process is optimised via Adam optimiser [30]. It serves as a baseline for an end-to-end classifier for the tasks, with a potential to capture the sequence of word semantics. However, it is not clear whether EDU- (or sentence-) level sequence information can be captured appropriately.

End-to-EndTreeLSTM utilises full structural information of the document. At first, it constructs an EDU embedding by composing word embeddings with a tree LSTM built on the constituency parse tree of the EDU. After that, it makes a document embedding with the same approach of DaNN. We employ a special relation type ‘CONSTITUENT’ to denote the relation among the constituents resulting from the constituency parsing. Note that no discourse roles are assigned to the nodes in the constituency parse tree since the atomic unit of discourse analysis is EDU. In the course of computing EDU embeddings, the nodes of the constituency parse tree inherit the weight matrices from the original EDU’s discourse role. In effect, this baseline allows for examining the role of the Paragraph Vector method by replacing the EDU embeddings trained with Paragraph Vector to the constituency parsing–based tree LSTM substitutes.

AvgEDUVec and LSTMEDUVec take the same settings of AvgWordVec and LSTMWordVec, respectively, except they utilise EDU embeddings instead of word embeddings. The same EDU embeddings used for the evaluation of DaNN in Sections 4.1 and 4.2 are used for these baselines.

The last baseline, Paragraph Vector, learns a document embedding directly from the text without any structure imposed on it. It is employed to compare the former does not need a composition method for weaving embeddings of smaller text segments. We train document embeddings according to the same experimental settings used for learning EDU embeddings and employ an SVM classifier for the classification tasks, similar to the evaluation method proposed in the original paper [11].
4.4.2. Results. The first lesson to learn from the result in Table 17 is that considering the structural information for the composition process leads to a better modelling of documents as in the comparison between \textit{AvgEDUVec} and \textit{AvgWordVec}. The big performance margins between the two approaches on all data sets make it clear that learning with EDUs first is more effective than learning with plain words. Put differently, words are not as good a unit as sentences in modelling a document. Modelling entire documents at once may be too coarse to handle the sophisticated changes of thematic flows in the text. The same conclusion can be drawn from the fact that DaNN shows better performance than \textit{End-to-EndTreeLSTM} and \textit{Paragraph Vector} baselines.

Nonetheless, \textit{End-to-EndTreeLSTM} gives the strongest result among those without using EDUs when all the three data sets are considered. The great performance of \textit{End-to-EndTreeLSTM} and DaNN strongly suggests that utilising discourse structure on composition-based learning is an effective approach. While \textit{End-to-EndTreeLSTM} seems to have a capability of learning the structure of a document, it suffers from the vanishing gradient problem in the backpropagation of the training stage because its hierarchy becomes very deep as it learns from the word level to a document level. We believe that this resulted in the performance worse than DaNN.

The second observation is that \textit{LSTMWordVec}, the word composition-based method using LSTM, as well as \textit{End-to-EndTreeLSTM}, has an advantage over \textit{AvgWordVec} and \textit{Paragraph Vector} that are both context-based methods. In other words, computing document embeddings with a thoughtful composition of embeddings of smaller text units has an advantage over learning an abstraction of document at once. It appears that the averaging method discards the characteristics of the individual units and the way they are laid out in a document.

It is interesting to note that \textit{LSTMEDUVec} results in a very poor performance, although it seems to be an ideal combination of LSTM and EDU. A deeper analysis shown that since sentences are more unique than words in the training data, a typical LSTM is expected to work poorly with sentences. The training set is overly sparse without repetitions of sentences. It would be difficult to capture the patterns among sentences with such a sparse data, unlike the way LSTM works with words.

An advantage of the proposed two-stage approach lies in computational efficiency in addition to the capability of avoiding the vanishing gradient problem. While the performance of \textit{End-to-EndTreeLSTM} is close to the ones using EDUs (DaNN and \textit{AvgEDUVec}), it suffers from a space problem. In our experiment, it was not possible to train \textit{End-to-EndTreeLSTM} via general-purpose computing on graphics processing unit (GPGPU), which usually improves the training speed of a DNN-based model dramatically. For example, we received the result of \textit{End-to-EndTreeLSTM} on the Cornell data set in 15 days with a 128G RAM computer, which was a solution to the failure of training the model with GeForce GTX Titan X (VRAM size: 12G). In addition, the \textit{End-to-EndTreeLSTM} result had to be obtained after two epochs using the Stanford data set because it always produced a memory error afterward even on the high-memory machine. From these experiences, it is safe to conclude that \textit{End-to-EndTreeLSTM} is not practical enough to be used in applications.

5. Conclusion and future work

We introduced a new method for modelling documents based on discourse-aware tree-structured neural networks, where a distributed representation of a document is built with its discourse information, including discourse structure, discourse role and discourse relation type. The main motivation is to make a document representation reflect upon its thematic structure and hence become more amenable to text classification tasks such as sentiment analysis and sarcasm detection. As such, we conducted experiments on the tasks to validate the proposed method. The results show that the proposed tree-structured method effectively incorporates the discourse information into an effective distributed representation to give substantial improvements for the tasks. The classification accuracy using the document model trained by the proposed method is better than that of the state-of-the-art sentiment analysis using discourse information, which even requires a pre-constructed sentiment lexicon. The proposed method also outperforms the state-of-the-art sarcasm detection approach that utilises textual features. It is worth noting that the proposed DaNN has a potential for even more improvements with the progress of the underlying techniques, for example, discourse parsing and embedding learning of arbitrary length text.

An important take-away message from this work is that the discourse information is an important contributor to modelling documents, especially on tasks that need high-level understanding of the thematic flow of the text, such as sentiment analysis and sarcasm detection. The detailed analysis of various alternatives in modelling text reveals that it is critical to set up a hierarchical structure for both the data and the learning mechanism and yet control the depths of the hierarchy, not to lose the benefit and the computational viability. Most importantly, this work suggested a novel way of incorporating discourse information into DNN-based text modelling and opens up a new research direction.

While the current work shows the great potential of the proposed approach, its validation can be extended to new application tasks, other than just sentiment analysis and sarcasm detection, in such a way that the fine-level discourse information can play a more distinctive and better understood role. For such a novel analysis with new functionality, we plan to design experiments for a more suitable task, construct our own data set for evaluation and show the availability.
of the proposed method as an immediate future work. Concurrently, we will extend the proposed method to support multi-label classification for added generalisability. On the other hand, for a more sophisticated use of the proposed method, we also need to ensure that the underlying discourse analysis technique is better tuned with less errors, in addition to more rigorous analysis of the nature of the errors and their impact on the test results, which require non-trivial linguistic knowledge.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding
This work was supported by ICT R&D program of MICT/IITP. [2013-0-00179, Development of Core Technology for Context-aware Deep-Symbolic Hybrid Learning and Construction of Language Resources].

Notes
1. https://github.com/TheEmancipator/DaNN
5. https://keras.io/

References


