

Structure Tree-LSTM: Structure-aware Attentional Document Encoders

Khalil Mrini	Claudiu Musat, Michael Baeriswyl	Martin Jaggi
Computer Science and Engineering	AI Group	Machine Learning and
University of California, San Diego	Swisscom AG	Optimization Lab, EPFL
La Jolla, CA 92093	Lausanne, Switzerland	Lausanne, Switzerland
KhalilMrini@ucsd.edu	firstname.lastname@swisscom.com	martin.jaggi@epfl.ch

Abstract

We propose a method to create document representations that reflect their internal structure. We modify Tree-LSTMs to hierarchically merge basic elements like words and sentences into blocks of increasing complexity. Our *Structure Tree-LSTM* implements a hierarchical attention mechanism over individual components and combinations thereof. We thus emphasize the usefulness of Tree-LSTMs for texts larger than a sentence. We show that structure-aware encoders can be used to improve the performance of document classification. We demonstrate that our method is resilient to changes to the basic building blocks, as it performs well with both sentence and word embeddings. The *Structure Tree-LSTM* outperforms all the baselines on two datasets when structural clues like sections are available, but also in the presence of mere paragraphs. On a third dataset from the medical domain, our model achieves competitive performance with the state of the art. This result shows the *Structure Tree-LSTM* can leverage dependency relations other than text structure, such as a set of reports on the same patient.

1 Introduction

Humans use structure to better represent information, and within that structure, elements vary in importance. For example, a table of contents helps in defining local context and, within the local context, focus the reader’s attention to what matters.

Long, unstructured sequences are hard to process for humans and machines alike. Even though neural network techniques have recently shown significant improvement to text classification, Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) perform poorly for long sequences (Cheng et al., 2016). Proposed solutions to overcome the problems that stem from flat representations include

LSTM variants such as the bidirectional variant (Graves and Schmidhuber, 2005; Huang et al., 2015; Chiu and Nichols, 2015) and the attention-based one (Wang et al., 2016). The latter focuses on relevant sections independent of their location. Flat attention cannot, however, cope with long sequences. Splitting a long text into smaller sections has the advantage of being able to flush the attention when the local context ends. A first step in this direction has been taken by Yang et al. (2016), who apply attention for words, and sentences as well, but documents are still viewed as a flat sequence of sentences.

Our first contribution is to adapt tools this far used only in the context of a single sentence, the tree-structured LSTM networks (Tai et al., 2015; Le and Zuidema, 2015; Zhu et al., 2015), to fit the document structure instead, therefore leveraging the principle of compositionality (Frege, 1884) to hierarchically combine embeddings in a document context.

Our second contribution is to extend previous attention mechanisms to all the structural levels of a document, and make them applicable in a hierarchical structure. To do so, we apply different initialization mechanisms to our Tree-LSTM model, leading to a transformation of the LSTM forget gates into a *de facto* attention mechanism.

By including hierarchical structure in the document representation and leveraging several layers of attention, we create **structure-aware attention-based document encoders**.

Our third contribution is to show that the structure-aware encoders are useful. We choose the task of supervised multi-class document classification as a first target, where each document has to be assigned to exactly one category. We first show that hierarchical documents can be better classified using tree-structured LSTMs. We obtain **improved document classification results**

over two datasets with varying document structure depths.

Then we show that our model can leverage structure beyond a single document, in settings where a training sample is a set of related documents. On a benchmark dataset from the medical domain, we model patients using their health reports to predict mortality. We obtain competitive results with the state-of-the-art method, emphasizing that our model can **efficiently model dependency relations other than simple textual structure**.

Finally, we also release our code along with our new dataset of hierarchical documents.

The rest of the paper is organised as follows: we summarize the literature on document embedding and classification in Section 2. Section 3 describes the proposed method and Section 4 details the experimental setup and the results. We draw conclusions in Section 5 and outline suggestions for future work.

2 Related Work

2.1 Text Embeddings

The use of vectorial word embeddings has become ubiquitous in applications of natural language processing. Among the most popular models are word2vec embeddings (Mikolov et al., 2013a,b), FastText (Bojanowski et al., 2016) and GloVe (Pennington et al., 2014). All three are built using word co-occurrence, and are widely used in their 300-dimensional form.

More recently, embedding methods for sentences have emerged. Zhao et al. (2015) introduce *AdaSent*, a self-adaptive hierarchical sentence model that represents short sequences using a multi-scale acyclic directed graph. Kiros et al. (2015) extend the skip-gram model to sentences to produce the *Skip-Thought* vectors. Their unsupervised pre-trained models produce either 2400- or 4800-dimensional vectors. The RNNs with GRU that are used in training are computationally expensive, something that Hill et al. (2016) try to address with their *FastSent* model. Arora et al. (2016) introduce an unsupervised method for sentence embedding they describe as simple: they compute a sentence’s embedding as the average of the embeddings of its words, minus the first principal component. Conneau et al. (2017) propose a supervised sentence embedding model trained on the Stanford Natural Language Inference dataset

(Bowman et al., 2015). Finally, the *sent2vec* 700-dimensional vectors (Pagliardini et al., 2017) are trained in an unsupervised manner, and represent sentences by looking at unigrams, as well as n-grams that compose them, in a similar fashion to FastText.

2.2 Document Encoders

Yang et al. (2016) introduce a hierarchical document encoder, used as a document classification model. It takes into account the compositionality of a document at sentence and word level. At each level, it applies an encoder with a bidirectional GRU and an attention mechanism. As a higher level structure is not used, they tested the method on shorter documents including reviews and answers.

2.3 Tree-structured LSTM networks

Tai et al. (2015) generalise the standard LSTM to adapt it to tree-like structures to create the *Tree-LSTM* architecture. They define two kinds of Tree-LSTMs. The first one is the Child-Sum variant. Here, for a unit j , the hidden variable h_{j-1} , that would be carried on from the previous LSTM unit $j - 1$ in a standard architecture, is replaced by the sum of the hidden variables of its children units $\tilde{h}_j = \sum_{k \in C(j)} h_k$, with $C(j)$ being the children units of unit j . In addition, there is one forget gate f_{jk} per child k of unit j . The parameter matrices enable the unit to determine the contributions of its children units in each gate. The second variant is the N -ary Tree-LSTM, where each non-leaf unit should have a branching factor of at most N and have ordered children. This variant allocates one parameter matrix per child, enabling it to learn conditioning based on the child’s position from 1 to N . However, it is not as modular as the Child-Sum Tree-LSTM as there is a constraint on the branching factor. Any Tree-LSTM unit still has to get an input x , whether it is a leaf unit or not.

Le and Zuidema (2015) develop the *LSTM-RNN* model, a binary tree-structured LSTM architecture, such that each non-leaf unit has exactly two children, with the corresponding pairs of input and forget gates. Zhu et al. (2015) introduce a similar binary tree-structured LSTM architecture, the *S-LSTM* model, in which there is one input gate per non-leaf unit, but still two forget gates. A non-leaf unit in these two architectures does not have an input of its own, but it takes the *inputs* of its children

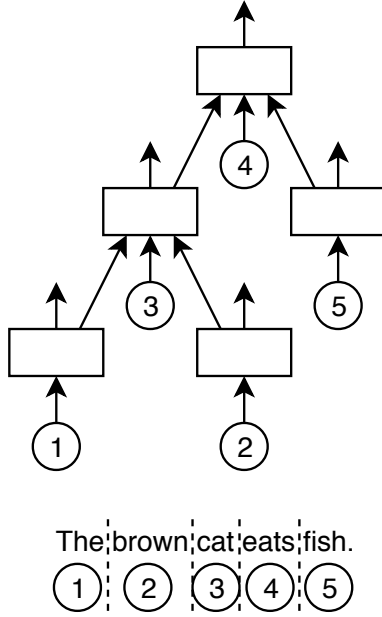


Figure 1: Example of a sentence encoding using the *Dependency Tree-LSTM* of Tai et al. (2015). Each LSTM unit receives a word embedding as input. This sentence’s reordering is based on its dependency tree.

units for the *LSTM-RNN* model, and their *outputs* for the *S-LSTM* model.

All three models are tested on sentiment classification of sentences using the Stanford Sentiment Treebank (Socher et al., 2013). This treebank’s labels are available as a fine-grained 5-way classification and as a binary classification. In the former, the *N*-ary Tree-LSTM model comes first in accuracy (51.0%), followed by the S-LSTM model (50.1%) and the LSTM-RNN model (49.9%).

In addition to sentence-level sentiment classification, Tai et al. (2015) test their model on semantic relatedness between sentence pairs. Eriguchi et al. (2016) extend the Tree-LSTM model to introduce a tree-to-sequence attentional neural machine translation model. Chen et al. (2017) use it in conjunction with the Bidirectional LSTM to form a hybrid model for natural language inference.

3 Structure-aware Attention-based Document Encoders

3.1 Structure Awareness

The starting assumption is that common documents have a hierarchical structure. Words are grouped in sentences, sentences in paragraphs, which in turn form subsections, sections and so on. From this observation, we derive the hypothesis

that hierarchical attention over a document’s structure allows the resulting representation to highlight the document’s important aspects.

Our *Structure Tree-LSTM* captures a document’s hierarchical structure by mirroring the corresponding *document tree*. For example, the document tree in Figure 2 corresponds to a document with the following outline: (1) *Introduction*: 1 paragraph with 3 sentences; (2) *History*: 2 subsections; (2.1) *19th Century*: 2 paragraphs, with 2 and 1 sentences respectively; (2.2) *20th Century*: 1 paragraph with 2 sentences. The structure granularity can be adjusted according to the downstream task and size of the dataset. Large datasets can have coarse-grained structure for the model to be less computationally expensive, whereas smaller datasets can have Tree-LSTMs include components all the way down to words.

The first major difference with respect to the existing Tree-LSTM, is that, as seen in the example, there is no imposed order on the semantic components. The *Dependency Tree-LSTM* of Tai et al. (2015) uses sentence-level word dependencies, as in the example in Figure 1. This is generally not extensible at the document level. Our *Structure Tree-LSTM* relaxes this assumption, making it more generally applicable.

The second major difference is the distinction between leaf and non-leaf units in a *Structure Tree-LSTM*. A leaf unit is the smallest component of the document: a word or a sentence in most cases. They have as input the component’s embedding. A non-leaf (parent) unit represents a larger component of the document: a sentence (group of words), a paragraph (group of sentences), or a section (group of sections and/or paragraphs). This generalization of the node contents allows for the extension of the method to more general contexts. In the original models of Tai et al. (2015), all units of a Tree-LSTM represent an original input (i.e., a word).

This distinction between unit types leads to the creation of two variants of *Structure Tree-LSTMs*, that differ in the strategy for filling the non-leaf units. We investigate two main strategies:

(1) **Structure Tree-LSTM with Zero Vectors**: non-leaf units have zero vectors as input (Figure 3).

(2) **Structure Tree-LSTM with Hierarchical Average**: non-leaf units have as input the average of the input vectors of its children (Figure 4).

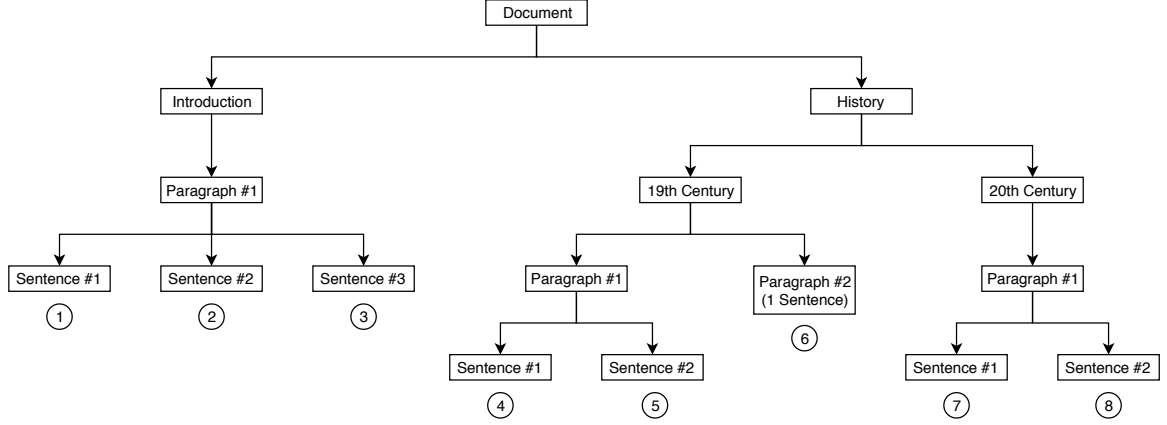


Figure 2: The *Structure Tree* corresponding to a document. It is the basis to form the document’s Tree-LSTM.

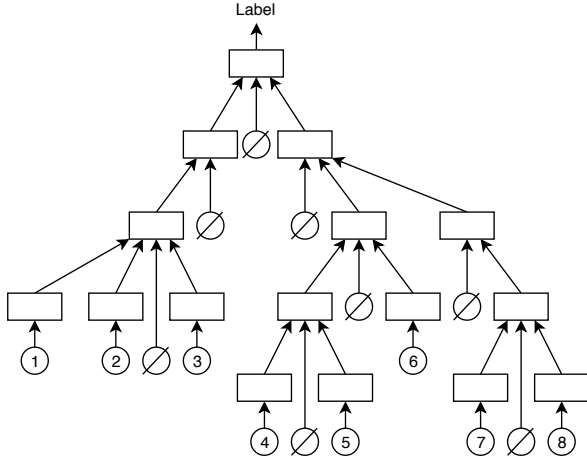


Figure 3: The *Structure Tree-LSTM with Zero Vectors* applied to the document in Figure 2, with the numbers in circles referring to the same input. The empty set symbols indicate a vector of zeros.

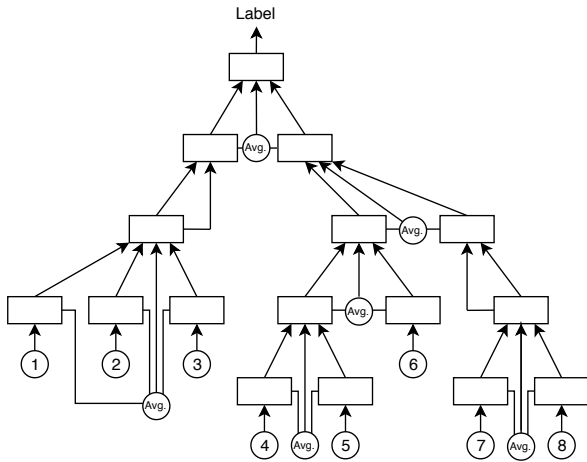


Figure 4: The *Structure Tree-LSTM with Hierarchical Average* applied to the document in Figure 2, with the numbers in circles referring to the same input. A given unit’s input is the average of the inputs of its children.

The *Structure Tree-LSTM* is easily extensible with additional initialization methods. One such method could be replacing the hierarchical averaging by a sum of the children’s inputs. Another one could be using section titles as input for the non-leaf units representing sections. This underlines the power of the *Structure Tree-LSTM* to incorporate all the information available. However, for the sake of fairness in comparison with the baselines, we did not use section titles in any of our models or baselines.

3.2 Hierarchical Attention

We use the same transition equations as in the Child-Sum Tree-LSTM described in [Tai et al. \(2015\)](#). We analyse them to explain the attention mechanisms of our proposed models.

For a unit j of a Child-Sum Tree-LSTM, the hidden state h_{j-1} , that is carried on from the previous LSTM unit $j - 1$ in a standard architecture, is replaced by the sum of the hidden states of its children units $\tilde{h}_j = \sum_{k \in C(j)} h_k$, with $C(j)$ being the children units of unit j . In addition, there is one forget gate f_{jk} per child k of unit j .

However, as the leaf units have no children units, we have that $|C(j)| = 0$, and as such $\tilde{h}_j = 0$. Therefore, the only contribution comes from the input (the word or sentence embeddings), without influence from other inputs. This changes the equations in practice, as for example the formula for the *input gate*:

$$i_j = \sigma \left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)} \right) \quad (1)$$

becomes for leaf units:

$$i_j = \sigma \left(W^{(i)} x_j + b^{(i)} \right) \quad (2)$$

The model with **Zero Vectors** *de facto* changes the formulas for the non-leaf units as well. Because a non-leaf unit's input is $x_j = 0$, the only contribution comes from the children units. This makes the *Structure Tree-LSTM* with zero vectors similar to a joint hierarchical attention mechanism. The formula for the *forget* gate for child unit k :

$$f_{jk} = \sigma \left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right) \quad (3)$$

becomes for non-leaf units:

$$f_{jk} = \sigma \left(U^{(f)} h_k + b^{(f)} \right) \quad (4)$$

making the *forget* gate an attention mechanism over individual child units. Likewise, the formulas of the *memory cell* c_j , the *input gate* i_j and *output gate* o_j change in practice. For example, the formula for the *output gate*:

$$o_j = \sigma \left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)} \right) \quad (5)$$

becomes for non-leaf units:

$$o_j = \sigma \left(U^{(o)} \tilde{h}_j + b^{(o)} \right) \quad (6)$$

such that the *output gate* can now be assimilated to an attention over linear combinations of individual children.

We can thus view the model with **Zero Vectors** as a generalization of hierarchical attention mechanisms.

4 Experiments

4.1 Experimental Setup

We design three experiments for the *Structure Tree-LSTM*. The datasets are split in 80% training, 10% validation for the model selection, and 10% testing.

To evaluate our model, we focus the analysis on datasets having a hierarchical structure. Therefore, we could not use the datasets of reviews on which the Hierarchical Attention Networks (Yang et al., 2016) are evaluated.

In the datasets of reviews in Yang et al. (2016), a training sample is only one paragraph with about 5 to 14 sentences on average. In our three selected datasets, we keep all information on internal structure: where each document part (paragraphs, sections, subsections...) starts and ends. We therefore

evaluate our models on datasets with at least three levels of hierarchy, with the highest hierarchy level being larger than a paragraph. More concretely, the highest hierarchy level in our datasets are an article, an email or a patient's medical record.

We evaluate the first two datasets using the Macro-F1 score, computed by averaging the individual F1 scores of each class. These first two datasets are about multi-class document classification, and the Macro-F1 score takes into account class imbalance.

The last dataset is evaluated using the Area under the ROC Curve or AUC (Hanley and McNeil, 1982), the same metric used in the benchmark baselines of Grnarova et al. (2016). This dataset is about predicting in-hospital patient mortality.

4.2 Text Structure in Document Classification

4.2.1 Baselines

For the first two datasets, we compare the *Structure Tree-LSTM* with the following baselines:

(3) MLP with Unweighted Average: a Multi-Layer Perceptron (MLP) with one hidden layer having a rectifier activation function. In this model, a document is represented by an unweighted average of all input embeddings.

(4) MLP with Hierarchical Average: same architecture as model (3), but the document representation is a hierarchically-weighted average of input embeddings, computed with depth-first search.

(5) Sequential LSTM: an LSTM layer that takes input embeddings in sequential order. The sets of input embeddings are not padded nor truncated. This model shares the same input as the Tree-LSTM, but processes them sequentially rather than hierarchically.

(6) Hierarchical Attention Networks: the model¹ designed by Yang et al. (2016). To remain faithful to the original model, this model is only tested on experiments with **word embeddings** as inputs. This model uses bidirectional GRUs, making it a hierarchical bidirectional RNN model. The model's hierarchy has two levels: one for words and one for sentences. Each level has separate encoders and attention weights, as well as a fixed number of elements. This means there is a fixed number of words (resp. sentences) per sentence

¹Code: <https://github.com/EdGENetworks/attention-networks-for-classification>

Model	Number of Parameters to Learn
Structure Tree-LSTM	$4(eh + h^2 + 2h) + hl + l$
Sequential LSTM	$4(eh + h^2 + 2h) + hl + l$
MLP	$eh + h^2 + 2h + hl + l$
Hierarchical Attention Networks (HAN)	$12(eh + h^2 + 2h) + w + s + sl + l$

Table 1: Number of parameters to learn for each model in the document classification datasets. e is the embedding dimension, h is the hidden layer dimension, l is the number of labels. For the HAN model, w is the number of words per sentence and s is the number of sentences per document.

(resp. document), and as such padding or truncating are applied where necessary.

The hidden layer dimension is 128 for the *sent2vec*-based experiments, and 64 for the *word2vec*-based experiments.

We compare the number of parameters for each model in Table 1. We take into account the hidden layer, as well as the *softmax* output layer. Our *Structure Tree-LSTM* models have as many parameters to learn as a sequential LSTM. Ignoring the output layer and the HAN attention weights, an MLP model has about 4 times less parameters to learn than an LSTM model, and the HAN model has about 3 times more parameters.

4.2.2 The Enron Email Dataset

This UC Berkeley-labelled dataset² contains 1,700 tagged emails with many overlapping categories. We select the two most common categories, and assign a third label for their intersection, and a fourth label for the emails that do not belong to any category. These real-world documents present meaningful, yet minimalist structure expressed as paragraphs.

The results in Table 2 show that the *Structure Tree-LSTM* model with zero vectors obtains the highest F1 scores, outperforming all other models. Moreover, the model’s performance is resilient regardless of the kind of building block the leaves represent – words or sentences. The extreme difference in macro-F1 scores with respect to the base LSTM underlines the importance of structure when fewer data points are available. We note the lacklustre macro-F1 of the Hierarchical Attention Networks, that suggests our structure-oriented attention model requires less data to train.

4.2.3 The Wikipedia Dataset

We validate these results with a second set: an English Wikipedia dataset of 494,657 articles. These

are relatively long articles, split into 24 disjoint categories and released as an open resource. More information is available in Appendix A. We evaluate this dataset with *sent2vec*, as it gave a higher macro-F1 score than *word2vec* for the previous experiment.

In Table 2, the two *Structure Tree-LSTM* models score visibly higher in both macro-F1 and accuracy than the baselines, confirming the efficiency of a document embedding inclusive of structure. The absolute gain is higher than for the previous dataset, suggesting that our models successfully leverage the additional structure in Wikipedia articles. In all cases, the best *Structure Tree-LSTM* variant is the one with zero vectors, showing that attention over children units is sufficient.

We analyse the classification errors of the *Structure Tree-LSTM* with Zero Vectors. The predictions of the *actors* category are correct 83.12% of the time. The two most common incorrect predictions for *actors* are *actresses* (3.89%) and *directors* (2.82%). Although our model is correct most of the time, its errors seem to stem from confusing semantically related categories. Other examples include the *airlines* category, with 85.54% accuracy and most commonly confused with *aircraft* (12.90%); and the *political parties* category, with 85.09% accuracy, and most commonly confused with *politicians* (11.69%).

4.3 Extended Structure: Modelling Sets of Documents

Textual structure is not the only way to represent the relations between the knowledge contained in a dataset. The types of documents and the links between the them can represent additional knowledge. We use a dataset from the medical domain, where the documents are medical reports of different categories. The reports can be grouped by patient to form sets of documents modelled as one *Structure Tree-LSTM*. By comparing to baseline

²Collected from: http://bailando.sims.berkeley.edu/enron_email.html

Dataset	Leaves	Model	Macro-F1	Accuracy
Enron Emails	Word Embeddings (<i>word2vec</i>)	Structure Tree-LSTM with Zero Vectors	0.4455	0.5235
		Structure Tree-LSTM with Hierarchical Average	0.4099	0.4824
		MLP with Unweighted Average	0.4063	0.4529
		MLP with Hierarchical Average	0.3934	0.4941
		Sequential LSTM	0.3429	0.4176
		Hierarchical Attention Networks	0.3632	0.5078
	Sentence Embeddings (<i>sent2vec</i>)	Structure Tree-LSTM with Zero Vectors	0.4533	0.5118
		Structure Tree-LSTM with Hierarchical Average	0.4278	0.4941
		MLP with Unweighted Average	0.4164	0.4588
		MLP with Hierarchical Average	0.3822	0.4706
Wikipedia	Sentence Embeddings (<i>sent2vec</i>)	Sequential LSTM	0.3002	0.4059
		Structure Tree-LSTM with Zero Vectors	0.8538	0.8877
		Structure Tree-LSTM with Hierarchical Average	0.8430	0.8814
		MLP with Unweighted Average	0.7870	0.8534
		MLP with Hierarchical Average	0.7790	0.8476
		Sequential LSTM	0.6405	0.7802

Table 2: Results of the multi-class classification experiments. The best scores are in bold.

models, we inquire whether the text structure, as extracted with *Structure Tree LSTMs*, can be useful in this extended knowledge framework.

4.3.1 The MIMIC-III Dataset

To compare our model to existing benchmark datasets, we use the MIMIC-III dataset (Johnson et al., 2016). It is a freely accessible database on critical medical care.

It contains over 2 million unstructured textual medical reports corresponding to 46,520 hospitalised patients, and information about whether a patient has eventually recovered. In case the patient died, it is indicated whether the death occurred at the hospital, within one month after leaving hospital care, or within a year afterwards.

Grunarova et al. (2016) use this dataset to predict patient mortality. They filter and pre-process the dataset according to the steps in Ghassemi et al. (2014). The resulting dataset contains 31,244 patients with 812,158 notes. Grunarova et al. (2016) approach this task as multiple binary classification problems: to predict mortality during the hospital stay, within one month later, or within a year later. We obtain the filtered dataset from Grunarova et al. (2016), and focus on in-hospital mortality prediction.

4.3.2 Baselines

Grunarova et al. (2016) devise a model using a word-level CNN for words within a sentence, and a sentence-level CNN processing each sentence

sequentially. They append the information about the corresponding medical report category as a vector to each sentence embedding.

They compare their CNN model to two baselines. The first one is the LDA-based Retrospective Topic Model (Ghassemi et al., 2014), a linear kernel SVM trained on the per-report topic distributions. This model is the state-of-the-art model for mortality prediction in the MIMIC-II dataset (Saeed et al., 2011). The second one is a linear SVM trained on doc2vec (Le and Mikolov, 2014) representations of the reports.

4.3.3 Target Replication

The authors also use Target Replication, following the approach of Lipton et al. (2015) and Dai and Le (2015). The intuition behind target replication is that the model learns better by replicating the loss at intermediate steps.

More formally, we add to the learning objective a cross entropy term $\ell(y_d^*, y_d^i | \vec{h}_i)$ for every intermediate step $i = 1, \dots, n$ of a training sample d , where y_d^* is the label associated to the corresponding training sample, y_d^i is the predicted label at the intermediate step i , and h_i is the hidden state of step i from which a softmax probability is computed. The loss function \mathcal{L}_d for the training sample d becomes:

$$\mathcal{L}_d = \ell(y_d^*, y_d | \vec{h}) + \frac{\lambda}{n} \sum_{i=1}^n \ell(y_d^*, y_d^i | \vec{h}_i) \quad (7)$$

Dataset	Leaves	Model	AUC score
MIMIC-III	Sentence Embeddings (<i>sent2vec</i>)	Structure Tree-LSTM with Zero Vectors	0.958
		LDA	0.930
		doc2vec	0.930
		CNN	0.963

Table 3: Results of the binary classification experiment in comparison with the baselines of Grnarova et al. (2016). The best scores are in bold.

In Equation 7, λ is a regularization parameter, y_d is the label predicted using the hidden state \vec{h} of the training sample d .

In the CNN model, target replication means predicting at each sentence and computing the corresponding loss. However, in our *Structure Tree-LSTM*, it means that we replicate the loss at *hierarchy levels*: we can predict at all units one level below the root (the root’s children), two levels (the children of the root’s children), or more. In this implementation, we use 1-level target replication.

4.3.4 Mortality Prediction Results

In addition to target replication, we follow the approach of Grnarova et al. (2016) and implement our *Structure Tree-LSTM* with categories encoded as one integer appended to sentence embeddings. In this experiment, the root node of the Structure Tree represents a patient, and its children units are the patient’s reports. Intuitively, we are replicating the loss at the *report nodes*. Each of the reports are divided into paragraphs and sentences. Similarly to the Wikipedia dataset, we use sentence embeddings for this experiment.

The training time for one epoch is over 24 hours. We choose to focus on the Zero-vector variant, as it performed best in the two first experiments. Our results are reported in Table 3. Our model only came 0.005 short of the CNN baseline, and beat the other two baselines.

This can be explained the difference between the CNN baseline and our *Structure Tree-LSTM* model: whereas the CNN baseline processes a patient’s reports in the temporal order in which they were issued by the hospital, our model does not incorporate this temporal information. This is important as there is a difference between a good health report coming after a bad health report (recovering patient) and the reverse situation (worsening health). More generally, the children of a Child-Sum Tree-LSTM unit are not processed in any sequential order, and as such sequential order is not preserved. Given that our model nonetheless

gives competitive results without temporal information, our future work will focus on modelling sequential order in Child-Sum Tree-LSTMs. It also indicates that our model can efficiently model dependency relations other than structure, such as a group of reports on the same patient.

Additional examples of applications of this ability include author or user modelling, using an author’s writings as the children of the root Tree-LSTM unit. Here, the dependency relation would be authorship.

5 Conclusion

To the best of our knowledge, the *Structure Tree-LSTM* is the first attempt to use Tree-LSTMs for texts larger than sentences. We show that our proposed structure-aware document encoders – the zero-vector variant – applies attention to all document structural levels. We apply the method for document classification and obtain an average of 9.00% relative improvement in macro-F1 score with respect to the the best baseline score. Moreover, the few errors the zero-vector *Structure Tree-LSTM* makes stem from confusing conceptually similar categories.

We also test this model on the MIMIC-III dataset, by modelling a patient as the root unit of the Tree-LSTM, and the corresponding medical reports as the children units. We obtain comparable results to the state of the art, coming only 0.005 short in AUC score. We hypothesize that the difference is because the Tree-LSTM cannot encode temporal order, but we note that it successfully modelled structure larger than a single document. This ability could have multiple practical applications, such as modelling people based on their writings.

Finally, we publish a novel document classification dataset of structured Wikipedia articles and release our code to encourage further research on long document encoders³.

³Links to be added in an updated version.

A Wikipedia Dataset Details

The Wikipedia dataset is collected from the English Wikipedia dump of February 1st, 2018⁴. We use a slightly modified version of the WikiExtractor⁵ to extract article from the unzipped .xml file.

The articles are filtered to have a certain length. To do so, we first compute the number of sentences, paragraphs and sections for each article. This is to get an idea of the average length of Wikipedia articles, and then to set a limit on them to filter out stubs. We check percentiles and decide to filter at 25% to get stubs out of our dataset. Practically, this corresponds to filtering out articles with less than 2 sections, 3 paragraphs and 5 sentences.

Afterwards, we get articles such that they belong to exactly one of 24 categories, and the numbers are detailed in Table 4. We determine the categories in the table by looking at keywords from the Wikipedia-tagged categories, not the articles themselves, and these categories are excluded from the body of the articles. The resulting dataset has 494,657 articles, and is released as an open resource⁶.

B Training Details and Hyperparameters

For all experiments, we used an Adam optimizer with a weight decay of $1e-4$ and a learning rate of $1e-2$. The batch size and hidden layer dimensions are respectively 64 and 128 for the sent2vec-based experiments, and 32 and 64 for the word2vec-based experiments. All models were trained using PyTorch.

We use 300-dimensional *word2vec* model pre-trained on the Google News Corpus⁷, and the 700-dimensional *sent2vec*⁸ model pre-trained on Wikipedia.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

⁴All Wikipedia dumps are freely available at <https://dumps.wikimedia.org/backup-index.html>

⁵The code used is available online at <https://github.com/attardi/wikiextractor>

⁶Dataset link to be added in the camera-ready version.

⁷Available at: <https://code.google.com/archive/p/word2vec/>

⁸Available at: <https://github.com/epfml/sent2vec>

Category	Number of articles
Actors	28 007
Actresses	22 208
Aircraft	12 278
Airlines	2 496
Artists	39 618
Cities	27 090
Comedy	14 680
Directors	19 218
Documentaries	3 848
Drama	20 523
Footballers	69 151
Horror	4 875
Journalists	15 363
Languages	6 779
Military Personnel	17 910
Musicians	17 603
Novelists	14 964
Novels	25 247
Political Parties	4 233
Politicians	56 130
Singers	17 055
Television	33 434
Video Games	20 059
Wars	1 888
Total	494 657

Table 4: Number of articles per category in the Wikipedia dataset used in the first experiment.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.
- Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.
- Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 823–833.
- Gottlob Frege. 1884. *Die Grundlagen der Arithmetik: eine logisch mathematische Untersuchung über den Begriff der Zahl*. w. Koebner.
- Marzyeh Ghassemi, Tristan Naumann, Finale Doshi-Velez, Nicole Brimmer, Rohit Joshi, Anna Rumshisky, and Peter Szolovits. 2014. Unfolding physiological state: Mortality modelling in intensive care units. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 75–84. ACM.
- Alex Graves and Jürgen Schmidhuber. 2005. Frame-wise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.
- Paulina Grnarova, Florian Schmidt, Stephanie L Hyland, and Carsten Eickhoff. 2016. Neural document embeddings for intensive care patient mortality prediction. *arXiv preprint arXiv:1612.00467*.
- James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.
- Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.
- Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.
- Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *Lexical and Computational Semantics (* SEM 2015)*, page 10.
- Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.
- Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. 2015. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. 2011. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.
- Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

- Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.
- Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*, pages 4069–4076.
- Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.