Project Checkpoint

Sentiment Classification

Imre **D**elgado & Rustam **T**ukhvatov
{i.delgado.villanueva,r.tukhvatov}@innopolis.university

April 1, 2019

1 Introduction

When it comes to express their very own opinions or ideas, people prefer social networks and virtual environments. This is one of the reasons social media has risen in the last years incredibly fast and why it has attracted so many investment from the public and private sectors. As the amount of this text-like information increases day bay day, companies get highly interested in sophisticated techniques that allow them to extract users' and customers' insights in order to make decisions, to predict outcomes or to generate incomes, for instance, to forecast election's results or to spot tendencies in marketing,business or education regarding some product/item/season. Here is where sentiment analysis comes into play because of the great amount of algorithms and different approaches it offers to tackle those kind of problems.

For this project we focus particularly on data provided by Twitter, a very popular neural network, with the goal of identifying the nature of such text messages. Generally speaking tweet classification deals with labeling comments as negative, neutral and positive, however some labels may extend its classification or add more labels, e.g. very positive and very negative, or they may just consider a binary one, such as neutral and negative, or positive and negative. That being said the problem statement is to detect any kind of negative messages in any given set of tweets, the nature of such negative comments can be related to, but not just or necessarily to, sexist comments, hate speech, racist expressions, bullying, or threatening words.

2 Datasets

2.1 Stanford Dataset

This dataset has tweets which are unrelated to each other, i.e. the topic between them can be different. It consists in the training set, which is a simplified binary version of a larger dataset which more labels, the values 0 and 4 represent negative and positive tweet respectively. The testing part consist again in an small set of tweets with all the original labels, ie. 0,1,2,3,4, however for our purposes some preprocessing will require just to consider negative and positive tweets.

2.2 Twitter US Airline Sentiment

This dataset contains Twitter data on US airlines which was scraped from February 2015. The tweets were classified by the contributors as positive, negative, and neutral. For this particular case the negative reasons involved here were then further categorised regarding the Airline's service, e.g. punctuality, rude service, logistics inside the airport, online help, availability, etc.

2.3 Sexist/Hate Speech Dataset

This dataset contains responses between users and comments regarding social topics in general, however among them there were some tweets classified as sexist, racist or just considered to be offensive for a certain group of people. Such tweets were labeled with 1 while the rest of them were labeled as 0. This dataset has one labeled training set and an unlabeled test set.

2.4 People's Mood Dataset

This dataset consists in comments by the users according to their moods regarding some upcoming event, e.g. before going to/during/after work, a concert, a trip, a haircut, having an appointment, etc. Similarly to the Sexist/Hate Speech Dataset it has a training part which is already labeled, and an unlabeled test set. Given the fact that the the target here is classifying a person's mood, the labels here are: hate, sadness, worry, neutral, fun, surprise, happiness, love, relief and empty.

3 Data Peprocessing

3.1 Tweets Cleaning

Given the fact that there's no restriction besides the number of string characters in social media, any tweet can be published (an it is) with a large number of extra non-textual-related characters (emojis, hashtags, slang words, text shortcuts, etc..) intentionally and unintentionally used, which had a big impact on how the comment was classified. Thus most of the text data (acxtually every dataset) we have will likely need some preprocessing in order for the chosen machine learning algorithm to perform well. When cleaning up the tweets it is a common practice to take advantage of regular expressions, in python we can import the `re` library for getting rid of the tweet mentions, responses between users, symbols, etc. It is important here to point out that every preprocessing is different, and may involve different strategies to get the raw text inside each tweet, e.g. transforming everything in lower case, deleting the mentions, removing any URL or extra spaces, rewriting any word shortcut, and so on, depending of course on the analysis to be performed.

3.2 Data Balancing

In some cases, specially where there is an unbalanced number of elements between classes, e.g. there are more negative tweets than positive ones, or the number of neutral comments outmatches the number of negatives ones, it is highly recommended to handle that imbalance inside the dataset, for instance if a learning algorithm trains with a dataset where the number of negative comments surpasses the number of neutral comments, then there is a high chance that all the predictions will be rated as negative too. There are some alternatives to cope with this situation, using **upsampling**, which consists in taking samples from the class that has less rate allowing repetition to create a bigger set with the same number of elements as the class with more rate or **downsampling** where reducing the number of elements of the class with more rate is the basic idea, one can get generally speaking good results.

4 Related Work

4.1 Word Embeddings

The text (word) embedding is NLP term, which mean transformation text into a numerical representation (an embedding) of the text's semantic meaning. The text is mapped to a vectors of real numbers. In siple case it is used Bag-of-Words, which focuses on the occurence of words in a document. More powerful approach is TF-IDF tecnique. In this case the vectors text represented as ratio of words appearance within a document and across documents. Another efficient method is Word2Vec. Word2Vec is a netword-based embedding model, which improves the predictive ability (surrounding context words) during learning. In this research we use TF-IDF and Word2vec techniques.

4.1.1 TF-IDF

TF-IDF [Rob04] stands for Term Frequency Inverse Document Frequency, it is a method which relies on counting the occurrences of words given a corpus C. This approach appears also in information retrieval problems and it aims to convert the text document into a vector model using the occurrence of words in the text as a non-ordered basis. To be more precise, with the **term frequency**, defined as the count of a term t in a document d, the **inverse document frequency**, defined as the logarithm of the ratio between the total number of documents in the corpus and the number of documents containing the term t. Therefore the product of those terms gives the importance of a term t in the corpus C, since the lower the TF-IDF then the less rarer the word and the higher it is then the rarer the word. Given this description a good practice is to lower case the text content, as many text classification tools rely on counting the occurrences of words. If both upper and lower case versions of the exact same word are found in the document then the algorithm will count them as different words even though the meaning is the same.

4.1.2 Word2Vec

Word2vec [Mik+13] takes as its input a large corpus of text and produces a high dimensional vectors. Each unique word in the corpus is assigned to a corresponding vector in the space. This approach allows to locate close-to-context words in close proximity in space and unrelated words will be far away from each other in space. Typically, these models are implemented using shallow neural networks that learn to restore the language context of words. The main idea of the Word2Vec is to map our one-hot encoded vectors to dense (fixed-size) vectors. There are 2 architectures:

Word2vec was created by a team of researchers led by Tomas Mikolov at Google and patented. In this project we use a model previously trained on 100 billion words from Google News. It includes word vectors for a vocabulary of 3 million words and phrases, the vector length is 300. The training is produced using

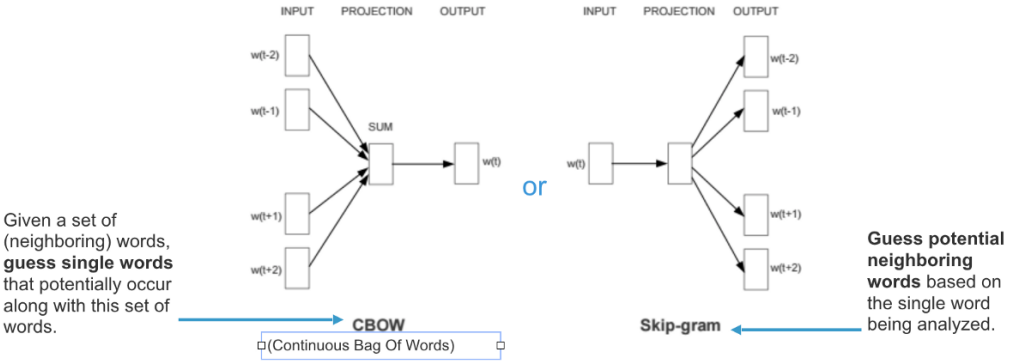

Figure 1: Word2Vec — CBOW and skip-gram model architectures.

the continuous bag-of-words architecture. Words which are not in the set of pretrained words are initialized randomly.

4.1.3 Procedure

in progress ... `jupyter notebook`

4.2 Models

4.2.1 SGD Classifier

Stochastic gradient descent (SGD[RM51]) is an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization. It is called stochastic, because the samples are selected randomly, and not as a single group (as with standard gradient descent) or in the order they appear in the training set.

The standard gradient descent algorithm updates the parameters θ of the objective $J(\theta)$ as:

$$\theta = \theta - \alpha \Delta_\theta E[J(\theta)]$$

SGD computes the gradient of the parameters using only a few training examples, so the new update is given by:

$$\theta = \theta - \alpha \Delta_\theta E(J(\theta; x^{(i)}, y^{(i)}))$$

where a pair $(x^{(i)}, y^{(i)})$ from the training set.

4.2.2 CNN

The next method chosen for tweets classification is a Convolutional Neural Network. The idea is to apply convolutions to the encoded sentence representation with a set of filters, and to take the encoded representations it produces as inputs of the next layers. Of course depending on the filter we apply, the output will either capture the key patterns we are interested in the most. In the context of sentiment analysis the filters will enable us to highlight the intensely positive or intensely negative words. At the same time training the filter's coefficients will help our model build extremely relevant features to feed the next layers. These features work like local patches that learn compositionality. They will enable us to understand the relation between negations and what follows, and similar ones. It will capture relevant information about how the words follow each other. It will also learn particular words or n-grams that bear sentiment information. The features it learns will be location-invariant. It will make a convolution exactly the same way an object that is at the bottom of the frame and an object that is at the top of the frame. This is key not only for object detection, but for sentiment analysis as well.

4.2.3 Procedure

in progress ... `jupyter notebook`

Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	(None, 49)	0	
embedding_layer_dynamic (Embedd	(None, 49, 200)	3062600	main_input[0][0]
Conv_dynamic_3 (Conv1D)	(None, 47, 100)	60100	embedding_layer_dynamic[0][0]
Conv_dynamic_4 (Conv1D)	(None, 46, 100)	80100	embedding_layer_dynamic[0][0]
Conv_dynamic_5 (Conv1D)	(None, 45, 100)	100100	embedding_layer_dynamic[0][0]
MaxPooling_dynamic_3 (MaxPoolin	(None, 23, 100)	0	Conv_dynamic_3[0][0]
MaxPooling_dynamic_4 (MaxPoolin	(None, 23, 100)	0	Conv_dynamic_4[0][0]
MaxPooling_dynamic_5 (MaxPoolin	(None, 22, 100)	0	Conv_dynamic_5[0][0]
Flatten_dynamic_3 (Flatten)	(None, 2300)	0	MaxPooling_dynamic_3[0][0]
Flatten_dynamic_4 (Flatten)	(None, 2300)	0	MaxPooling_dynamic_4[0][0]
Flatten_dynamic_5 (Flatten)	(None, 2200)	0	MaxPooling_dynamic_5[0][0]
concatenate_15 (Concatenate)	(None, 6800)	0	Flatten_dynamic_3[0][0] Flatten_dynamic_4[0][0] Flatten_dynamic_5[0][0]
dropout_15 (Dropout)	(None, 6800)	0	concatenate_15[0][0]
output (Dense)	(None, 5)	34005	dropout_15[0][0]

Total params: 3,336,905
Trainable params: 3,336,905
Non-trainable params: 0

Figure 2: CNN structure from the original paper

4.2.4 BERT

BERT stands for Bidirectional Encoder Representations from Transformers, it was published in a recent paper by Google AI Language researchers. In the field of machine learning it has caused a stir by obtaining very accurate results in a wide variety of NLP tasks, including Question Answering, Natural Language Inference and others. The key point in BERT's state-of-the-art model relies in applying the bidirectional training of a Transformer, which is a popular attention model, to language modelling. It is an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, the Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

The results provided by BERT show that a bidirectionally trained language model can get a deeper sense of language context and flow than the single-detection models already mentioned. The researchers also included a new technique called Masked LM, which allows the bidirectional training in models in which it was previously impossible.

The chart below is a description of the Transformer encoder. The input is a sequence of tokens, which are first embedded into vectors and then processed into the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

4.2.5 Procedure

in progress ...

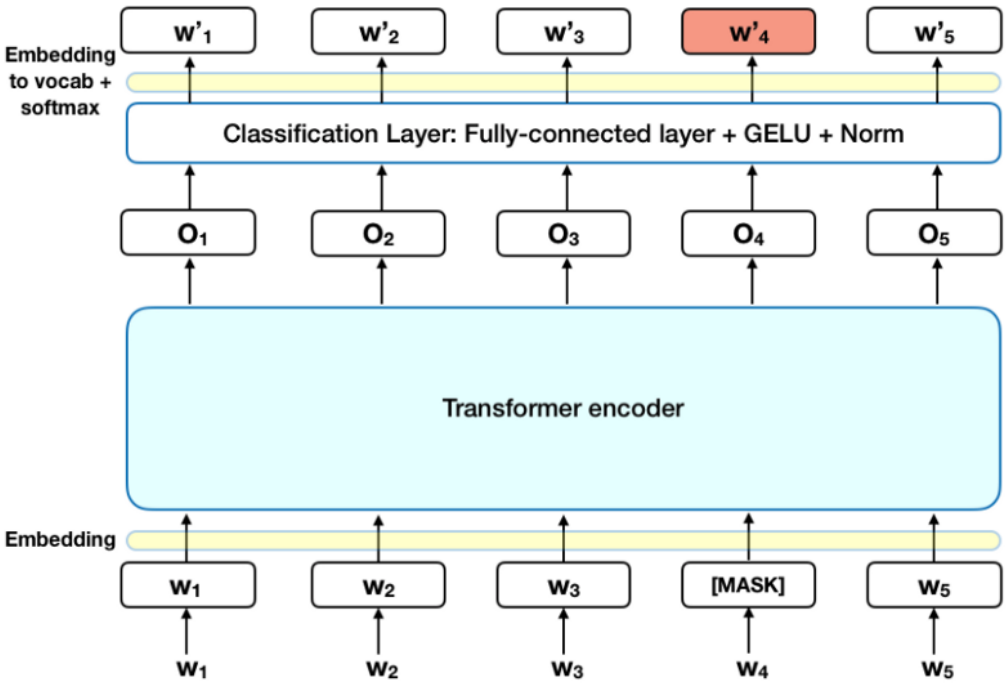

Figure 3: CNN structure from the original paper

References

[RM51] H. Robbins and S. Monro. "A stochastic approximation method". In: *Annals of Mathematical Statistics* 22 (1951), pp. 400–407.

[Rob04] Stephen Robertson. "Understanding inverse document frequency: On theoretical arguments for IDF". In: *Journal of Documentation* 60 (2004), p. 2004.

[Mik+13] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. URL: http://arxiv.org/abs/1301.3781.

[al] Richard Socher et al. *Parsing With Compositional Vector Grammars*. URL: https://www.aclweb.org/anthology/P13-1045. (accessed: April 1, 2019).

[Hor] Rani Horev. *BERT Explained: State of the art language model for Natural Language Processing*. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. (accessed: April 1, 2019).

[Jab] Hafsa Jabeen. *Stemming and Lemmatization in Python*. URL: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python. (accessed: April 1, 2019).

[Kaz] KazAnova. *Sentiment140 dataset with 1.6 million tweets*. URL: https://www.kaggle.com/kazanova/sentiment140. (accessed: April 1, 2019).

[Lea] Monkey Learn. *Sentiment Analysis, nearly everything you need to know*. URL: https://monkeylearn.com/sentiment-analysis/. (accessed: April 1, 2019).

[Spa] Cambridge Spark. *50 free Machine Learning datasets: Sentiment Analysis*. URL: https://blog.cambridgespark.com/50-free-machine-learning-datasets-sentiment-analysis-b9388f79c124. (accessed: April 1, 2019).

[Vic] Rebecca Vickery. *Classifying Tweets for Sentiment Analysis: Natural Language Processing in Python for Beginners*. URL: https://medium.com/vickdata/detecting-hate-speech-in-tweets-natural-language-processing-in-python-for-beginners-4e591952223. (accessed:April 1, 2019).

[Wik] Wikipedia. *Sentiment analysis*. URL: https://en.wikipedia.org/wiki/Sentiment_analysis. (accessed: April 1, 2019).