A Sequential Child-Combination Tree-LSTM Network for Sentiment Analysis

Name of author

Address - Line 1
Address - Line 2
Address - Line 3

Abstract

This paper describes a sentiment analysis model based on tree-structured long short-term memory (LSTM) neural networks. As a potential improvement over currently used approaches, we propose a sequential mechanism of combining network units representing syntactic dependencies, which provides an alternative way of preserving the compositionality of parent phrases as combinations of their syntactic dependents. We train and evaluate our system on a datasets made available in the PolEval 2017 competition. Our best result obtained with the proposed approach outperforms both basic, linearly combined LSTMs and a related tree-structured variant of such networks.

1. Introduction

Sentiment Analysis (SA) is an active area of natural language processing research with applications in product and political marketing, customer relations management systems and social media communication studies to mention just a few examples. A common objective of SA is to automatically detect the attitudinal value of an utterance or otherwise coherent stretch of text which can be attributed to a particular author. The distribution of such attitudinal values in an utterance is usually known as its *polarity* and it usually ranges from positive to neutral and negative (Cambria et al., 2013), possibly with finer-grained distinctions between these main categories, such as *somewhat positive* or *very negative* (Socher et al., 2013). More recent approaches to SA focus on phrase-level polarity classification, whereby attitudinal or emotional values are detected at the level of syntactic phrases and only then compositionally combined to compute the polarity of an entire utterance. Such approaches are particularly important in aspect-based SA, where different aspects of one's opinion about a product, movie, person, etc. have to be detected in addition to classifying the overall sentiment of a text unit. Also, phrase-level models of SA are better suited to deal with basic syntactic phenomena such as negation or modality, which may cause a significant shift in the predominant sentiment of an utterance (Wilson et al., 2009).

The present paper evaluates a deep-learning based system for syntax-driven SA, which was submitted to the PolEval 2017 competition. We first briefly describe the PolEval sentiment treebank (a syntactically annotated sentiment dataset) and the nature of the classification task. Next, we present the Sequential Child-Combination Tree-LSTM Network modelof the proposed system and evaluate its performance on the PolEval test set.

2. PolEval 2017

The dataset released in the PolEval competition for the task described in this paper can be described as a sentiment dependency treebank. In general, sentiment treebanks are collections of syntactically parsed sentences which have sentiment annotations assigned to their constituent words or phrases. The type of sub-sentential word combinations which are annotated with sentiment labels may depend on the exact syntactic formalism used in a given treebank. For example, the Stanford Sentiment Treebank (Socher et al., 2013) contains full constituency parse trees comprising phrases associated with five sentiment classes. On the other hand, the sentences in the PolEval dataset, are dependency-parsed. As we explain below, this distinction has some implications for modelling an SA system which makes use of syntactic annotations. For example, unlike the above-mentioned constituency representations, most sentence dependency graphs are N-ary trees of arbitrary length in that a word node may directly govern only one or more than two other word vertices as in the sentence shown in Fig. 1, where the verb *drażni* has four direct dependents (if we count the punctuation mark as a dependent of the sentence root), and the noun *dniu* has only one dependent adjective *całym*.

This example sentence[1] also illustrates the point of using syntactic annotation in sentiment analysis. It is quite evident that it expresses a customer's overall satisfaction with a particular brand of perfume, even though its (directly negated) main verb *drażnić* (irritate) could be found in a list of keywords which generally denote a negative sentiment. On a more subtle level, the positive attitude of this utterance towards this particular product is further reinforced by the indirectly negated adverbial clause (i.e. *even after a whole day*), which denotes a typical condition in which other fragrances might become irritating.

Table 1 illustrates the type of annotations made available in the PolEval 2017 sentiment analysis datasets. For each word in a sentence (listed in row `W`) its parent word node was indicated (see row `P`) to-

[1] Its word-for-word into English could be *(It) doesn't irritate even after a whole day.*

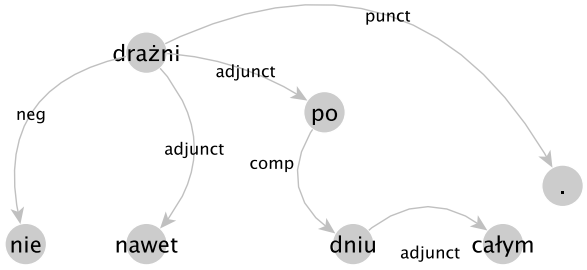

Figure 1: A dependency representation of the example sentence shown in Table 1.

gether with its sentiment label (as in row S). The parent labels could be used to reconstruct the syntactic dependency tree of this sentence as shown in Fig. 1. The goal of the PoleVal SA subtask was to a) correctly identify sentiment labels of each leaf word node and b) to correctly predict the sentiment label of every complete subtree starting in every non-terminal node of the sentence tree. The list of such leaf nodes and subtrees for which sentiment labels would have to be detected if the example sentence was found in the test set is shown in Table 2. The overall sentiment of an utterance is thus predicted from the sentiment value of its root node, which is usually the main verb of a sentence and the subtrees of the dependency trees considered in this task (other than the terminal word nodes) are self-contained phrases.

W	Nie	drażni	nawet	po	całym	dniu	.
P	2	0	2	2	6	4	2
S	0	1	0	0	0	0	0

Table 1: An example PolEval dataset sentence with phrase-level sentiment annotation (W – word nodes, P – syntactic parents, S – polarity value).

Word	Polarity
Nie	0
Nie drażni nawet po całym dniu .	1
nawet	0
po całym dniu	0
całym dniu	0
.	0

Table 2: Phrase- and word-level sentiment values to be predicted for the example sentence shown in Table 1.

3. The Model

3.1. LSTM Neural Networks

Different variants of Recurrent Neural Networks (RNNs) have been widely used for many task in natural language processing such named entity recognition (Lample et al., 2016) or text classification (Lai et al., 2015). RNNs can process arbitrarily long inputs by recurrent application of a transition function over hidden states. The most common form of an RNN transition function is an affine transformation followed by a hyperbolic tangent function:

$$h_t = tanh(W_{x_t} + U_{h_{t-1}} + b) \tag{1}$$

A potential advantage of RNNs in processing natural language stems from their ability to use information gathered sequentially from previous states, corresponding to units of language such as words or phrases when dealing with current input. In theory this makes them capable of tracking a long history of previous states when processing sequences of words. In practice, however, RNNs may they suffer from so-called vanishing gradient problem, which means that during training the gradient can grow or decay exponentially over long sequences (Bengio et al., 1994; Hochreiter, 1998). The LSTM architecture (Hochreiter and Schmidhuber, 1997) addresses the problem of learning long-term dependencies by introducing a memory cell that is capable of preserving previous state information over relatively long sequences of states.

While there are many different types of LSTMs, in our work we used a variant described by the following equations:

$$\begin{aligned} i_t &= \sigma(U^i x_t + W^i s_{t-1} + b^i), \\ f_t &= \sigma(U^f x_t + W^f s_{t-1} + b^f), \\ o_t &= \sigma(U^o x_t + W^o s_{t-1} + b^o), \\ g_t &= \tanh(U^g x_t + W^g s_{t-1} + b^g), \\ c_t &= c_{t-1} \bullet f_t + g_t \bullet i_t, \\ s_t &= \tanh(c_t) \bullet o_t \end{aligned} \tag{2}$$

where x_t is input at current time step, i_t is an input gate, f_t a forget gate, o_t an output gate and c stands for LSTM memory. The gating mechanism controls how much information from past states and memory is used at the current time step.

3.2. Tree LSTMs

One problem with applying classical LSTMs in natural language processing is that similarly to standard RNNs, they are linear chains, which makes them difficult to use directly for processing more sophisticated linguistic structures. For example, syntactic annotation which could be useful in predicting more subtle aspects of sentiment of the kind illustrated above usually takes the form of hierarchically-structured dependency or constituency trees rather then linear sequences of word segments. In order to process tree-structured data, the standard LSTM architecture has been elaborated into the so-called Tree LSTMs (Tai et al., 2015). In general, Tree-Structured Recurrent Neural Networks expand basic RNNs by creating a way to plug in more than one previous state to hidden unit at given time step as shown in Figure 2. Each

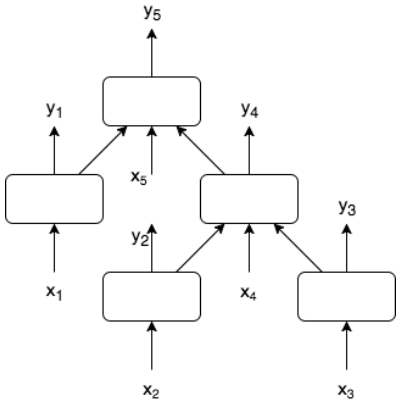

Figure 2: A Tree LSTM network architecture.

neural network unit takes an input x_j and emits a label y_j as in a standard RNN. Additionally, however, each such unit can take more than one previous hidden state. Tree-LSTMs also add their memory-handling capacities to this elaborated architecture.

3.3. Tree LSTM label classification

As already mentioned, in sentiment treebanks, sentences are represented as trees whose (selected) components are also labelled for sentiment. Generally, in a Tree LSTM network used for language processing tasks which can be described as label classification, each sentence component is a unit x_j represented by a distributional vector known as its *(word) embedding* (Mikolov et al., 2013). The hidden states of network units which represent the leaves of a syntactic sentence tree are embeddings of the corresponding words. Network units which represent non-terminal tree nodes, have hidden states representing the embeddings of the entire subtree which they happen to head. By computing an activation function on such hidden states, we obtain a sentiment label of an entire subtree. The hidden state of the root unit is the embedding of the entire sentence and its label is the label of the entire sentence.

Tree LSTM architectures may vary with respect to the exact way in which the child units are incorporated into their respective parents. In this paper we consider the so-called Child-Sum Tree-LSTM architecture (Tai et al., 2015) and present a new architecture, which we call a Sequential Child-Combination Tree-LSTM and evaluate its performance in syntax-based sentiment analysis. The Child-Sum Tree-LSTM model (ibid.) provides a way of representing dependency syntax relations between parent word nodes and their children and it is based on the following transitions equations:

$$
\begin{aligned}
s_{ch} &= \sum_k s_{ch_k}, \\
i_j &= \sigma(U^i x_j + W^i s_{ch} + b^i), \\
f_{jk} &= \sigma(U^f x_j + W^f s_{ch_k} + b^f), \\
o_j &= \sigma(U^o x_j + W^o s_{ch} + b^o), \\
g_j &= \tanh(U^g x_j + W^g s_{ch} + b^g), \\
c_j &= \sum_k c_k \bullet f_{jk} + g_j \bullet i_j, \\
s_j &= \tanh(c_j) \bullet o_j,
\end{aligned}
\tag{3}
$$

where k is a subscript of k-th child of node j and s_{ch_k} is its hidden state. There is no arbitrary limit on the branching factor of the syntactic trees to be represented in this architecture. Moreover, as explained by its authors, this model is capable of learning parameters W^i which 'open' the input gate i_j for semantically discriminative content words such as adjectives or verbs and 'close' it for low-information function word inputs such as determiners.

3.4. Sequential-Child-Combination Tree-LSTM

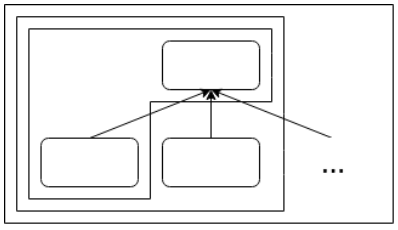

Figure 3: Sequentially combining children with their parents in an LSTM network.

The key contribution to LSTM-based models for sentiment analysis proposed in the present paper is what we call a Sequential Child-Combination Tree-LSTM architecture. Formally, in order to combine, children nodes with their parents, we follow the tran-

sition equations specified below:

$$
\begin{aligned}
i_{i_1} &= \sigma(U^i x_t + W^i s_{ch_1} + b^i),\\
f_{i_1} &= \sigma(U^f x_t + W^f s_{ch_1} + b^f),\\
o_{i_1} &= \sigma(U^o x_t + W^o s_{ch_1} + b^o),\\
g_{i_1} &= \tanh(U^g x_t + W^g s_{ch_1} + b^g),\\
c_{i_1} &= c_{t-1} \bullet f_{i_1} + g_{i_1} \bullet i_{i_1},\\
s_{i_1} &= \tanh(c_{i_1}) \bullet o_{i_1},\\
\\
i_{i_2} &= \sigma(U^i s_{i_1} + W^i s_{ch_2} + b^i),\\
f_{i_2} &= \sigma(U^f s_{i_1} + W^f s_{ch_2} + b^f),\\
o_{i_2} &= \sigma(U^o s_{i_1} + W^o s_{ch_2} + b^o),\\
g_{i_2} &= \tanh(U^g s_{i_1} + W^g s_{ch_1} + b^g),\\
c_{i_2} &= c_{i_1} \bullet f_{i_2} + g_{i_2} \bullet i_{i_2},\\
s_{i_2} &= \tanh(c_{i_2}) \bullet o_{i_2},
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
&\dots\\
\\
i_{i_n} &= \sigma(U^i s_{n-1} + W^i s_{ch_n} + b^i),\\
f_{i_n} &= \sigma(U^f s_{n-1} + W^f s_{ch_n} + b^f),\\
o_{i_n} &= \sigma(U^o s_{n-1} + W^o s_{ch_n} + b^o),\\
g_{i_n} &= \tanh(U^g s_{n-1} + W^g s_{ch_n} + b^g),\\
c_{i_n} &= c_{i_{n-1}} \bullet f_{i_n} + g_{i_n} \bullet i_{i_n},\\
s_{i_n} &= \tanh(c_{i_n}) \bullet o_{i_n},\\
\\
c_t &= c_{i_n}\\
s_j &= s_{i_n}
\end{aligned}
$$

where s_{i_k} is an intermediate state obtained after combining with its k-th child. That means that we can see each Tree-LSTM unit as a linear LSTM chain representing each child respectively with c as one global, linearly updated memory.

Even though dependency trees do not explicitly encode word order information, the proposed method of combining them allows for creating embeddings composed of subphrases or single word dependents of particular children and 'remember' them in the learning process, which is not directly possible in case of the Child-Sum Tree-LSTM architecture. Moreover, word order information can be preserved in our model by combining first the children that occur earlier in the sentence. The intuition behind this approach is to provide a mechanism of encoding the relations between the parent and its particular children in a way which is potentially more subtle and more discriminative than the Child-Sum Tree-LSTM model. To phrase it differently, we combine parents with their children from left to right by combining the intermediate embeddings of the roots of child phrases with the current parent phrase. The final result of this process is a compositional embedding of an entire sentence. This process is further illustrated in Fig. 3 and in Fig. 4, where it is applied to the example sentence from the PolEval sentiment treebank introduced above. Although it is possible to manage the memory of such a network model in a different way, for example by using a separate memory for each unit, our best results were obtained using the single global memory presented here.

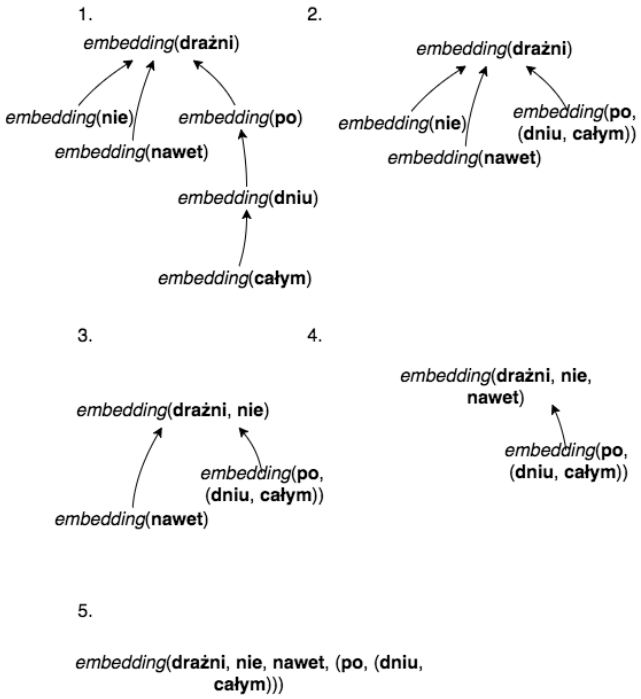

Figure 4: An example of combining dependent phrases with their parent in the Sequential Child-Combination Tree-LSTM model.

4. Evaluation

The model proposed in this paper was evaluated on the held-out test set provided in the PolEval 2017 competition. As baselines, we used a simple, linearly chained LSTM network, Conditional Random Fields (Lafferty et al., 2001) with rather standard contextual and word-shape features and no syntactic information as well as the Child-Sum Tree LSTM model mentioned above. The result submitted to the single-blind evaluation stage of the PolEval competition was approx. 76,80% accuracy of sentiment label assignments. In Table 3 we report the best results obtained on the same test set with a modification of our originally submitted system, which involved lemmatizing the word tokens in the training and test sentences. After this modification, our approach seems to have slightly outperformed the Child-Sum Tree model on the PolEval test set. It is interesting to note that some improvement resulting from lemmatization was consistent across the different approaches we tested.

5. Conclusions and future work

In the present paper we described a Sequential Child-Combination Tree-LSTM model for the task of syntax-based sentiment analysis in Polish. The theoretical motivation behind this model was to provide

Model	Score
LSTM	70,70%
Lemmatized CRF	75,63%
Child-Sum Tree-LSTM	77,69%
Sequential Child-Combination Tree-LSTM	**76,80**%
Lemmatized Child-Sum Tree-LSTM	79,35%
Lemmatized Sequential Child-Combination Tree-LSTM	**79,89%**

Table 3: Sentiment classification accuracy on the PolEval test set.

a relatively robust mechanism of considering syntactic relations between words and phrases in natural language sentences as a potentially relevant feature in sentiment detection. In particular, we proposed a mechanism of compositionally generating governor or parent-phrase embeddings from the embeddings of their dependent phrases, which preserves some of the compositionality of the latter type of units. The results obtained for the PolEval 2017 test set were promising, but given the rather limited size of these datasets, it would be necessary to run similar comparisons on larger sentiment treebanks, in order to validate the proposed approach of tracking the syntactic and semantic compositionality of sentences in SA. The method described in this paper can be applied to dependency-parsed data in other languages, possibly also in other tasks, such as measuring the semantic relatedness of sentence pairs, where syntax-driven deep learning approaches proved to be particularly successful (Tai et al., 2015).

6. Availability

The source code of the system described in this paper, together with its trained models are available at `https://github.com/ANONYMIZED`.

7. Acknowledgments

Research described in this paper was funded by XXX.

8. References

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi, 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Cambria, E., B. Schuller, Y. Xia, and C. Havasi, 2013. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2):15–21.

Hochreiter, Sepp, 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, Sepp and Jürgen Schmidhuber, 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira, 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lai, Siwei, Liheng Xu, Kang Liu, and Jun Zhao, 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333.

Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer, 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Socher, Richard, 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*.

Tai, Kai Sheng, Richard Socher, and Christopher D Manning, 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann, 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433.