Movie Rater and Classifier

Saar Egozi (ID. 204268940)

Submitted as final project report for Deep Learning, IDC, 2019

1 Introduction

The steep climb of Neural Networks' technology opened the world to many exciting and helpful solutions to aid humans with daunting day-to-day tasks, which otherwise would require a great deal of time or resources (typically both). The common ones are object detection and recognition - problems that were considered impossible not too long ago to be done correctly by anything other than humans, in real-time at least. Today of course these problems are solved, and utilised in the upcoming autonomous vehicles [4, 2]. But image proccessing is not the only thing NN can help us with. One of the most common usages of the NN architectures is the process of natural language (NLP). The work regarding the world of NLP is vast, and designs of new models of networks are born every year, either improving training time, prediction results and accuracy, or prediction time (latency).

There are many example of how those NLP networks are used, starting from the simple problems of text-classification and regression problems, and continuing with text-generation. My project is encountering exactly these two, at once. I've decided to build a network with summarised plot information of a movie information, and then let it try to predict both the genre and the score of the movie. The summarised information is any summery of the movie, constructed of usually no more than 8 sentences. The motive for such a task was the following: with a sufficient amount of data, and given there is in fact some connection that is too hard to detect by humans between a movie's plot and its final public score, one could simply type its idea for a film in several sentences, and see if there is a future to his work.

The genre classification is less glamorous obviously, but mostly arouse academic interests: it is usually a pretty easy task to guess the genre of a movie based on its script, but apparently most movies fall underneath several categories, typically three or four. Such classification is significantly more difficult, even for humans. Therefore it might be interesting if a machine could pick on signs that usually eludes human eyes.

1.1 Related Works

Other than the mentioned works in the bibliography, I mainly used several blogs and tutorials to aid me with my work. The idea itself of both classification and regression is composed of several basic concepts, but I didn't find a relevant specific paper that would help me with my task.

2 Solution

2.1 General approach

The general idea of the network is quite simple. The network itself is based on a simple text classifier model, including both word embedding and multi-label classification, with multi-task based training and predictions.

The datasets in my disposal were limited. To save time and avoid scraping the data from the internet, I searched for movies' reviews and scoring in several datasets search engines (the most rewarding one was Kaggle) It turns out there are no datasets composed of both movies' overviews tagged with IMDB scores, probably because the relation between the two might be very subtle. (see 2.2) To overcome such difficulty, I've used four datasets in total: two containing movies' overviews (along with additional unrelated metadata) [5, 1], and two IMDB datasets containing the scores together with yet additional details such as cast, nominations etc... [3, 6] The main challenge with it was the pre-proccecing stage, which described in full in 2.2.

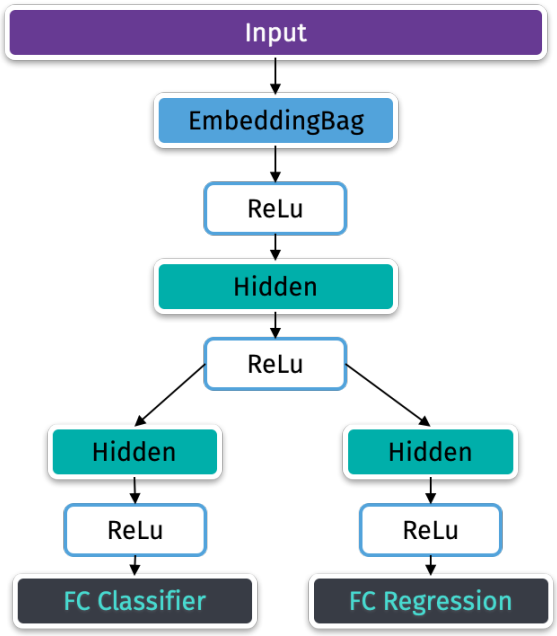
My project revolved around two main questions: (1) **How good can we approximate a movie's potential score** when given a mere short description of the movie's plot, and (2) **How is the task of training a single network to perform both tasks** (multi-label classification along with regression) is evaluated against two networks, each given a single task to handle.

2.2 Challenges and Solutions

Like many other problems that are chosen to be solved using NN, I as well encountered the difficulty to gather enough data to train the network. Another very significant challenge is the correlation between the input and the output of the network. The main issue is loss of information. A great movie could be based on a mediocre plot, great actors, amazaing visual effects, and captivating music. The link between the score and the input is supposedly very weak, but I've decided it would be interesting to see if the network could find some link, even if not a great one.

The next challenge was how to construct my network's dataset. I've found two main datasets containing the IMDB scores of movies, and two containing the plots and overviews, but none contained both. The main difficulty was to try and match the movies, with the movie's title as its only identifier. For that I've had to filter both sets, by "cleaning" the titles as much as possible without losing information, and then matching them. The technical details can be found in the code itself.

Figure 1: Network's design



2.3 Design

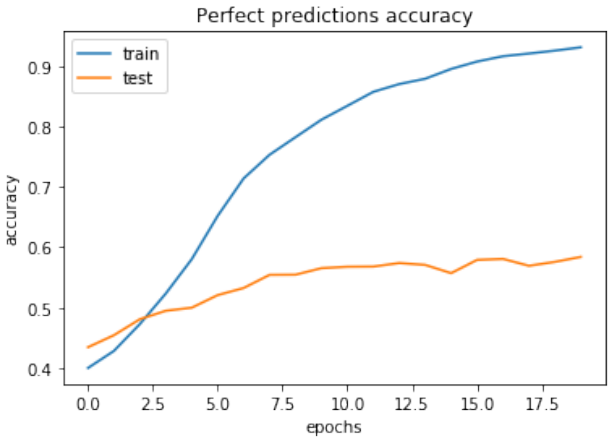
As mentioned, my model was based on a basic text classification network. Since the input is read as a whole, there is no need for recurrent information or any attention modelling of the network. The network itself is composed of an embedding layer, another hidden layer, and then split into two hidden layers, and each goes to its corresponding output layer, one for classifying the genres and one to predict the score. Between the layers we used a ReLu activation function, which seems to improve the results of the network.

3 Experimental results

General parameters: During the process of developing the network I examined several alternatives to the network, in order to extract the main causes of the results I was getting. There are several parameters that I used across all my experiments: **Data split** - 0.85 of the data was used for training; **Batch size** - 16; **Number of genres** - 24, full list in colab notebook; **Optimizer** - Adam; **Learning rate** - 0.01; **Multi-label classification loss function** - 'BCEWithLogitsLoss'; **Regression scoring function** - Huber Loss ('SmoothL1Loss').

Phase 1: Testing how much data is really needed for the simple classification task. To try and solve this issue, I used 2 datasets: the first was the complete dataset I had, that had both scores and plots; the second was only a partial dataset, i.e without a matching IMDB score. Luckily the original datasets containing the overviews also included genres, so I could use them. Overall - the complete dataset was composed of ˜7,000 items, while the partial included ˜20,000 items. At first,

(a) Perfect prediction accuracy



(b) OneOf prediction accuracy

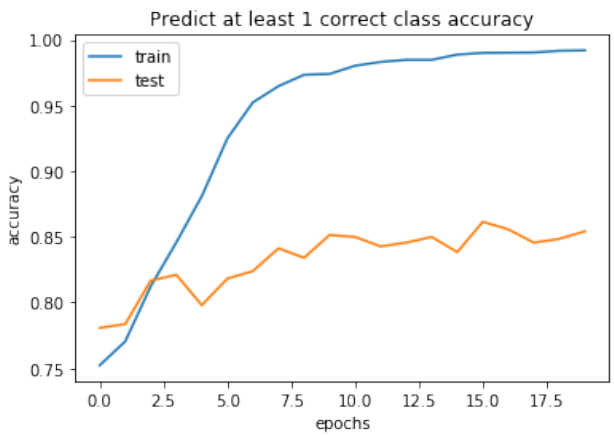


Figure 2: The accuracies of the network in both train and test. 20 epochs. Measurements taken after each epoch

I used a naive architecture for the network, including only one hidden layer and an output layer. The embedding dimension I used was small - only 32, and the hidden size was 100.

I used two measurements of estimation: *Perfect prediction* and *OneOf prediction*. To measure perfect prediction rate, I counted only the times where the net predicted the exact genres (all of them). For example: Comedy, Animation and Family for "Toy Story". To estimate OneOf prediction's success rate, I counted as a success each time the network predicted at least 1 correct genre.

After 7 epochs, the network reached ˜61% perfect prediction accuracy, and ˜91% OneOf accuracy. The results were similar for both datasets, as well as when using a more complex network: larger embedding size (150), larger hidden layer (450), additional smaller hidden layer (250).

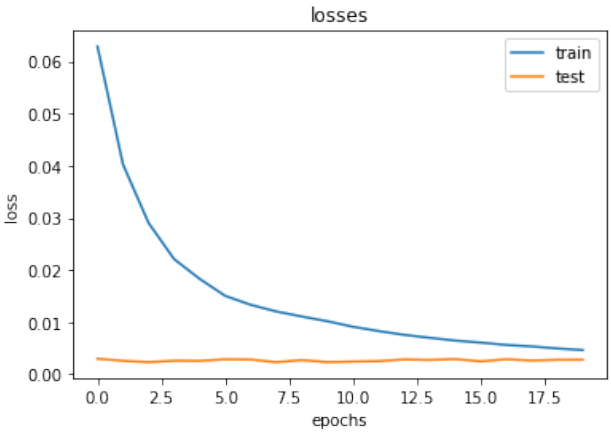
Phase 1 results: (1) Movie plots are rarely enough to try and predict all of a movie's genres (a task that is difficult for humans as well), but a 60% success rate for a perfect prediction does suggest the task in several cases might be reasonable, if eliminating the more vague genres (e.g. "Short" which is hard to guess from the plot alone). (2) There was sufficient amount of data - a dataset of almost three times the size yielded similar results. (3) The data gathered from the text itself does not require a very deep or large network, even though the task is hard. It doesn't seem as there is much more to extract from the given input.

Phase 2: The next step was to see if the network's capabilities could be preserved when introducing another task inside the same network. The design I used for the net is the one shown in Fig 1. I kept the large sizes of the hidden layers in order to allow more flexibility.

The results of the final experiment are shown in Figs 2,3

From these results we can see that the network's capabilities for classifying were not preserved entirely. The more interesting result however is the IMDB mean score error. This measurement is

(a) Network loss



(b) Mean error of IMDB score prediction

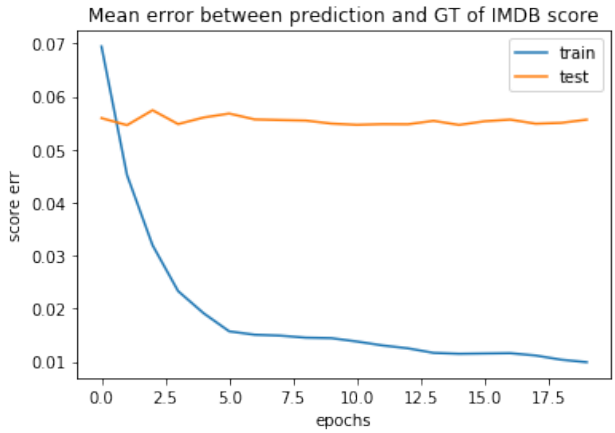


Figure 3: The losses and mean error when predicting the IMDB score

simply the mean distance between the actual IMDB score to the one the network predicted. We see that the network can predict with an error of a ˜0.06, which is remarkable given the small amount of data (a mere plot - a movie summarised in a few sentences). These results are a cause to suspicion: the fact that an IMDB score could be calculated in a precision of 0.06 error from only an overview does not seem realistic. It might be that the measurement I've used is misleading, like averages sometimes do, and the network was tuned to estimate some kind of an average to movies scores. I elaborate further in the Discussion section regarding these results.

4 Discussion

Following the described experiments and their results, several main conclusions could be drawn:

Genre Classification is a difficult task, and predicting a complete genre set for a certain movie based on its overview alone might be too presumptuous. I tested several networks' architectures and several different sized datasets, and saw no significant change in the network's success rate. Since we know such task is hard even for humans, given some genres are not always tagged exactly the same, and movies' overviews could be written in different fashions, we tend to think that the input simply might not contain enough data to succeed perfectly. We did see that at least one correct genre is predicted with a promising success rate (90%). Such results suggest that at least one genre could be classified correctly, and the same is probably true for humans as well. The results show the network could predict genres perfectly with a success rate of 55%-60%. Usually such numbers are considered bad for classification, but for multi-label classification with a somewhat vague-natured task those results might not be all too bad. Of course such claims must be verified using larger datasets, and perhaps some human experiments to confirm such task is indeed hard.

Score Prediction should have been a very difficult task, and my initial hypothesis was that the correlation between the overview and the IMDB score is thin to non-existent. The results the network yielded were surprisingly good: a mean error of 0.06 (when scores are in the range of [0,10]). The immediate thought regarding these results is that the dataset itself is just not vast enough (7,000 samples), or the real scores are not scattered enough, and a simple mean value might yield similar results. It may be interesting to see the results using a larger dataset, or try to develop a heuristic algorithm and see if the results were in fact extracted from the given texts, or could be just as well be derived from statistics.

Hybrid Solution was an experiment I was interested in for research purposes only. It was interesting to see that even though I added more hidden layers and tried to allow some "manoeuvring space", I could not reach the same success rate. The conclusion that could be drawn from this experiment is that better results could be achieved by a single-purposed network. With good usage of parallel computing training two networks might be even less time-consuming. To sum up, in this kind of problem there is no real advantage in using a single network to solve two unrelated tasks.

5 Code

Colab Notebook

References

[1] Rounak B. The movies dataset, 2017.

[2] A De La Escalera, L Moreno, EA Puente, and MA Salichs. Neural traffic sign recognition for autonomous vehicles. In *Proceedings of IECON'94-20th Annual Conference of IEEE Industrial Electronics*, volume 2, pages 841–846. IEEE, 1994.

[3] Orges L. Imdb movies dataset, 2016.

[4] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[5] Justin R. Wikipedia movie plots, 2018.

[6] Yueming. Imdb 5000 movie dataset, 2017.