Think Positive: Towards Twitter Sentiment Analysis from Scratch

Cícero Nogueira dos Santos
Brazilian Research Lab
IBM Research
cicerons@br.ibm.com

Abstract

In this paper we describe a Deep Convolutional Neural Network (DNN) approach to perform two sentiment detection tasks: message polarity classification and contextual polarity disambiguation. We apply the proposed approach for the SemEval-2014 Task 9: Sentiment Analysis in Twitter. Despite not using any handcrafted feature or sentiment lexicons, our system achieves very competitive results for Twitter data.

1 Introduction

In this work we apply a recently proposed deep convolutional neural network (dos Santos and Gatti, 2014) that exploits from character- to sentence-level information to perform sentiment analysis of Twitter messages (tweets). The network proposed by dos Santos and Gatti (2014), named Character to Sentence Convolutional Neural Network (CharSCNN), uses two convolutional layers to extract relevant features from words and messages of any size.

We evaluate CharSCNN in the unconstrained track of the SemEval-2014 Task 9: Sentiment Analysis in Twitter (Rosenthal et al., 2014). Two subtasks are proposed in the SemEval-2014 Task 9: the contextual polarity disambiguation (SubtaskA), which consists in determining the polarity (positive, negative, or neutral) of a marked word or phrase in a given message; and the message polarity classification (SubtaskB), which consists in classifying the polarity of the whole message. We use the same neural network to perform both tasks. The only difference is that in SubtaskA, CharSCNN is fed with a text segment composed by the words in a context window centered at the target word/phrase. While in SubtaskB, CharSCNN is fed with the whole message.

The use of deep neural networks for sentiment analysis has been the focus of recent research. However, instead of convolutional neural network, most investigation has been done in the use of recursive neural networks (Socher et al., 2011; Socher et al., 2012; Socher et al., 2013).

2 Neural Network Architecture

Given a segment of text (e.g. a tweet), CharSCNN computes a score for each sentiment label $\tau \in T = \{positive, negative, neutral\}$. In order to score a text segment, the network takes as input the sequence of words in the segment, and passes it through a sequence of layers where features with increasing levels of complexity are extracted. The network extracts features from the character-level up to the sentence-level.

2.1 Initial Representation Levels

The first layer of the network transforms words into real-valued feature vectors (embeddings) that capture morphological, syntactic and semantic information about the words. We use a fixed-sized word vocabulary V^{wrd}, and we consider that words are composed of characters from a fixed-sized character vocabulary V^{chr}. Given a sentence consisting of N words $\{w_1, w_2, ..., w_N\}$, every word w_n is converted into a vector $u_n = [r^{wrd}; r^{wch}]$, which is composed of two subvectors: the *word-level embedding* $r^{wrd} \in \mathbb{R}^{d^{wrd}}$ and the *character-level embedding* $r^{wch} \in \mathbb{R}^{cl_u^0}$ of w_n. While word-level embeddings are meant to capture syntactic and semantic information, character-level embeddings capture morphological and shape information.

Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 647–651,
Dublin, Ireland, August 23-24, 2014.

2.1.1 Word-Level Embeddings

Word-level embeddings are encoded by column vectors in an embedding matrix $W^{wrd} \in \mathbb{R}^{d^{wrd} \times |V^{wrd}|}$. Each column $W_i^{wrd} \in \mathbb{R}^{d^{wrd}}$ corresponds to the word-level embedding of the i-th word in the vocabulary. We transform a word w into its word-level embedding r^{wrd} by using the matrix-vector product:

$$r^{wrd} = W^{wrd} v^w \qquad (1)$$

where v^w is a vector of size $\left|V^{wrd}\right|$ which has value 1 at index w and zero in all other positions. The matrix W^{wrd} is a parameter to be learned, and the size of the word-level embedding d^{wrd} is a hyper-parameter to be chosen by the user.

2.1.2 Character-Level Embeddings

In the task of sentiment analysis of Twitter data, important information can appear in different parts of a hash tag (e.g., "*#SoSad*", "*#ILikeIt*") and many informative adverbs end with the suffix "*ly*" (e.g. "*beautifully*", "*perfectly*" and "*badly*"). Therefore, robust methods to extract morphological and shape information from this type of tokens must take into consideration all characters of the token and select which features are more important for sentiment analysis. Like in (dos Santos and Zadrozny, 2014), we tackle this problem using a convolutional approach (Waibel et al., 1989), which works by producing local features around each character of the word and then combining them using a max operation to create a fixed-sized character-level embedding of the word.

Given a word w composed of M characters $\{c_1, c_2, ..., c_M\}$, we first transform each character c_m into a character embedding r_m^{chr}. Character embeddings are encoded by column vectors in the embedding matrix $W^{chr} \in \mathbb{R}^{d^{chr} \times |V^{chr}|}$. Given a character c, its embedding r^{chr} is obtained by the matrix-vector product:

$$r^{chr} = W^{chr} v^c \qquad (2)$$

where v^c is a vector of size $\left|V^{chr}\right|$ which has value 1 at index c and zero in all other positions. The input for the convolutional layer is the sequence of character embeddings $\{r_1^{chr}, r_2^{chr}, ..., r_M^{chr}\}$.

The convolutional layer applies a matrix-vector operation to each window of size k^{chr} of successive windows in the sequence $\{r_1^{chr}, r_2^{chr}, ..., r_M^{chr}\}$. Let us define the vector $z_m \in \mathbb{R}^{d^{chr} k^{chr}}$ as the concatenation of the character embedding m, its $(k^{chr} - 1)/2$ left neighbors, and its $(k^{chr} - 1)/2$ right neighbors:

$$z_m = \left(r^{chr}_{m-(k^{chr}-1)/2}, ..., r^{chr}_{m+(k^{chr}-1)/2}\right)^T$$

The convolutional layer computes the j-th element of the vector r^{wch}, which is the character-level embedding of w, as follows:

$$\left[r^{wch}\right]_j = \max_{1<m<M} \left[W^0 z_m + b^0\right]_j \qquad (3)$$

where $W^0 \in \mathbb{R}^{cl_u^0 \times d^{chr} k^{chr}}$ is the weight matrix of the convolutional layer. The same matrix is used to extract local features around each character window of the given word. Using the max over all character windows of the word, we extract a "global" fixed-sized feature vector for the word.

Matrices W^{chr} and W^0, and vector b^0 are parameters to be learned. The size of the character vector d^{chr}, the number of convolutional units cl_u^0 (which corresponds to the size of the character-level embedding of a word), and the size of the character context window k^{chr} are hyper-parameters.

2.2 Sentence-Level Representation and Scoring

Given a text segment x with N words $\{w_1, w_2, ..., w_N\}$, which have been converted to joint word-level and character-level embedding $\{u_1, u_2, ..., u_N\}$, the next step in CharSCNN consists in extracting a segment-level representation r_x^{seg}. Methods to extract a segment-wide feature set most deal with two main problems: text segments have different sizes; and important information can appear at any position in the segment. A convolutional approach is a good option to tackle this problems, and therefore we use a convolutional layer to compute the segment-wide feature vector r^{seg}. This second convolutional layer works in a very similar way to the one used to extract character-level features for words. This layer produces local features around each word in the text segment and then combines them using a max operation to create a fixed-sized feature vector for the segment.

The second convolutional layer applies a matrix-vector operation to each window of size k^{wrd} of successive windows in the sequence $\{u_1, u_2, ..., u_N\}$. Let us define the vector $z_n \in \mathbb{R}^{(d^{wrd}+cl_u^0)k^{wrd}}$ as the concatenation of a sequence of k^{wrd} embeddings, centralized in the n-th word[1]:

$$z_n = \left(u_{n-(k^{wrd}-1)/2}, ..., u_{n+(k^{wrd}-1)/2}\right)^T$$

The convolutional layer computes the j-th element of the vector r^{seg} as follows:

$$[r^{seg}]_j = \max_{1<n<N} \left[W^1 z_n + b^1\right]_j \quad (4)$$

where $W^1 \in \mathbb{R}^{cl_u^1 \times (d^{wrd}+cl_u^0)k^{wrd}}$ is the weight matrix of the convolutional layer. The same matrix is used to extract local features around each word window of the given segment. Using the max over all word windows of the segment, we extract a "global" fixed-sized feature vector for the segment. Matrix W^1 and vector b^1 are parameters to be learned. The number of convolutional units cl_u^1 (which corresponds to the size of the segment-level feature vector), and the size of the word context window k^{wrd} are hyper-parameters to be chosen by the user.

Finally, the vector r_x^{seg}, the "global' feature vector of text segment x, is processed by two usual neural network layers, which extract one more level of representation and compute a score for each sentiment label $\tau \in T$:

$$s(x) = W^3 h(W^2 r_x^{seg} + b^2) + b^3 \quad (5)$$

where matrices $W^2 \in \mathbb{R}^{hl_u \times cl_u^1}$ and $W^3 \in \mathbb{R}^{|T| \times hl_u}$, and vectors $b^2 \in \mathbb{R}^{hl_u}$ and $b^3 \in \mathbb{R}^{|T|}$ are parameters to be learned. The transfer function $h(.)$ is the hyperbolic tangent. The size of the number of hidden units hl_u is a hyper-parameter to be chosen by the user.

2.3 Network Training

Our network is trained by minimizing a negative likelihood over the training set D. Given a text segment x, the network with parameter set θ computes a score $s_\theta(x)_\tau$ for each sentiment label $\tau \in T$. In order to transform this score into a conditional probability $p(\tau|x, \theta)$ of the label given the segment and the set of network parameters θ, we apply a softmax operation over all tags:

$$p(\tau|x, \theta) = \frac{e^{s_\theta(x)_\tau}}{\sum_i e^{s_\theta(x)_i}} \quad (6)$$

Taking the log, we arrive at the following conditional log-probability:

$$\log p(\tau|x, \theta) = s_\theta(x)_\tau - \log\left(\sum_{\forall i \in T} e^{s_\theta(x)_i}\right) \quad (7)$$

We use stochastic gradient descent (SGD) to minimize the negative log-likelihood with respect to θ:

$$\theta \mapsto \sum_{(x,y) \in D} -\log p(y|x, \theta) \quad (8)$$

where (x, y) corresponds to a text segment (e.g. a tweet) in the training corpus D and y represents its respective sentiment class label.

We use the backpropagation algorithm to compute the gradients of the network (Lecun et al., 1998; Collobert, 2011). We implement the CharSCNN architecture using the automatic differentiation capabilities of the *Theano* library (Bergstra et al., 2010).

3 Experimental Setup and Results

3.1 Unsupervised Learning of Word-Level Embeddings

Unsupervised pre-training of word embeddings has shown to be an effective approach to improve model accuracy (Collobert et al., 2011; Luong et al., 2013; Zheng et al., 2013). In our experiments, we perform unsupervised learning of word-level embeddings using the *word2vec* tool[2].

We use two Twitter datasets as sources of unlabeled data: the Stanford Twitter Sentiment corpus (Go et al., 2009), which contains 1.6 million tweets; and a dataset containing 10.4 million tweets that were collected in October 2012 for a previous work by the author (Gatti et al., 2013). We tokenize these corpora using Gimpel et al.'s (2011) tokenizer, and removed messages that are less than 5 characters long (including white spaces) or have less than 3 tokens. Like in (Collobert et al., 2011) and (Luong et al., 2013), we lowercase all words and substitute each numerical digit by a 0 (e.g., *1967* becomes *0000*). The resulting corpus contains about 12 million tweets.

We do not perform unsupervised learning of character-level embeddings, which are initialized by randomly sampling each value from an uniform distribution: $\mathcal{U}(-r, r)$, where $r = \sqrt{\frac{6}{|V^{chr}| + d^{chr}}}$. The character vocabulary is

[1] We use a special *padding token* for the words with indices outside of the text segment boundaries.

[2] https://code.google.com/p/word2vec/

constructed by the (not lowercased) words in the training set, which allows the neural network to capture relevant information about capitalization.

3.2 Sentiment Corpora and Model Setup

SemEval-2014 Task 9 is a rerun of the SemEval-2013 Task 2 (Nakov et al., 2013), hence the training set used in 2014 is the same of the 2013 task. However, as we downloaded the Twitter training and development sets in 2014 only, we were not able to download the complete dataset since some tweets have been deleted by their respective creators. In Table 1, we show the number of messages in our SemEval-2013 Task 2 datasets.

Dataset	SubtaskA	SubtaskB
Train	7390	8213
Dev.	904	1415
Twitter2013 (test)	3491	3265
SMS2013 (test)	2,334	2,093

Table 1: Number of tweets in our version of SemEval-2013 Task2 datasets.

In SemEval-2014 Task 9, three different test sets are used: Twitter2014, Twitter2014Sarcarm and LiveJournal2014. While the two first contain Twitter messages, the last one contains sentences from LiveJournal blogs. In Table 2, we show the number of messages in the SemEval-2014 Task 9 test datasets.

Test Dataset	SubtaskA	SubtaskB
Twitter2014	2597	1939
Twitter2014Sarcasm	124	86
LiveJournal2014	1315	1142

Table 2: Number of tweets in the SemEval-2014 Task9 test datasets.

We use the copora Twitter2013 (test) and SMS2013 to tune CharSCNN's hyper-parameter values. In Table 3, we show the selected hyper-parameter values, which are the same for both SubtaskA and SubtaskB. We concatenate the SemEval-2013 Task 2 training and development sets to train the submitted model.

Parameter	Parameter Name	Value
d^{wrd}	Word-Level Emb. dim.	100
k^{wrd}	Word Context window	3
d^{chr}	Char. Emb. dim.	5
k^{chr}	Char. Context window	5
cl_u^0	Char. Convol. Units	30
cl_u^1	Word Convol. Units	100
hl_u	Hidden Units	300
λ	Learning Rate	0.02

Table 3: Neural Network Hyper-Parameters.

3.3 Sentiment Prediction Results

In Table 4, we present the official results of our submission to the SemEval-2014 Task9. In SubtaskB, CharSCNN's result for the Twitter2014 test corpus is the top 11 out of 50 submissions, and is 3.9 F-measure points from the top performing system. In the SubtaskA, CharSCNN's result for the Twitter2014 test corpus is the top 6 out of 27 submissions. These are very promising results, since our approach do not use any handcrafted features or lexicons, all features (representations) are automatically learned from unlabeled and labeled data.

Nevertheless, our system result for the LiveJournal2014 corpus in SubtaskB is regular. For this dataset CharSCNN achieves only the top 25 out of 50 submissions, and is 7.9 F-measure points behind the top performing system. We believe the main reason for this poor result is the exclusive use of Twitter data in the unsupervised pre-training.

Test Subset	SubtaskA	SubtaskB
Twitter2014	82.05	67.04
Twitter2014Sarcasm	76.74	47.85
LiveJournal2014	80.90	66.96
Twitter2013	88.06	68.15
SMS2013	87.65	63.20

Table 4: Average F-measure of CharSCNN for different test sets.

4 Conclusions

In this work we describe a sentiment analysis system based on a deep neural network architecture that analyses text at multiple levels, from character-level to sentence-level. We apply the proposed system to the SemEval-2014 Task 9 and achieve very competitive results for Twitter data in both contextual polarity disambiguation and message polarity classification subtasks. As a future work, we would like to investigate the impact of the system performance for the LiveJournal2014 corpus when the unsupervised pre-training is performed using in-domain texts.

References

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 224–232.

Cícero Nogueira dos Santos and Maíra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, Dublin, Ireland.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML), JMLR: W&CP volume 32*, Beijing, China.

Maíra Gatti, Ana Paula Appel, Cícero Nogueira dos Santos, Claudio Santos Pinhanez, Paulo Rodrigo Cavalin, and Samuel Martins Barbosa Neto. 2013. A simulation-based approach to analyze the information diffusion in microblogging online social network. In *Winter Simulation Conference*, pages 1685–1696.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 42–47.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report, Stanford University.

Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Conference on Computational Natural Language Learning*, Sofia, Bulgaria.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Sara Rosenthal, Preslav Nakov, Alan Ritter, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment Analysis in Twitter. In Preslav Nakov and Torsten Zesch, editors, *Proceedings of the 8th International Workshop on Semantic Evaluation*, SemEval'14, Dublin, Ireland.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of theConference on Empirical Methods in Natural Language Processing*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 647–657.