REPORT

PROJECT: Tweet Sentiment Analysis

(Semeval2019 – Task 03)

Authors

David BENSABAT
Xavier YVONNE

Teacher

Abdessalam BOUCHEKIF

January 13, 2019

Table of contents

1 Problem statement

Twitter is a popular social networking website where members create and interact with messages known as “tweets”. This serves as a mean for individuals to express their thoughts or feelings about different subjects. Various different parties such as consumers and marketers have done sentiment analysis on such tweets to gather insights into products or to conduct market analysis.

Furthermore, with the recent advancements in machine learning algorithms, we are able to improve the accuracy of our sentiment analysis predictions.

This topic is subject to various contests, and in particular SemEval2019 - Task #3[1], which stemmed this very project. **The goal is to classify the tweets from the provided testing dataset into 4 categories: happy, sad, angry, and others**.

Actually, a single piece of the corpus actually consists in 3 short messages written by 2 users (the first and third messages from a User A, and the second from a User B); nevertheless, with a slight abuse, we keep calling a tweet such a 3-sentence dialog. If several sentiment intensities are found in the tweet (e.g. because User A says something angry and User B says something happy in it), the outcome must be “others”.

To help us, the Kaggle team itself provides a dataset consisting in 30,000 labeled tweets: 15,000 for the class “others”, and 15,000 in total for the 3 other classes.

The classification is evaluated with the F1-score, which is the harmonic mean between the precision and the recall.

1 https://www.humanizing-ai.com/emocontext.html

2 Methodology and implementation

2.1 Ovierview

Our code is wholly written in Python 3.
In addition to the dataset from Kaggle (called Dataset C), we use 2 other datasets (called A and B), see *Section 2.2*.
After suitable pre-processing operations (see *Section 2.3*), these datasets are given to a distinct neural networks (aka DNN) called CNN-A, CNN-B or CNN-C, depending on the dataset it is fed with. We describe these DNNs in *Section 2.4*. CNN-A is trained from scratch, and the next 2 others using Transfer Learning.

2.2 Data description

Our classification problem is deceitfully simple: whereas any standard supervised learning method could do quite a good job (SVM, decision tree, random forest, neural networks, DNNs, etc), the major issue actually comes from finding a good dataset to train our models on. Indeed, the very form of each sample (a very short dialog between 2 persons, instead of a single sentence), and the fact that 4 classes must be predicted - and not only 2 or 3 -, are the core of the problem.

Of course, we do have the Kaggle dataset, but it is so small that it is not hard to convince oneself that if we were feeding an untrained model with it, especially a DNN, it were very likely bound to over-fit. On the other hand, we were not able to find any dataset that could directly used for our classification problems. We thus had to find larger datasets addressing similar problems.

1. The first dataset we found consist in 1,600,000 labeled tweets given in a Kaggle contest[2]: each tweet is made with 1 sentence each, and 2 categories must be predicted (positive/negative).
   The size of this dataset makes it difficult to feed a DNN with it on a regular laptop without a GPU, which unfortunately we don't have. We were therefore forced to keep less than the half of it (approximately 640,000 tweets) and even then, training our models on it took hours and hours.

2. The second dataset we use comes from our teacher Abdessalam Bouchekif. It consists in 416,000 labeled tweets (made with 1 sentence each) with 6 categories to predict (anger/sadness/fear/surprise/joy/love).

3. The third dataset comes from Kaggle team itself, consisting in 30,000 labeled tweets: 15,000 for the class "others", and 15,000 in total for the 3 other classes (anger/sad/happy).

2 https://www.kaggle.com/c/cs5228-project-2

To summarize, we use 3 datasets whose main features are as follows:

Dataset name	Size (train + validation)	Number of sentences/Tweet	Output for each tweet
A	800,000	1	positive / negative
B	416,000	1	anger / sadness / fear / surprise / joy / love
C	30,000	3	happy / sad / angry / other

2.3 Pre-processing and feature extraction

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as re-tweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. Following Abdul Fatir Ansari et al.[3], we apply an extensive number of pre-processing steps to standardize the dataset and reduce its size. The statistics given here refer to Dataset A (see *Section 2.2*), but we applied these steps on Datasets B and C as well.

We first do some general pre-processing on tweets which is as follows.

- Convert the tweet to lower case.
- Replace 2 or more dots (.) with space.
- Strip spaces and quotes (" and ') from the ends of tweet.
- Replace 2 or more spaces with a single space.
- Remove hashtag symbols (#), e.g. replace #hello with hello. The regular expression used to match hashtags is #(\S+).

We handle special twitter features as follows.

- URLs. As any particular URL is not important for text classification, let alone sentiment analysis, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).
- User Mention. Users often mention other users in their tweets by @handle, but again this is irrelevant for sentiment analysis. We replace all user mentions with the word USER_MENTION. The regular expression used to match user mention is @[\S]+.
- Retweet. Retweets are tweets which have already been sent by someone else and are shared by other users. Retweets begin with the letters RT. We remove RT from the tweets as it is not an important feature for text classification. The regular expression used to match retweets is \brt\b.
- Emoticons. They give very important clues on the user sentiment, because users use them in their tweets to convey different emotions. It is impossible to exhaustively match all the different emoticons used on social media as the number is ever increasing. However, we

3 https://github.com/abdulfatir/twitter-sentiment-analysis

match some common emoticons which are used very frequently. We replace the matched emoticons with either EMO_POS or EMO_NEG depending on whether it is conveying a positive or a negative emotion, according to the following table.

Emoticon(s)	**Type**	**Regex**	**Replacement**
:), :), :-), (:, (:, (-:, :’)	Smile	(:\s?\)|:-\)|\(\s?:|\(-:|:\’\))	EMO_POS
:D, : D, :-D, xD, x-D, XD, X-D	Laugh	(:\s?D|:-D|x-?D|X-?D)	EMO_POS
;-), ;), ;-D, ;D, (;, (-;	Wink	(:\s?\(|:-\(|\)\s?:|\)-:)	EMO_POS
<3, :*	Love	(<3|:*)	EMO_POS
:-(, : (, :(,):,)-:	Sad	(:\s?\(|:-\(|\)\s?:|\)-:)	EMO_NEG
:,(, :’(, :"(	Cry	(:,\(|:\’\(|:"\()	EMO_NEG

After applying tweet level pre-processing, we processed individual words of tweets as follows.

- Strip any punctuation [’"?!,.():;] from the word.
- Convert 2 or more letter repetitions to 2 letters. Some people send tweets like I am sooooo happppppy adding multiple characters to emphasize on certain words. This is done to handle such tweets by converting them to I am soo happy.
- Remove - and ’. This is done to handle words like t-shirt and their’s by converting them to the more general form tshirt and theirs.
- Check if the word is valid and accept it only if it is. We define a valid word as a word which begins with an alphabet with successive characters being alphabets, numbers or one of dot (.) and underscore(_).

Once this pre-processing is made, we extract unigrams (words) from our dataset and create a frequency distribution of these words. A total of 181,232 unique words are extracted from the dataset. Out of these words, most of the words at end of frequency spectrum are noise and occur very few times to influence classification. We, therefore, only use top 90,000 unigrams (words) from these to create our vocabulary.

After extracting the words, we represent each tweet as a sequence of a feature vectors in a dense vector representation. We assign an integer index to each word depending on its rank (starting from 1) which means that the most common word is assigned the number 1, the second most common word is assigned the number 2 and so on. The integer index 0 is reserved for the special padding word: indeed, to feed our DNNs, all tweet sequences must have the same length, so shorter tweets have to be padded. The length of all our tweet sequences is 40.
A word having a given index is then represented by a 200 dimensional vector. It turns out that all our 90,000 extracted unigrams belong to the GloVe vocabulary[4], so each of them has a GloVe representation. The special padding word is represented with the zero vector.

4 https://nlp.stanford.edu/projects/glove/

2.4 Models Learning

We used keras with TensorFlow backend to implement the Convolutional Neural Network model.

CNN-A

The dense vector representation we just described above for tweets in Dataset A gives rise to a Embedding layer which is a matrix of shape (v + 1) × d where v is vocabulary size (=90,000), and d is the dimension of each word vector (=200). This matrix has v+1 rows because we have taken the padding word into account. This layer is trainable and initialized with weights taken at random from $\mathcal{N}$ (0, 0.01).

We train our models CNN-A, CNN-B and CNN-C using categorical cross entropy loss with the weight update scheme being the one defined by Adam et. al. The best model we found for training on Dataset A is inspired from a Convolution Neural Network (CNN) described in an open Github project[5]. Let denote it by CNN-A. It has the following architecture:

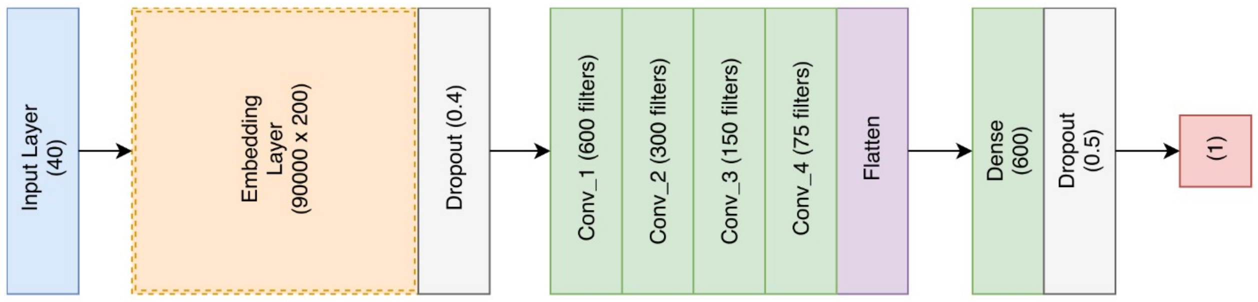

Figure 1: Neural Network Architecture (with 4 Conv Layers)

Validation accuracy: 0.828 (random accuracy: 0.5)

CNN-B

Recall from *Section 2.1* that our training on Dataset B is made by Transfer Learning from CNN-A. CNN-B is the neural network obtained from CNN-A by deleting the last layers

$$\text{dense}(1) \rightarrow \text{sigmoid}$$

from the above architecture and replacing them with

$$\text{dense}(400) \rightarrow \text{relu}$$
$$\text{dense}(6) \rightarrow \text{softmax}$$

We freeze all the weights from the layers we have kept, and train only the added layers with weights initialized at random from $\mathcal{N}$ (0, 0.01).

Validation accuracy: 0.517 (random accuracy: 0.166)

5 https://github.com/abdulfatir/twitter-sentiment-analysis

CNN-C

Recall from *Section 2.1* that our training on Dataset C is made by Transfer Learning from CNN-B. CNN-C is the neural network obtained from CNN-B by deleting the last layers

$$\text{dense}(6) \rightarrow \text{softmax}$$

from the above architecture and replacing them with

$$\text{dense}(350) \rightarrow \text{relu}$$
$$\text{dense}(4) \rightarrow \text{softmax}.$$

We freeze all the weights from the layers we have kept, and train only the last layers with weights initialized at random from $\mathcal{N}(0, 0.01)$.

Validation accuracy: 0.592 (random accuracy: 0.25)

3 Conclusion

3.1 Our results

Once our final model CNN-C is trained, the validation accuracy we get on Dataset C is: 0.592

3.2 Future work

Finding relevant datasets, and train our models on them, took us a huge amount of time.
We were only 2 in our project, so we run very short of time to implement all the ideas we have. If we had more time, we would have been happy to:

- refine our pre-processing methods, and in particular emoticon handling. Some of the emoticons like :((sad) or :@ (angry) could be endowed with a specific polarity, making it possible to map them directly to the right class, instead of merely be considered as negative;
- try other DNN architectures (GRU, (B)LSTM…), compare their performances with the one given in Section 3.1, and finally combine them using ensemble methods like AdaBoost to get better performances;
- pass more time to fine tune each of our CNN, like playing with optimizer, learning rate, momentum, etc.

Despite the difficulties we met, we found this project very inspiring – a very good experience.