Polish Language Sentiment Analysis with Tree-Structured Long Short-Term Memory Network

Norbert Ryciak

(1) Faculty of Mathematics and Information Science, Warsaw University of Technology
Koszykowa 75, 00-662 Warsaw, Poland
(2) Institute of Computer Science, Polish Academy of Sciences
Jana Kazimierza 5, 01-248 Warsaw, Poland
n.ryciak@mini.pw.edu.pl

Abstract

This article shows newly-established state-of-the-art results in sentiment analysis over Polish language. It is the first time to report the results of applying tree-structured neural network for sentiment analysis for Polish language. We present the outcome obtained on the first freely available Polish sentiment treebank dataset with fine grained labels. The dataset consists of sentences in dependency trees format with sentiment labels over all nodes of the tree which represent phrases. The task is to predict phrase-level sentiment - including whole sentences and single tokens. This is the winning solution of the task 2 of PolEval 2017 - SemEval-inspired competition for natural language processing tools for Polish.

1. Introduction

Sentiment analysis is one of the important aspects of natural language processing domain. It deals with detecting the sentiment of human-generated text. There are various variants of this problem. Firstly, there may be various study subjects: documents (long text), sentences or phrases. Secondly, there can be different outcomes expected - just positive/negative, fine-grained with miscellaneous precision or continuous describing strength of sentiment. In this article we study phrase-level sentiment analysis with the negative/neutral/positive scale.

There are many approaches to this problem and they are constantly improved. We can distinguish four groups of methods which can be viewed as stages of development with ascending complexity. The most simple solutions (working well only on simple data) are based on predefined dictionaries of sentiment-labeled words. Having specified the list of words with human labeled sentiment we determine text sentiment based on occurrence of these words. This can be also extended by adding some predefined heuristics improving the results for more complex texts.

The approach described above is far from satisfying in general. Thus, the natural development direction for sentiment recognition tools is machine learning (Melville et al., 2009). Further approaches treat sentiment prediction as classification task (or regression when we have continuous sentiment scale). First level of complexity is based on conventional classification. In this case we have to vectorize our data first. It means that we need to represent each text as a vector of features, so that we obtain representation matrix of set of texts. This is often crucial element of this approach - the way in which we create feature vectors strongly influence the results. The second step is finding a classifier which will perform best, which is not trivial either, although there are some common solutions in this aspect like Naive Bayes classifier or Support Vector Machines. These solutions are widely used in practice and often give satisfying results. However, they have many limitations and cannot cope with the more complex texts (Pang et al., 2002).

The next stage of development are methods making use of structure of the text. This is natural approach because it is apparent that the order of words is important for text meaning and, in particular, sentiment. Thus, instead of bag-of-words we treat sentences as a sequence of words and model the data using machine learning models dedicated for this kind of data. This is the case of structured-prediction learning problem, where the object to classify is a sequence of basic entities. However, there are two issues in this case. Firstly, these entities - words in text case - have to be represented as vectors. Secondly, we have to choose the appropriate model. Notwithstanding, in this case the choice is simpler than before - we can use deep learning models - the recurrent neural networks (RNNs) which are known to perform well in this kind of problem (Mikolov et al., 2010). RNN scans the words one after another with propagating forward information collected from already seen words. At the end of a sequence the network gives prediction. This model requires words represented as vectors. The standard choice for RNNs is word2vec model (Mikolov et al., 2013), as is for the next method.

Currently, the most sophisticated approaches utilize deeper look at the sentence structure - not only the order of the words, but also the structure in the sense of linguistic information (Socher et al., 2013). Specifically, we use tree-structured representation of sentences (or phrases). In sentiment analysis two kinds of trees are used - dependency and constituency and both of them can lead to comparable results of sentence sentiment classification (Tai et al., 2015). The models used in this approach belong to deep learning methods, and are the generalizations of the recurrent neural networks. Tree structured neural networks ”read” sentence along branches from leafs to the root transferring information from children to parents, in each node aggregating information from its subtrees. In the end, the network gives prediction for a sentence when it reaches the root. This method requires that words are represented as vectors and word2vec model is a commonly used solution.

In this paper we describe Tree-Structured Long Short-Term Memory Network (Tree-LSTM) (Tai et al., 2015) which performed best in the task of phrase-level sentiment analysis over Polish dependency sentiment treebank. This is the winning solution of sentiment analysis task at the PolEval 2017 competition[1]. We provide detailed description of the model and we evaluate its performance with comparison to other methods.

2. Tree-Structured Long Short-Term Memory Network

2.1. Long Short-Term Memory Network

Recurrent neural networks (RNNs) take sequences of arbitrary length as input, where elements of a sequence are represented as vectors. RNN takes entities sequentially and in each time step it uses the information gathered earlier. Formally, in time step t it calculates hidden state h_t as a function of current input vector x_t and the previous state h_{t-1}. The prediction for a sequence is calculated basing on hidden state of the last element (it can also give an output for each element of a sequence based on corresponding hidden state).

Long Short-Term Memory Network - LSTM (Hochreiter, 1998) is one of the most widely used representative of recurrent neural networks because of its effectiveness. The idea behind this model is to deal with the problem of catching dependencies among entities which are not close in a sequence, i.e. strengthen the network ability to recognize salient patterns in long sequences. Information is propagated by two channels - hidden state as usual and memory component. They are calculated with the use of sophisticated mechanism defined with following formulas:

$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right), \tag{1}$$

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \tag{2}$$

$$u_t = \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right), \tag{3}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1}, \tag{4}$$

$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right), \tag{5}$$

$$h_t = o_t \odot \tanh(c_t), \tag{6}$$

where $W^{(\cdot)}$ are matrix parameters - network weights, $b^{(\cdot)}$ are biases vectors and x_t is t-th input vector. Despite the complexity of the equations, all components have easily interpretable role. f_t is "forget gate layer" and it is a filter - it decides how to keep information from the past. It gives the features weights of their predicted importance. i_t called "input gate layer" is for filtering input data - it is responsible for determining which features and to what extent we want to use. Well learned network should be able to recognize importance of features. u_t transforms input in order to represent it in adjusted form for further processing. It takes into account information gathered earlier. c_t is memory cell - this is the "box" responsible for keeping all and only important information through a long time. It is calculated with the use of two elements: pieces of information stored in that moment and the new ones currently provided. Thus, it is the sum of transformed input filtered by input gate and previous memory state filtered with forget gate. This memory state is passed forward. o_t is again a kind of filter, which can be called "output gate". It is used to calculate hidden state h_t which is the transformation of newly established memory state with features multiplied by o_t. It is transmitted forward and to the output calculation for currently considered element at the same time.

2.2. Tree-LSTM

Tree-structured neural network is a kind of recurrent neural network designed for tree-structured data and is a generalization of recurrent neural networks (RNNs). It works on the principle of propagating information along branches of a tree. The algorithm begins in leaves and moves upwards aggregating information in each node from its subtrees. In the end, network reaches root of the tree and gives an output - prediction for considered tree-stuctured observation. Especially, Tree-Structured Long Short-Term Memory Network is the generalization of LSTM and it was introduced in (Tai et al., 2015).

As in standard LSTM information is propagated through two channels - memory cell c_t and hidden state h_t. However, the mechanism is adapted to tree topology. Thus, recursion in tree-LSTM is done through aggregation of information from all subtrees of currently visited node taking into account node's content. In case of dependency parse tree of sentence the node corresponds to a word. Formally, tree-LSTM unit is defined with the following formulas, where j is the node's index in tree and $C(j)$ is set of children of node j:

$$f_{jk} = \sigma\left(W^{(f)}x_j + U^{(f)}h_k + b^{(f)}\right), \quad \text{for } k \in C(j) \tag{7}$$

$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \tag{8}$$

$$i_j = \sigma\left(W^{(i)}x_j + U^{(i)}\tilde{h}_j + b^{(i)}\right), \tag{9}$$

$$o_j = \sigma\left(W^{(o)}x_j + U^{(o)}\tilde{h}_j + b^{(o)}\right), \tag{10}$$

$$u_j = \tanh\left(W^{(u)}x_j + U^{(u)}\tilde{h}_j + b^{(u)}\right), \tag{11}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{12}$$

$$h_j = o_j \odot \tanh(c_j), \tag{13}$$

where $W^{(\cdot)}$ and $b^{(\cdot)}$ are network parameters. Input x_j for current node is a word represented as vector - the natural choice is word2vec embeddings. As in standard LSTM we have a filter for memory but in tree-structured data we have many "forget gates" f_{jk} - all of them are calculated individually for each subtree (Eq. 7). Thus, based on current input x_j and subtree hidden state the network filters information from each branch in a special way. This solution allows treating each sub-phrase in individual way,

[1] http://poleval.pl/

passing different information that is found important from different children.

All other layers appear only once. In sequential LSTM they are calculated using previous hidden state and memory cell but in tree there can be an arbitrary number of them - one for each child. Thus, instead of hidden state, we use the sum of subtrees' hidden states and then i_j, o_j and u_j are calculated in the same way as in LSTM. The interpretation is preserved - we pass current and past transformed and filtered information through some "gates" with different roles. Then we update memory state c_j and we aggregate children's memory by taking the sum of memory cells that passed through "forget gate" - multiplying c_k by f_{jk} for $k \in C(j)$. In the end, we calculate node's hidden state h_j identically as in LSTM.

3. Experiments

3.1. Data

Dataset used in this work consists of two kinds of texts: product reviews of two types - perfumes and clothing (965 sentences), and sentiment-bearing sentences from the Skladnica [2] (265 sentences). This is a dependency treebank with sentiment annotations. These sentences were parsed using the Polish dependency parser models available online [3]. For each sentence in the treebank, sentiment of each phrase (whole sentences and sub-phrases of sentences) has been assigned by a linguist. Sentiment of each leaf word has been labeled according to Polish sentiment dictionary, an extension of http://zil.ipipan.waw.pl/SlownikWydzwieku/ and also verified manually. Sentiment labels of both phrases and leaves include three classes: neutral, positive and negative. The treebank consists of 12 861 sentiment-annotated phrases and words from the parse trees of 1 200 sentences. Test data consists of 350 sentences (5047 subphrases) of perfumes and clothing reviews.

Neural networks included in the research (sequential LSTM and Tree-structured LSTM) required vector representations of words. For this purpose 300-dimensional word2vec vectors trained on orthographical word forms on combination of Polish Wikipedia and the National Corpus of Polish were used[4].

3.2. Experiment settings

We compere four models: naive Bayes classifier, support vector machine (SVM), LSTM network and Tree-LSTM (own implementation in Theano[5] library available on GitHub[6]). The former two work on matrix-shaped data, the third one take as input sequences and the last is dedicated to trees.

The performance of naive Bayes classifier and SVM was tested using various representation - combinations of words counts weighting with and without dimension reduction (by manipulating vocabulary set and with SVD decomposition). The best results were obtained with the simplest solution - words counts without any weighting or dimension manipulation.

Neural networks - sequential and tree-structured LSTMs were trained analogously. First, there was separated validation set made of 165 sentences in order to determine optimal number of training epochs. Learning was stopped when there was no improvement of accuracy on validation set through 5 epochs. Both networks were tested with the hidden state of size 100 (experiments suggested this value to be optimal - as showed in table 1). In both cases stochastic gradient descend was used as learning algorithm and word vectors were tuned. We don't use any kind of regularization as we did not notice the improvement. The only difference is in batches construction. In case of sequential LSTM the batch size was constant and equal to 10. For Tree-LSTM every batch was single sentence, thus the number of labeled phrases in batch was a variable.

Hidden state dimension	50	100	150	200
LSTM	84.9	**85.2**	84.5	83.0
TreeLSTM	84.4	**85.1**	83.5	84.0

Table 1: LSTM and TreeLSTM accuracies on validation set depending on the hidden state dimension. Validation part was made by 10% of observation from training set. In both cases 100-dimesnional hidden state is optimum.

3.3. Results

During the experiments it turned out that learning models only on product reviews subset gave better results. That means that including the remaining 265 sentences of Skladnica in training set made the results worse. It is for sure the consequence of the fact that test set consists only of product reviews of the same type and that was already seen during the analysis of validation set. It shows how sentiment analysis is domain-sensitive. We would expect from the best models to be resistant to this issue but in this case it did not happen. We can suppose that it results from too small training sample that does not allow models to generalize well.

The label values distribution in test set is as follows: 73% neutral, 20% positive and 7% negative. Thus, 73% can be viewed as a baseline. Our results are summarized in Table 2. The table shows that tree-structured LSTM gives the best results among considered models. However, its superiority over sequential LSTM is not definite. We can also notice that both deep neural networks making use of text structure have a definite advantage over classifiers basing on bag-of-words model.

More detailed analysis of TreeLSTM performance is shown in Table 3. As we can see the best scores were obtained for the most frequent class, that is neutral.

Additionally we show in Table 4 some examples where both Naive Bayes and SVM failed but TreeLSTM predicted sentiment properly.

[2]http://treebank.nlp.ipipan.waw.pl/
[3]http://zil.ipipan.waw.pl/Sk%C5%82adnica
[4]Trained vectors can be found at http://mozart.ipipan.waw.pl/ axw/models/orth/
[5]http://deeplearning.net/software/theano/index.html
[6]https://github.com/norbertryciak/poleval

Method	Phrase structure	Accuracy
Naive Bayes	vector	75.4
SVM	vector	75.1
LSTM	sequence	79.2
Tree-LSTM	tree	79.5

Table 2: Test set accuracies. Phrase structure describes the format of data used by corresponding method. Models were learned only on subset of 965 sentences of product reviews.

Class	precision	recall	F1	support
negative	0.53	0.14	0.22	365
neutral	0.83	0.93	0.88	3666
positive	0.65	0.55	0.59	1016

Table 3: Classification report for TreeLSTM.

Phrase	Sentiment
zaskakujacy zapach	positive
ogólnie bardzo udane	positive
jeden z najciekawszych flakonów, jaki można by sobie wyobrazić na swojej półce	positive
Moim zdaniem zapach wyróżniajacy sie z tłumu	positive
niewygodny flakonik	negative
bo obawiam sie, że bedzie szybko ulatywał	negative

Table 4: Examples of phrases for which both NB and SVM failed but they were correctly classified by TreeLSTM. Most of them have positive sentiment.

4. Conclusion

In this paper, we presented the winning solution for sentiment analysis task - tree-structured Long Short Term Memory network, and compared it with a few other methods. We demonstrated the effectiveness of deep learning model utilizing the linguistic structure of the text - dependency trees.

5. References

Hochreiter, Sepp, 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.

Melville, Prem, Wojciech Gryc, and Richard D. Lawrence, 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09. New York, NY, USA: ACM.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean, 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, 2010. Recurrent neural network based language model. In *Interspeech*, volume 2.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan, 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

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*. Seattle, Washington, USA: Association for Computational Linguistics.

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