A Sequential Child Combination Tree-LSTM Network for Sentiment Analysis

ML[†]
PęP[*]

[†]VoiceLab
michal.lew@voicelab.pl

[*]University of Lodz
piotr.pezik@uni.lodz.pl

Abstract

This paper evaluates thsdfh.

1. Introduction

Sentiment Analysis (SA) is an active area of natural language processing research with applications in general opinion mining, customer relations management systems, marketing 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). More recent approaches to SA focus on phrase-level polarity classification, whereby attitudinal 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 SA is 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). For example, the overall polarity of the Polish sentence shown in Table 1, which describes a buyer's overall satisfaction with a particular brand of perfume, is positive, even though its (directly negated) main verb *drażnić* (irritate) could be found in a list of keywords denoting negative sentiment.

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 syntactically annotated sentiment datasets used and the nature of the classification task. Next, we present the Sequential Child-Combination Tree-LSTM Network implementation of our system and evaluate its performance on the PolEval test set.

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).

2. PolEval 2017

3. Model

3.1. LSTM

Recurrent Neural Networks have been widely used for many task in natural language processing. They can process the arbitrary long inputs by recurrent application of transition function over hidden state. The most common form of RNN's transition function is an affine transformation followed by hyperbolic tangent function

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

This gives them ability to use the information gathered in previous states when processing text. In theory it makes them able to look at the infinite history while processing sequences of words. Unfortunately they suffer from so called vanishing gradient problem, which means that during training gradient can grow or decay exponentially over long sequences [s (Hochreiter, 1998; Bengio et al., 1994)] The LSTM architecture (Hochreiter and Schmidhuber, 1997) addresses this problem of learning long-term dependencies by introducing a memory cell that is able to preserve state over long periods of time. [citation] There are many different variants of LSTMs, in our work we used 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 timestep, i_t is an input gate, f_t a forget gate, o_t an output gate and c stands for LSTM memory. The gates mechanism controles how much information from past state and memory is used at current timestep.

3.2. Tree-LSTM

The problem with classical LSTM's is that they are linear chain which makes them useless for processing more sophisticated structures than sequences. In particular, many linguistic beings are represented with tree-structure [citation?]. In order to process tree-structured data we need to extend the LSTM architecture. In our work we propose Sequential Child-Combination Tree-LSTM. 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 current time step. [obrazek podłączanie 2 poprzednich stanów do nowego] The essence lie in the way in which the children states are combined with parent state.

3.3. Child-Sum Tree-LSTM

Child-Sum Tree-LSTM equations

3.4. Sequential-Child-Combination Tree-LSTM

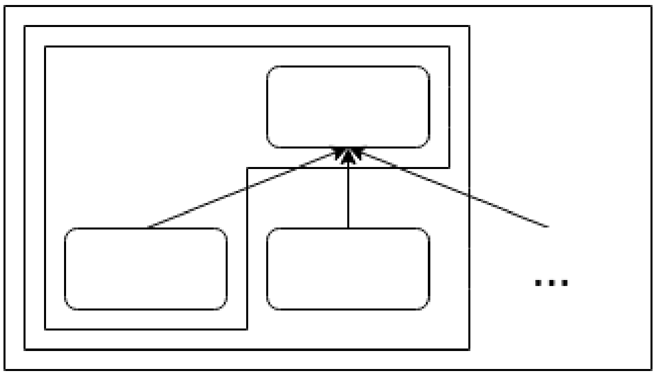

Figure 1: Sequentially combining the children with parent

In our work we combine children with its parent as follows:

$$\begin{aligned}
i_{ch_1} &= \sigma(U^i_{x_t} + W^i_{s_{t-1}} + b^i),\\
f_{ch_1} &= \sigma(U^f_{x_t} + W^f_{s_{t-1}} + b^f),\\
o_{ch_1} &= \sigma(U^o_{x_t} + W^o_{s_{t-1}} + b^o),\\
g_{ch_1} &= \tanh(U^g_{x_t} + W^g_{s_{t-1}} + b^g),\\
c_{ch_1} &= c_{t-1} \bullet f_{ch_1} + g_{ch_1} \bullet i_{ch_1},\\
s_{ch_1} &= \tanh(c_{ch_1}) \bullet o_{ch_1},\\
\\
i_{ch_2} &= \sigma(U^i_{x_t} + W^i_{ch_1} + b^i),\\
f_{ch_2} &= \sigma(U^f_{x_t} + W^f_{ch_1} + b^f),\\
o_{ch_2} &= \sigma(U^o_{x_t} + W^o_{ch_1} + b^o),\\
g_{ch_2} &= \tanh(U^g_{x_t} + W^g_{ch_1} + b^g),\\
c_{ch_2} &= c_{ch_1} \bullet f_{ch_2} + g_{ch_2} \bullet i_{ch_2},\\
s_{ch_2} &= \tanh(c_{ch_2}) \bullet o_{ch_2},\\
\\
&\ldots\\
\\
i_{ch_n} &= \sigma(U^i_{x_t} + W^i_{ch_{n-1}} + b^i),\\
f_{ch_n} &= \sigma(U^f_{x_t} + W^f_{ch_{n-1}} + b^f),\\
o_{ch_n} &= \sigma(U^o_{x_t} + W^o_{ch_{n-1}} + b^o),\\
g_{ch_n} &= \tanh(U^g_{x_t} + W^g_{ch_{n-1}} + b^g),\\
c_{ch_n} &= c_{ch_n} \bullet f_{ch_n} + g_{ch_n} \bullet i_{ch_n},\\
s_{ch_n} &= \tanh(c_{ch_n}) \bullet o_{ch_n},\\
\\
s_t &= s_{ch_n}
\end{aligned} \tag{3}$$

where s_{ch_i} is an intermediate state after combining with i-th children.
Intuicyjnie, łączymy dzieci z rodzicem od lewej do prawej w taki sposób, że tworzymy pośrednie zanurzenia niosące informację o ojcu uzupełnioną o informację o poddrzewie, którego korzeniem jest dane dziecko. Ostateczny stan po połączeniu ze wszystkimi dziećmi tworzy znaczenie całego drzewa. Przykład 'Stary rudy kot wlazł na płot' mamy kot-stary kot-rudy kot-wlazł z tego wychodzi, że stary kot-rudy stary kot-wlazł stary rudy kot-wlazł itd. można przedstawić taki przykład, żeby pokazać intuicję stojącą za tym sposobem.

4. Evaluation

5. Conclusions and future work

6. Availability

The source code of the tagger described in this paper, together with the trained models, datasets, tagset translations and other resources such as the NCP Brown clusters are publicly available at `https://gitlab.com/piotr.pezik/apt_pl`.

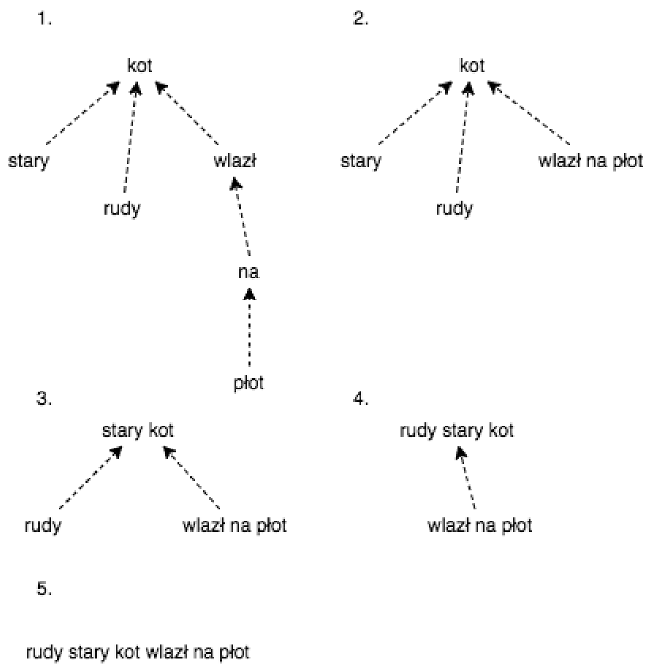

Figure 2: Przykład

Model	Score
LSTM	70,70%
Lemmatized CRF	75,63%
Child-Sum Tree-LSTM	77,69%
Sequential-Child-Combination Tree-LSTM	78,56%
Lemmatized Child-Sum Tree-LSTM	78,90%
Lemmatized Sequential-Child-Combination Tree-LSTM	**79,89%**

Table 2: Speed and accuracy evaluation of an AP-based ‘flexeme’ tagger.

7. References

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.

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.