Sentiment Analysis

ML[*]
PęP[†]

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

[†]VoiceLab
dupa@voicelab.pl

Abstract

This paper evaluates thsdfh.

1. Introduction

Sentiment analysis is one of fundamental task in natural language processing.

(De Marneffe et al., 2014).

2. Previous work

3. The Data

3.1. POLEVAL2017 dataset

It should be noted that the sample used to evaluate the tagger is rather small compared with the conventional test splits used in machine learning experiments. Nevertheless, using this test set esured some consistency of the reported accuracy scores with the results reported for the other systems submitted to PolEval.

4. Model

4.1. LSTM

Recurrent Neural Networks has 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: [równania LSTM] 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.

4.2. 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.

4.3. Child-Sum Tree-LSTM

Child-Sum Tree-LSTM equations

4.4. Sequential-Child-Combination Tree-LSTM

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

5. Evaluation

Number of tags (without *ign*)	35
Token accuracy	98.2%
Training set	PolEval TR
Test set	PolEval TE
Training time	5m 40s (5 iterations)
Tagging time	1.42s

Table 1: Speed and accuracy evaluation of an AP-based 'flexeme' tagger.

6. Conclusions and future work

7. 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`.

8. References

De Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning, 2014. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14.