INF554 – Machine and Deep learning Data Challenge

Joe Khawand – Victor Morand – Paul Woringer
Kaggle team: FreedomOfSpeech

Introduction

Our task in this challenge was to implement a machine learning algorithm to predict the number of times that a tweet will be retweeted using concepts from the class INF554 – Machine and Deep learning. This challenge has been organized in the form of a Kaggle competition opposing teams of three students from this class. The loss function used to measure our performance is the Mean Absolute Error:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |p_i - a_i|$$

where p and a are the vectors of predictions and real retweet numbers, respectively.

I – Feature selection/extraction

A – Basic features extraction

- **mentions** could be discarded straightaway since it was equal to zero for all entries.
- **TweetID** could also be discarded as it contains no valuable information on the tweet.

We selected a couple of numerical features without changing them:

- 'verified': A one-bit encoding of whether the author of the tweet is verified
- 'followers_count'
- 'friends_count'
- 'favorites_count'
- 'statuses_count'

From the 'timestamp' feature we extracted:

- The day of the week
- The time of day

This is relevant because on social media publishing at certain times, when there is peak activity on the platform, can lead to higher visibility for users' posts.

B – Preprocessing of hashtags

We also thought it relevant to extract features from the content of the tweet, including:

- The number of urls
- The number of hashtags

In preprocessing hashtags, we first counted the **number of times that each hashtag was used** in the training set to measure their popularity. Then we built a one-hot encoding of the presence of some of the 15 most popular hashtags, adding 15 features to our data.

C – Preprocessing of Text

The text provided has already been through a first step of **removing the stop words**, which contain very little information and can often be discarded.

The vectorization method proposed in the data challenge baseline programs is a **TF-IDF vectorizer**. Term Frequency – Inverse Document Frequency is a formula that aims to define the importance of a keyword or phrase within a document or a web page. This yields more interesting results than plain one-hot encodings as it scales down the importance of terms that appear frequently throughout the dataset and are therefore less informative than words that appear on a small fraction of the corpus.

Fig. 1: Word cloud of popular tokens

We used this TF-IDF vectorizer to get 100 additional features and get broad information on its content, but that gives too much information to the text relative to the more basic features. We have therefore implemented **NMF embedding** to reduce the dimensionality of the representation of the text, while keeping a maximum of information.

II – Model Choice, Tuning, and Comparison

1 – Looking at the data

The most important feature seems to be the **number of favorites**, which is highly correlated to the number of retweets (a simple linear regression between the two yields MAE=7.65 on the training data).

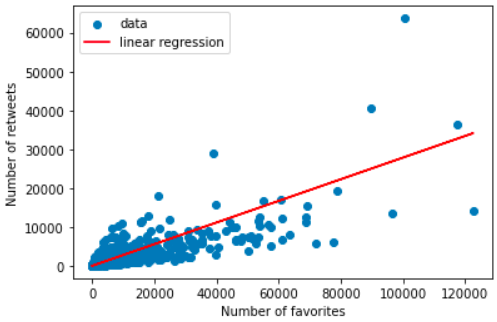

Fig. 2: Correlation of favorites and retweets

There is a high proportion of data points with very low retweet counts (49% have 0, and 93% less than 10). It is easy to have a relatively low MAE on these points, but the few **points with very high retweet counts cause the MAE of our models to explode**. Therefore, in our later attempts, we thought it appropriate to classify the data in two or three categories: those that will probably get no retweets, and those that will probably get at least one. This allows us to make a model more specific to the points with high retweet counts with a better accuracy, as it is no longer skewed by the others that represent most of the dataset. We use the RandomForestClassifier from Scikit-learn to do this classification, and then another model, like a neural network, to predict the retweets count on each category (except for the category which is expected to have 0 retweets).

2 – Exploration

In a first approximation, we attempted to implement a simple **linear regression** using the linear model from scikit-learn, with only the **basic features and hashtags**. This proved to be a slight improvement from the baselines using a constant estimation of zero (MAE = 15.5) or the mean value (MAE = 26.1), with a resulting **MAE of around 10**. However, this was still nowhere near the results achieved by the other teams (MAE around 6 at the time), or even the linear regression using just the favorites count. This is in part because this linear model from scikit-learn can only be optimized with the mean square error as a loss function and is therefore not adapted to our situation. However, that quick implementation gave us some idea of the impact of each feature on the result.

We then tried to add features representing the **text**, using the TF-IDF vectorizer and NMF dimensionality reduction, and using this new data set as input for a **neural network** with 3 fully connected layers.

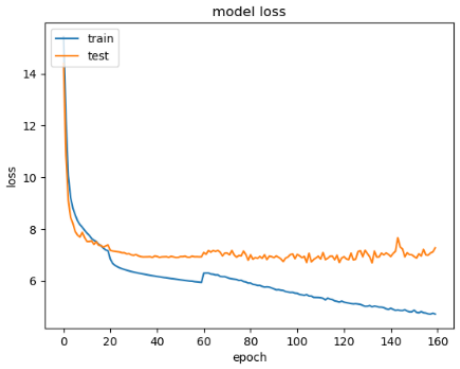

Fig. 3: Training history of the neural network

We first note that the loss on the training set oscillates highly throughout the training. To improve the stability of our model, we chose to implement **batch normalization**, which yields much smoother learning curves without affecting our results. Another issue we encountered is that we could not reach obtain good predictions on our training set without overfitting the data, as seen on the learning curves displayed above (with three different learning rates used along the training of the model). We then added dropout during the training, which greatly reduced this overfit. However, we did not manage to drop the MAE below 6.

We then attempted to use the Scikit-learn **RandomForestRegressor** and **GradientBoostingRegressor** on the whole data set. We experimented with the various hyper parameters using random grid search to get an overall sense of the best ones and then local grid search to improve our results, both on the data set with the text and hashtags and on the data set with only the most basic features, but we did not manage to do any better. Also, this was very computationally expensive to do, but using the libraries like **HistGradientBoostingRegressor**, optimized for large datasets, did not do much to improve our results.

We noted that in the case of neural networks, using **normalized data** (either reduced and centered or shifted to have values range from 0 to 1) was most effective and allowed for more stability and better results. However, for algorithms that rely on decision trees, using the original data proved to be more efficient in our case.

3 – Final stages

Finally, we opted for the two-step approach of first classifying our data, and then training a model on each category.

We tried with two categories (tweets that we expect will get zero retweets, and tweets that we expect will get at least one) and with three (0 retweets, 1-10 retweets, more than 10). In the category expected to have zero retweets, we simply use constant prediction of 0, and for each of the other categories we train a model.

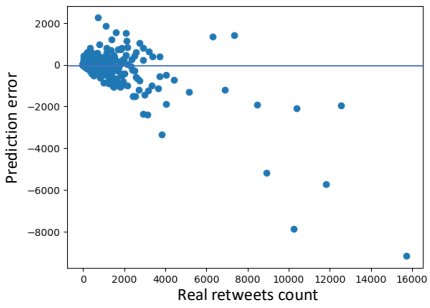

Fig. 4: Error estimation on predictions

	[0]	[1-10]	[>10]
[0]	24492	10235	0
[1-10]	4142	26100	672
[>10]	0	864	4289

Fig. 5: Confusion matrix on test set

We achieved an accuracy of 0.8 for classification, but the confusion matrix gives more information. It is more of a problem in our metric to have data wrongfully assigned to the category with zero retweets than the other way around, since the prediction models on the other categories can fix that mistake. Also, we note that there is no confusion between the first and last category, which would have been the source of terrible predictions.

In the end, we tried to use either a neural network or a gradient boosting regressor as a model. With the neural network, we reached a MAE of 6.1. Fig 4 shows that the error lies mainly on data points with high retweet values, for which we tend to underestimate the result. With the gradient boosting regressor, we were able to reach a MAE of 11.65 on the category of at least one retweet, and overall slightly over 6.