TWITTER SENTIMENT ANALYSIS (ALMOST) FROM SCRATCH

Rémi Lebret[a] Pedro H. O. Pinheiro
Ronan Collobert

Idiap-RR-15-2016

MAY 2016

[a]Idiap

Twitter Sentiment Analysis (Almost) from Scratch

Rémi Lebret^{1,2} and **Pedro O. Pinheiro**^{1,2} and **Ronan Collobert**^{1}
^{1}Idiap Research Institute, Martigny, Switzrland
^{2}École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
remi@lebret.ch, pedro@opinheiro.com, ronan@collobert.com

Abstract

A popular application in Natural Language Processing (NLP) is the Sentiment Analysis, *i.e.*, the task of extracting contextual polarity from a given text. The social network Twitter provides an immense amount of text (called tweets) generated by users with a maximum number of 140 characters. In this paper, we propose to learn a tweet representation from publicly provided data from tweets in order to infer sentiment from them. One challenge of this task is the fact that tweets are generated from very different users, making the data very heterogeneous (different from regular data which is written in proper English). Another challenge is, clearly, the large scale of the problem. We propose a deep learning sentence representation (called *tweet representation*) from user generated data to infer sentiment from tweets. This representation is learned from scratch (directly from the words in tweets) over a large unlabeled corpus of tweets. We demonstrate that we achieve state-of-the-art results for sentiment analysis on tweets.

Introduction

Twitter is an online social networking service that enables users to send and read short 140-character messages called "tweets". Users post messages where they can express opinions about different topics, which includes products or services. These tweets are publicly visible by default, which makes Twitter a gold mine for consumers, marketers or companies. Consumers can analyze the opinions of Twitter users about products or services before making a purchase. Marketers can analyze customer satisfaction or research public opinion of their company and products. Companies can gather critical feedback about problems in newly released products. Identifying and extracting this subjective information has therefore become a key point. This explains why sentiment analysis is an important current research area.

Previous research in sentiment analysis have focused on classifying larger pieces of text, like reviews of products or movies (Pang, Lee, and Vaithyanathan 2002). Twitter data posses many unique properties that make sentiment analysis application much more challenging than in other domains: (i) Maximum length of a tweet is 140 characters. The dataset considered has in average 14 words and 78 characters. (ii) Twitter users post messages from many different media (*e.g.* cellphone). The quantity of misspelling, slang and informal language is much higher than in other types of data. (iii) Twitter users post an infinitude of different subjects. This differs from classical sentiment analysis approach, which are usually focused on a specific domain (such as movie reviews).

Successful methods for sentiment analysis are traditionally based on *bag-of-words*. In this framework, each tweet is represented as a set of words present in it (a sparse vector). These words are transformed into numeric values, such as their frequencies of occurrence or binary values (appearing or not in the tweet). Some term weightings (*e.g.* the popular *td-idf*) have also been defined to reflect how important a word is for describing a tweet. These are considered as features for training a classifier. In order to enrich the model, some other features might be included. It is thus common to use part of speech or information coming from lexical databases (such as WordNet). Naive Bayes, Maximum Entropy, and Support Vector Machine are often the classifiers of choice.

In this paper, we propose instead a model based on convolutional neural networks that is able to learn efficient tweet representations for sentiment analysis. These tweet representations are directly learned from the words contained in tweets, without the use of any additional features. They are continuous low-dimensional vectors, and used to predict the sentiment polarity of tweets. We show that our model beats the current state-of-the-art results for this task. Aside from outperforming baseline approaches on sentiment analysis, this tweet representation framework can be applied in many other NLP applications, such as information retrieval or tweet classification.

Convolutional Neural Network for Tweet Classificaton

Traditional NLP approaches extract a rich set of hand-designed features from documents which are then fed to a standard classification algorithm. The choice of features in this scenario is a task-specific empirical process. In contrast, we want to pre-process our features as little as possible. In that respect, a multilayer neural network architecture seems appropriate as it can be trained in an end-to-end fashion on the task of interest (Collobert et al. 2011). Representations of each word in a tweet are fed to a convolutional layer followed by a max layer. A tweet representation is thus obtained. A softmax classifier is then trained to infer whether a tweet is positive or negative.

Notations

We consider a neural network $f_\theta(\cdot)$, with parameters θ. Any feed-forward neural network with L layers, can be seen as a composition of functions $f^l_\theta(\cdot)$, corresponding to each layer l:

$$f_\theta(\cdot) = f^L_\theta(f^{L-1}_\theta(\dots f^1_\theta(\cdot)\dots)) \,. \tag{1}$$

Given a matrix A we denote $[A]_{i,j}$ the coefficient at row i and column j in the matrix. $\langle A\rangle^1_i$ represents the i^{th} column of matrix A. We also denote $\langle A\rangle^{k_{sz}}_i$ the vector obtained by concatenating the k_{sz} column vectors around the i^{th} column vector of matrix A. For a vector v, we denote $[v]_i$ the scalar at index i in the vector. Finally, a sequence of elements $\{x_1, x_2, \dots, x_T\}$ is written $[x]^T_1$. The i^{th} element of the sequence is $[x]_i$.

Lookup-Table Layer

We consider a fixed-sized word dictionary $\mathcal{D}$. Given a tweet of T words $w_1, w_2, \dots, w_T$, each word $w_t \in \mathcal{D}$ is first embedded into a d_{wrd}-dimensional vector space, by applying a lookup-table operation:

$$LT_W(w_t) = W \left(\begin{array}{c} 0, \dots, \quad 1 \quad , \dots, 0 \\ \text{at index } w_t \end{array} \right) = \langle W\rangle^1_{w_t} \,, \tag{2}$$

where the matrix $W \in \mathbb{R}^{d_{wrd} \times |\mathcal{D}|}$ represents the words to be learned in this lookup-table layer. $\langle W\rangle^1_{w_t} \in \mathbb{R}^{d_{wrd}}$ is the w^{th}_t column of W and d_{wrd} is the word vector size. Given any sequence of T words $[w]^T_1$ in $\mathcal{D}$, the lookup table layer applies the same operation for each word in the sequence, producing the following output matrix:

$$LT_W([w]^T_1) = \left(\langle W\rangle^1_{[w]_1} \;\; \langle W\rangle^1_{[w]_2} \;\; \dots \;\; \langle W\rangle^1_{[w]_T} \right) \,. \tag{3}$$

This matrix can then be fed to further neural network layers.

Convolutional Layer

Our convolutional network successively takes the complete tweets and produces local features around each word of the tweet. Considering a fixed size k_{sz} (a hyper-parameter) window of words. Each word in the window is first passed through the lookup table layer (3), producing a matrix of word features of fixed size $d_{wrd} \times k_{sz}$. This matrix can be viewed as a $d_{wrd}k_{sz}$-dimensional vector by concatenating each column vector, which can be fed to further neural network layers. More formally, the word feature window given by the first network layer can be written as:

$$\langle f^1_\theta\rangle^1_t = \langle LT_W([w]^T_1)\rangle^{k_{sz}}_t = \begin{pmatrix} \langle W\rangle^1_{[w]_{t-k_{sz}/2}} \\ \vdots \\ \langle W\rangle^1_{[w]_t} \\ \vdots \\ \langle W\rangle^1_{[w]_{t+k_{sz}/2}} \end{pmatrix} \tag{4}$$

A matrix-vector operation is applied to each window of successive windows in the tweet. The t^{th} output column of the convolution layer can be computed as:

$$\langle f^2_\theta\rangle^1_t = W^2 \langle f^1_\theta\rangle^1_t + b^2 \;\; \forall t \,, \tag{5}$$

where $W^2 \in \mathbb{R}^{n_{nfilter} \times d_{wrd}k_{sz}}$ and $b \in \mathbb{R}^{n_{nfilter}}$ are the parameters to be trained. The hyper-parameter $n_{nfilter}$ is the *number of filters* of the convolution layer. The weight matrix W^2 is the same across all windows in the tweet.

Max Layer

The size of the output will depend on the number of words in the tweet fed to the network. Local feature vectors extracted by the convolutional layer have to be combined to obtain a global feature vector, with a fixed size independent of the tweet length, in order to apply subsequent standard affine layers. We used a max approach, which forces the network to capture the most useful local features produced by the convolutional layer, for the task at hand. Given a matrix f^2_θ output by the convolutional layer, the Max layer outputs a vector f^3_θ:

$$[f^3_\theta]_i = \max_t [f^2_\theta]_{i,t} \quad 1 \le i \le n_{nfilter} \,. \tag{6}$$

In the case of sentiment analysis, the i^{th} filter might capture positive or negative sentiment. This fixed sized global feature vector can be seen as a *tweet representation*. It is then used as input to a fully connected layer for classification.

Class Prediction

As any classical neural network, the architecture performs several matrix-vector operations on its inputs, interleaved with some non-linear transfer function. As transfer function, we chose a (fast) “hard” version of the hyperbolic tangent:

$$\left[f^l_\theta\right]_i = \text{HardTanh}(\left[f^{l-1}_\theta\right]_i) \,, \tag{7}$$

where

$$\text{HardTanh(x)} = \begin{cases} -1 & \text{if } x < 1 \\ x & \text{if } -1 \le x < 1 \\ 1 & \text{if } x > 1 \end{cases} \tag{8}$$

Finally, the output size of the last layer L is equal to the number of possible classes for the task of interest. Each output can be then interpreted as a score of the corresponding class (given the input of the network).

Training

The network is trained by maximizing a likelihood over the training data, using stochastic gradient ascent. We note $[f_\theta]_y$ the y^{th} output of the network and θ all the trainable parameters. Using a training set $\mathcal{T}$, we want to maximize the following log-likelihood with respect to θ:

$$\theta \mapsto \sum_{(x,y)\in T} \log p(y|x,\theta) \tag{9}$$

where x corresponds to a training tweet and y represents the corresponding class. The probability $p(y|x,\theta)$ is computed from the outputs of the neural network by adding a softmax operation over all the classes:

$$p(i|x,\theta) = \frac{e^{[f_\theta]_i}}{\sum_j e^{[f_\theta]_j}} \tag{10}$$

We can express the log-likelihood for one training example (x,y) as follows:

$$\log p(y|x,\theta) = [f_\theta]_y - \log(\sum_j e^{[f_\theta]_j}) \tag{11}$$

Related Works

In 2009, Go, Bhayani, and Huang proposed a model to automatically extract sentiment from tweets. They consider three different feature-based classic machine learning classifiers to infer sentiment on tweets: (i) Naive Bayes (NB), (ii) Max-Entropy (MaxEnt) and (iii) Support Vector Machine (SVM). They report results for different set of features: Unigram, Unigram+Bigram and Unigram+Part of Speech (POS). NB classifiers attempt to build a probabilistic classifiers based on modeling the underlying word features in different classes. Tweets are then classified based on the posterior probability of the tweets belonging to the positive or negative classes on the basis of the word presence in the tweets. An important disadvantage of NB modeling is that it has strong feature independence assumptions. Maximum entropy is a general technique for estimating probability distributions from data, which does not suffer from any independence assumptions. This means that features like bigrams and phrases can be added to MaxEnt without worrying about features overlapping. Because these feature-based models represent tweets in sparse high-dimensional vectors, SVM is also appropriate for learning text classifiers (Joachims 1998). SVM classifiers attempt to partition the data space with the use of linear or non-linear delineations between the different classes. The key in such classifiers is to determine the optimal boundaries between the different classes and use them for the purposes of classification.

More recently, Poria et al. (2014) outperform these baselines methods by employing lexical resources to provide polarity scores (from SenticNet) or emotion labels (from WordNet-Affect) for words and concepts. Also in 2014, Kalchbrenner, Grefenstette, and Blunsom have proposed a dynamic convolutional neural network (DCNN) for modelling sentences. While we propose to simply extract a global feature vector with a max approach after one convolutional layer, they used multiple layers of convolution followed by dynamic k-max pooling to induce a structured feature graph over a given tweet. This approach is therefore much more complex (deeper) than our proposed model.

Evaluation

In this section we describe the data used in our experiments, the experimental results and a brief discussion of the results.

Experimental Setup

For our experiments, we consider the same dataset used by Go, Bhayani, and Huang (2009). For the training data, the tweets were extracted using the official Twitter Application Programming Interface (API)[1]. The sentiment of Twitter posts have been predicted using distant supervision. The positive tweets were selected with a query for tweets containing ":)", ":-)", ":)", ":D", "=)". The negative ones with a query for tweets containing ":(", ":-(", ": (". The tweets in the training set are from the time period between April 6, 2009 to June 25, 2009. The following filtering were applied on the data: (i) Emoticons were removed from the tweets, (ii) tweets containing both positive and negative emoticons were removed and (iii) retweets were removed to avoid giving extra weight to a particular weight. Stripping out the emoticons causes the classifier to learn from the other features (the words in our case) present in the tweet. The final training data consists of a total of 1.6M tweets, half labeled as positive and half labeled as negative. The test data is manually collected, using the web application. A set of 177 negative tweets and 182 positive tweets were manually marked. Not all the test tweets has emoticons.

We also consider the particularity of Twitter language model to reduce the dictionary size and make the data more concise. This is achieved with the following data preprocessing:

- **Target**: every word started with the character @ is replaced by the special token "TARGET".
- **Link**: every URL (http://...) is replaced by the special token "URL".
- **Hashtag**: every word started with the character # is replaced by the special token "HASHTAG".
- **Repeated Letters**: every letter occurring more than two times is replaced with two occurrences (*e.g.*, 'huuuuuuuungry' is replaced by 'huungry')
- **Digit**: all occurrences of sequences of numbers within a word are replaced with the special token "NUMBER".

Finally, all words are lowercased. The resulting tweets have been tokenized using the CMU ARK Twitter NLP tools[2] (Gimpel et al. 2011). This results in a 323,393 words dictionary $\mathcal{D}$.

It has been shown that the generalization performance can be improved on several NLP tasks by using pre-trained word representations (Turian, Ratinov, and Bengio 2010). The lookup-table parameter W is thus initialized with some

[1]Available at `http://apiwiki.twitter.com`

[2]Available at `http://www.ark.cs.cmu.edu/TweetNLP/`

available pre-trained word representations[3] (Lebret and Collobert 2014). The hyper-parameters were chosen considering a validation set extracted from the training data. The optimal hyper-parameters used are:

- word vector size: $d_{wrd} = 50$,
- word window size: $k_{sz} = 3$,
- number of filters in the convolution: $n_{filter} = 30$,
- number of hidden units in the classifier: $n_{hu} = 50$.

Results

Results reported in Table 1 show that our model significantly outperforms the baseline models, and a model with a prior knowledge on sentiments (EmoSenticSpace). It also slightly outperforms a much deeper convolutional neural network (DCNN), which indicates that there is no need for multiple convolutional layers in sentiment analysis. The number of filters used in our model is very low, $n_{filter} = 30$. This means that tweets are represented in 30-dimensional continuous vectors after the max layer and before the softmax classifier. Conversely, traditional bag-of-words based classifiers will represent tweets with as many features as there are words in the dictionary. Considering that our dictionary $\mathcal{D}$ contains 323,393 words, this is about ten thousand times higher than our tweet representations.

Model	**Accuracy** (%)
SVM	81.6
BINB	82.7
MAXENT	83.0
EMOSENTICSPACE	85.1
DCNN	87.4
OUR MODEL	**88.3**

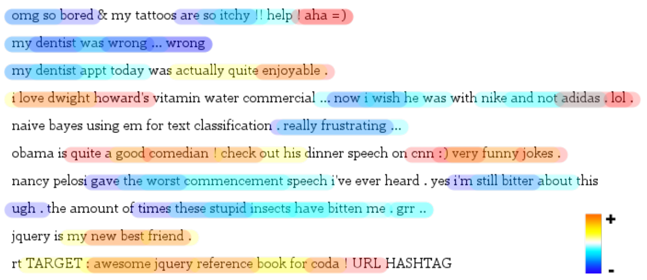
Table 1: Accuracy on the Twitter sentiment analysis test set. The three classical models are based on unigram and bigram features; the results are reported from Go, Bhayani, and Huang (2009).

At inference time, our model can output polarity scores for every windows of k_{sz} words in a given tweet by simply removing the max layer. This is a valuable asset to detect which parts of a tweet are positive or negative, as illustrated in Figure 1. This also helps to understand why certain tweets are misclassified. Some examples of misclassified tweets are in Figure 1 where both sentiments have been detected.

Conclusion

In this paper, we introduce a model capable of learning to classify Twitter messages into sentiment categories using distant supervised learning. Contrary to classical approaches, our model learns a low-dimensional continuous representation for tweets. Moreover, the model is trained (almost) from scratch, using the raw tweets as training data.

[3]Available at `http://lebret.ch/words/`



omg so bored & my tattoos are so itchy !! help ! aha =)
my dentist was wrong ... wrong
my dentist appt today was actually quite enjoyable .
i love dwight howard's vitamin water commercial ... now i wish he was with nike and not adidas . lol .
naive bayes using em for text classification . really frustrating ...
obama is quite a good comedian ! check out his dinner speech on cnn :) very funny jokes .
nancy pelosi gave the worst commencement speech i've ever heard . yes i'm still bitter about this
ugh . the amount of times these stupid insects have bitten me . grr ..
jquery is my new best friend .
rt TARGET : awesome jquery reference book for coda ! URL HASHTAG

Figure 1: Selection of tweets from the test set where sentiments are highlighted using our model outputs. The blue color scale indicates negative sentiment, the red one indicates positive sentiments.

The proposed approach outperforms baseline methods and a more complex (deeper) model. Code and data will be released online at the time of the conference.

References

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research.*

Gimpel, K.; Schneider, N.; O'Connor, B.; Das, D.; Mills, D.; Eisenstein, J.; Heilman, M.; Yogatama, D.; Flanigan, J.; and Smith, N. A. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49nd Annual Meeting of the Association for Computational Linguistics.*

Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter Sentiment Classification using Distant Supervision. *CS224N Project Report.*

Joachims, T. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning.*

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A Convolutional Neural Network for Modelling Sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.*

Lebret, R., and Collobert, R. 2014. Word Embeddings through Hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics.*

Pang, B.; Lee, L.; and Vaithyanathan, S. 2002. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In *Proceedings of the 40nd Annual Meeting of the Association for Computational Linguistics.*

Poria, S.; Gelbukh, A. F.; Cambria, E.; Hussain, A.; and Huang, G.-B. 2014. EmoSenticSpace: A novel framework for affective common-sense reasoning.

Turian, J.; Ratinov, L.; and Bengio, Y. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48nd Annual Meeting of the Association for Computational Linguistics.*