# Sentiment analysis 2

Alessandro Grassi 221224
University of Trento
alessandro.grassi@studenti.unitn.it

## Abstract

*The work focuses on the well-known task of sentiment analysis. This represents a relatively important task in Natural Language Processing since automating the retrieval of customers' opinions has great value for a lot of businesses. This report explains two ways of performing SA: at sentence level and at document level. It shows also the performance obtained and the results over various techniques.*

## 1. Introduction

Sentiment analysis is useful to a wide range of problems that are of interest to human-computer interaction practitioners and researchers, as well as those from fields such as sociology, marketing and advertising, psychology, economics, and political science. What sentiment analysis, a.k.a. Opinion Mining (OM), tries to do is to classify a span of text into positive or negative and in some cases in objective and subjective. For objective it's meant statements that does not express any opinion, they just express facts. Subjective statements express a judgment about someone or something, these subjective statements are the ones that weight for the positive/negative classification.

Usually, the purpose of Sentiment Analysis is to analyze reviews to some product. There are several depths of knowledge that can be mined from the review. Some algorithms, usually driven by machine learning, can recognize the target of the opinion and understand its polarity [1] and even understand which component of the product reviewed need to be improved to get better rating. The algorithms proposed in this paper analyze just the polarity of the entire review. One method is built with a GRU, which is a variant of the vanilla RNN that leads to better accuracy and handle longer sequences. This GRU analyze the entire review and return a prediction for its polarity: positive or negative. The other proposed approach uses a lexicon-based algorithm called V.A.D.E.R. [2]. V.A.D.E.R. is used to obtain polarity of single sentences, then these polarities are summed up and the final score define the polarity of the whole review.

As will be explained later in the report the GRU approach leads to higher results.

## 2. Problem statement

Given a sequence of tokens $< t_1, \ldots, t_2 >$ and given a polarity *p*, that can be *positive* or *negative*, we want to find the most likely *p* such that:

$$p = \underset{p}{\operatorname{argmax}}\, P(p|t_1, \ldots, t_n)$$

The *P* distribution has been approximated with a GRU in one approach and with VADER algorithm in the other.

## 3. Data analysis

The dataset used is called IMDB [3] and it is provided by Pytorch. It is composed by 50'000 reviews with its sentiment ground truth, half of the dataset has been used for training and the other half for testing. The ground truth provided refers only to the entire review, a.k.a. document, and not for the single sentences, so only a document level sentiment analysis machine learning model could be trained with this dataset.

Pytorch provides an iterator for both testing and training, the dataset is iterated so to build an array of dictionaries containing the review and its sentiment. For document level analysis each review is preprocessed with the following steps:

1. Lowercase every token
2. Remove html tags left from the web scraping phase
3. Remove punctuation
4. Remove stop words
5. Lemmatize tokens

For the sentence level SA only the html tags removal is performed.

Before being fed into the GRU the data need to be padded/truncated and vectorized. The padding consists in adding zeros to the array of tokens so to reach a desired length, truncating keep just the desired length of a review that is longer. The GRU does not require that every sequence is of the same length, but it is useful to train the network in batches. The vectorization is the process of transforming each word into a vector of numbers. I used the vectorizer from spacy with the model *en_core_web_lg*.

| Approach | F1 score | Compute time |
|---|---|---|
| Document level with LSTM | 0.83 | 55 min |
| Document level with GRU | 0.88 | 50 min |
| Sentence level with V.A.D.E.R. | 0.71 | 1 min |

Table 1: best results with the three main approaches. The compute time provided refer both training and validation for document level approaches. Pre-processing phase is not included in any compute time.

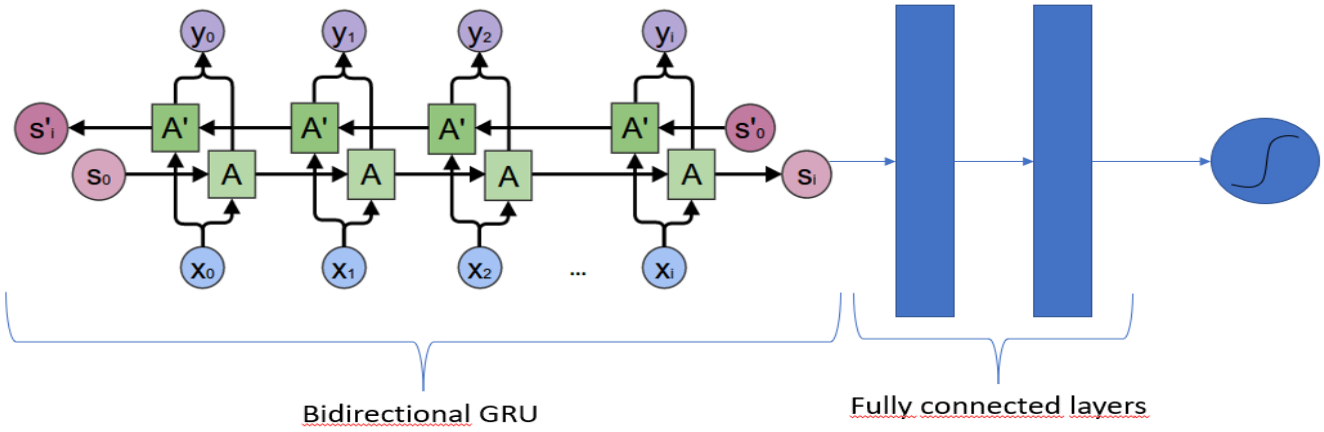


Figure 1: Neural network model used to compute sentiment polarity of entire sentence. It consists of a bidirectional GRU that output a hidden vector that will be classified by two fully connected layers and a sigmoid function. The GRU scheme has been taken by http://colah.github.io/posts/2015-09-NN-Types-FP/

## 4. Methods

As previously anticipated the Sentiment Analysis task has been solved in two ways, at document level and at sentence level. The first one with a machine learning technique and the second one with a lexicon-based algorithm. Each approach will be well explained in the following.

### 4.1. Document level approach

The approach consists in training a GRU [4] combined with two fully connected layers for classification of text into sentiment polarity. A GRU is a variant of the vanilla RNN. A Recursive Neural Network (RNN) is a deep learning model that call recursively the same units, this allows the model to process sequences of variable lengths and keep track of information present before (or after for bivariate RNN) in the sequence. The problem with vanilla RNN is the drop in accuracy when sequences become too long. To overcome this problem two alternative models have been proposed over the years. One is LSTM (Long Short Term Memory) [5], LSTM introduces the concept of a memory cell that stores information of previous states, this help to keep data of previous element in the sequence, even far from the current one. With the cell three gates are also introduced that modulate data flow with a sigmoid: an input gate that modulate how much of the previous output keep, a cell gate that modulate how much of the previous cell state keep and an output gate that modulate how much of the current output forward to the next recursion. All the gates' parameters are learned through backpropagation. The LSTM approach has been the first trial to solve the Sentiment Analysis problem, it provided a good f1 score of *0.83*. Although it provided a good score, I decided to try the GRU model, called SentimentGRU, since it has less parameters to be learned making it more resistant to overfitting. The GRU is a RNN where the previous hidden output and the current hidden output are modulated through sigmoid gates, no memory cells are involved. The GRU achieved a better f1 score: *0.88*. The implementation of both GRU and LSTM is bidirectional, this helps to keep track of both previous information and future information. A bidirectional RNN simply run forward the sequence in

both ways and then combine the results. To complete the classification the GRU is not sufficient by itself, the network just provides a hidden representation of the sequence, to get a classification I built two fully connected layers that output to a single sigmoid: 0 is considered negative and 1 is considered positive. To have better performance each word has been vectorized with a vectorized trained by Spacy on *en_core_web_lg* language model.

Since each token is transformed to a vector of dimension 300 the amount of RAM needed to keep all the dataset vectorized is above 15 GB. To overcome this issue the vectorization is performed every time the review is requested by the Dataloader. This slows down the process of training and evaluating because the vectorization needs to be performed for each token and for each epoch.

### 4.2. Sentence level approach

This approach consists of computing a score for each sentence of the review and combine all these scores to classify it. Since the dataset is not provided with a ground truth for each sentence, but just for the entire review, it is not possible to use a supervised machine learning approach on these data. One solution would be to train another GRU (the previously described GRU could be recycled for this task) on a different dataset that provide sentence polarity. Such model could then be used to define the polarity of subjective sentences. Subjective sentences are the ones that provide an opinion. Objective sentences can be filtered out by V.A.D.E.R.. I tried this approach but the time for both training this second GRU and to analyze each sentence required more than four hours using Google GPUs on Google Collab making this approach not feasible.

To have a fast and reliable sentence level sentiment analysis is to use VA.D.E.R. It is a lexicon-based algorithm that is capable not only to recognize if a sentence is positive, negative, or neutral, but is also capable to give a score about how much of its polarity has. The score ranges from -4 to 4. After scoring each sentence I needed to aggregate the values somehow to get a general polarity on the entre review. I tried two different methods:

1. Classify positive the review that has more positive classified sentences than negative, classify it negative otherwise.
2. Sum up the scoring of all the sentences and classify the review as positive if it has a score greater than 0, negative otherwise.

The first approach gets a f1 score of *0.55* and an accuracy of *0.60*. It has to be noted that the f1 score is much higher for the positive class: *0.71*, and really bad for the negative: *0.39*.

The second approach shows more reliable results with a f1 score of *0.71* with an accuracy of *0.71*. Also here the positive class is more accurate, but the difference is only of *0.09*: *0.75* for positive, and *0.66* for negative.

## 5. Results

For the document level sentiment analysis, I performed an intensive hyperparameter search to find the combination that get the best performance on the GRU. The final configuration is a GRU with 3 hidden layers, a hidden dimension of 256 units and a probability of 0.5 for dropout regularization. The feed forward classifier has two layers with 256 units, leaky ReLU as non-linearity and a probability of 0.3 for dropout regularization. Only changing the number of layers in the GRU consists in a significant change in accuracy and time requested for a forward and backward pass.

For the sentence level sentiment analysis there were no hyper parameters to be tuned because no learning algorithm are used. To obtain the best accuracy the sentence split has been performed with Spacy's senter trained with *en_core_web_lg* model.

## 6. Discussion

I report two algorithms capable of classify the sentiment polarity of movie reviews. The best score has been achieved by analysing reviews at document level, but there's no proof that it is the best way to do opinion mining. Indeed I was provided with labels for the whole document, not for single sentences. Even if it is unlikely to me, a model trained with single sentences could outperform SentimentGRU. I do not think that it is the case because a GRU that process the whole document could keep track of information from multiple sentences leading to a better understanding of the text.

## References


[1] Minghao Hu† , Yuxing Peng† , Zhen Huang† , Dongsheng Li† , Yiwei Lv§. Open-Domain Targeted Sentiment Analysis via Span-Based Extraction and Classification.

[2] C.J. Hutto, Eric Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text

[3] https://pytorch.org/text/0.8.1/_modules/torchtext/datasets/imdb.html. Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher. Learning Word Vectors for Sentiment Analysis. 2011

[4] Guizhus Hena, Qingping Tana, Haoyu Zhanga, Ping Zenga, Jianjun Xua. Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions. 2018

[5] Sepp Hochreiter, Jürgen Schmidhuber. Long Short-term Memory. 1997