Retweet prediction - Kaggle project

Dannel Cassuto, Benjamin Hebras, Fabien Roger
Team *dbf.chr*

December 2022

1 Feature Selection/Extraction

1.1 Analysis of Candidate Features

Here are the features we used or constructed :

favorites count (log-scale)	number of hashtags	text length
followers count (log-scale)	time since release	trigger words (empirical selection)
statuses count	hour	sentiment analysis (NLKT)
friends count	weekday	Word2Vec (Gensim)
mentions	td_idf	verified account
URLs (binary)	sentence_embedding	word length

The correlation between **favorites_count** and the number of retweets is roughly 0.8. As such, this is our most important features. For popular tweets, the relationship between log retweets and log favorites seems to be roughly linear (see Figure 4).

1.1.1 Text-related features

Since the correlation between favorites and retweets is so high, the most informative pieces of data were the ones which informed us if the tweet had been very retweeted *compared to what we would expect given the number of favorites*. This is why we investigated the influence of the presence of some trigger words such as "rt" or "fav" (case-insensitive), which might lead users to retweet or favorite.
The sentiment analysis features were constructed by summing the *negative score* and the *positive score* of the NLKT[2] sentiment classifier (a neutral, a positive and a negative score that sum up to one). The idea is that tweets that are emotion-dense might be more likely to be retweeted than neutral tweets (relative to their number of favorites).
We also explored the non-stop words with high frequency by using TD-IDF, which extracts how frequent a word with relation to the general corpus.
Finally, we explored pretrained sentence embedding. We settled for a sentence Bert model, which uses a Transformer trained on many different tasks to encode sentences in 512-dimensional vectors[3]. The particular model we used was `distiluse-base-multilingual-cased-v1`, a distilled and faster to run version of Bert, which is able to process French text.

1.1.2 Time-related features

Old tweets might be more likely to be more popular, which is why we investigated the time since publication. But the effect was even stronger than we anticipated, which might be due to the way tweets were collected: which might have also selected old tweets recently retweeted. See Figure 3.
Given that hours are periodic, we added the sine and cosine of hour / 24. These functions can then be linearly combined into any 1-day periodic sinusoidal functions, and an MLP can even make sharper features encoding whether we are in a small period of the day using only 3 weights. This was inspired by the use of sinusoidal functions in the positional embeddings of Transformers[4].

1.2 Empirical Evaluation of Feature Importance

First, we tried the different text embedding ideas separately. None of them yielded measurable improvements over the model without those (see Table 1) while being much more expensive to run. Sentiment analysis proved unhelpful because almost all tweets were classified as "neutral". These failures might be due to some anonymization scheme that might have altered the text in a way that makes most tweets unreadable.
We then evaluated the importance of the other features which seemed to be relevant and some triggers words by measuring for each feature the loss when it was left out of the set of features fed to the MLP. Overall, only **favorites_count** has a large effect, though some others are also useful to increase performance, such as **normed_time**, the log of the time since the tweet was published.

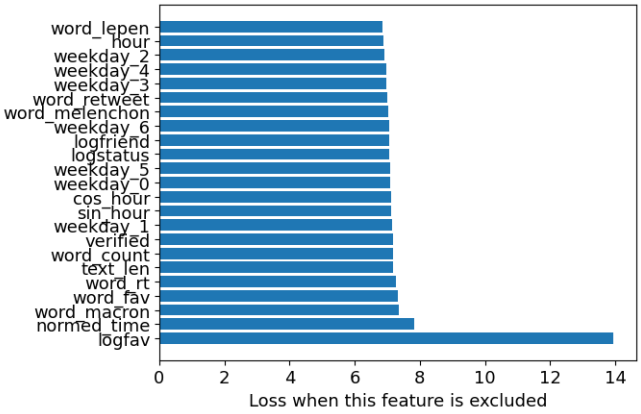


Figure 1: Features importance

In the end, using this data and some additional trial and error, we kept the following features for the MLP: the log of the **favorites_count**, the log of the number of friends, the log of the number of status, the word count and character counts, the **normed_time**, the verified boolean, and the presence of the word "rt". For the Decision Tree, we also kept the hour feature.

2 Model Choice, Tuning and Comparison

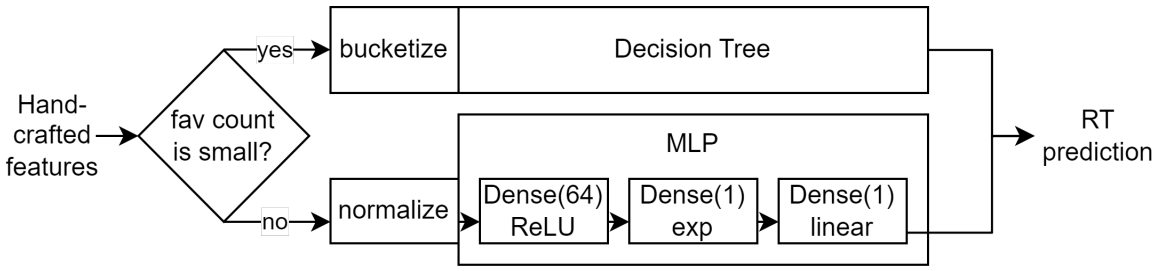
2.1 Our Best Model: a Two-Phase Model

2.1.1 The Reason Behind a Two-Phase Model

Our best model outputs either the result of a decision tree on inputs with a low number of favorites or the one of an MLP on inputs with a higher number of MLP. Indeed, the decision tree has a structure that is easily adaptable to the minimization of the MAE (using a greedy search algorithm) and produces good results on low number of tweets since it will always produce a prediction which is an integer (the median of integers is an integer). This goes well with an MLP for tweets we suspect to have larger amount of retweets, since MLP are better than decision tree at predicting continuous distribution of values. The MLP is trained only on data with many favorites, which considerably reduces training time and avoids learning patterns which do not apply to very popular tweets. This is similar to the idea used by [1].

2.1.2 Architecture

Because the favorite count correlates very highly with the retweet count, we chose it as decision criteria.
The Decision Tree uses the MAE as decision criteria, and operates on bucketized features. The number of buckets was determined after assessing their relative importance during our data analysis. Contrary to the many-favorites regime, using the "hour" feature proved helpful.
The relationship between the log number of favorites and the log number of retweets seem to be roughly linear, which is why the input to the MLP is the log number of favorites, and why we chose to apply an exponential activation function at the end of the MLP. The final linear layer enables the network to cheaply add an offset and a scaling factor.



2.1.3 Attempts to improve this model

We tried to improve the performance of this two-phase model by replacing the Decision Tree with a KNN classifier, which ended up being a slightly less good and longer to run solution.
Also, we tried to add a third phase, to try to address the outliers (very high number of retweets) better. we used the same idea as for the 2-phase model, but we using a linear model on the data above a certain favorite_count threshold, to avoid overfitting on the most difficult examples. We tried a few combinations of threshold and picked the best (see Table 2), but none were better than the 2-phase mode.

2.1.4 Choosing the Hyperparameters

We ran a search over many possible values of the critical number of favorites above which data is included in the MLP training data, and the critical number above which the prediction is the one made by the MLP. Results are in Figure **??**. Though our experiments on validation data sometimes suggested benefits of training on more data than we evaluated it one, those benefit did not translate to better submission scores, since we were then able to train on more data for the final submission.
We also ran a grid search to find the optimal number of neurons in the hidden layer of the MLP, and the amount of L2-regularization to be applied to the large kernel of the first weight matrix (see Figure **??**).

2.2 Comparison with baselines

We used the following baselines:

- **k-NN**: We used a k-NN and scaled the dimension of the number of favorite, as it is much more important than the others. See Figure 6 for our hyperparameter search.
- **Linear Regression**: We used a simple linear regression using the features used with the MLP. We searched over different L2-regularization values, and predicted the log number of retweet (Figure 7).
- **Decision tree alone**: We used the linear tree used as the first phase of our two-phase model.
- **MLP alone**: We used the MLP used as the first phase of our two-phase model.

The results are the MAE on a validation set, which is 30% subset of the data. The training data is the remaining 70%. (Some methods being very slow to run, cross-validation was too expensive.) We used the whole dataset for the final submission.

Model	MAE
2-phase model	**6.6**
3-phase model	6.9
k-NN	7.2
Linear Regression	7.6
Decision tree alone	10.1
MLP alone	10.1

References

[1] Gang Liu et al. "A two-phase model for retweet number prediction". In: *International conference on web-age information management*. Springer. 2014, pp. 781–792.

[2] Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit*. 2002. DOI: 10.48550/ARXIV.CS/0205028. URL: https://arxiv.org/abs/cs/0205028.

[3] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: http://arxiv.org/abs/1908.10084.

[4] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

Appendix A: Analysis of Feature Importance

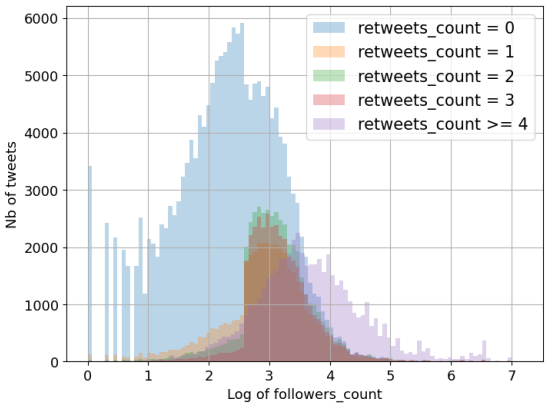


Figure 2: Histogram of the number of followers for fixed numbers of retweets

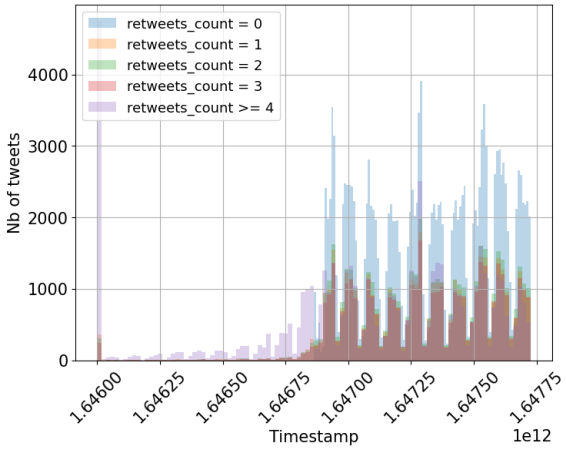


Figure 3: Histogram of the timestamps for fixed numbers of retweets

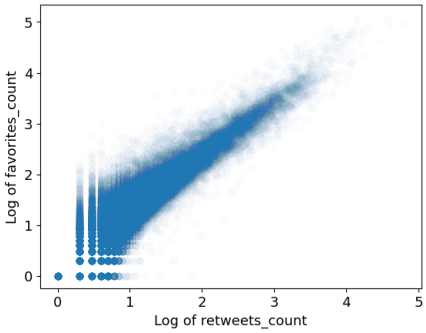


Figure 4: Retweets count vs favorites count, in log space

Number of features added	MAE
TD-IDF - 1 dimension	6.6
TD-IDF - 10 dimensions	6.6
TD-IDF - 100 dimensions	**6.5**
TD-IDF - 1000 dimensions	**6.5**
Pretrained embeddings	**6.5**

Table 1: Text features parameter search

Appendix B: Hyperparameter Searches

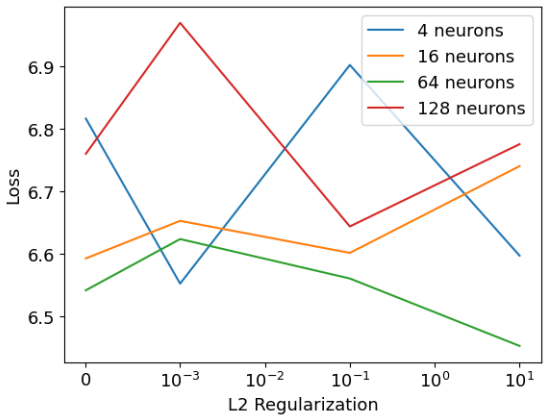


Figure 5: Loss depending on the regularization and number of hidden neurons

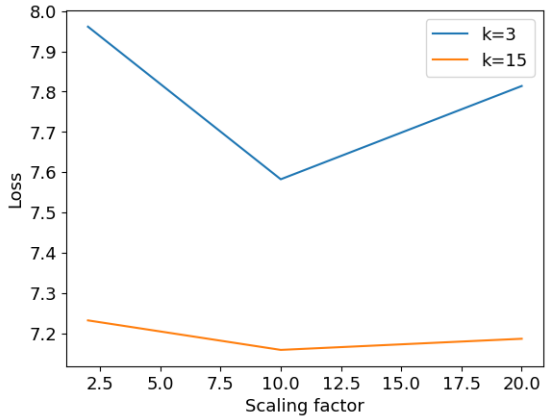


Figure 6: Loss depending on the scaling factor and k

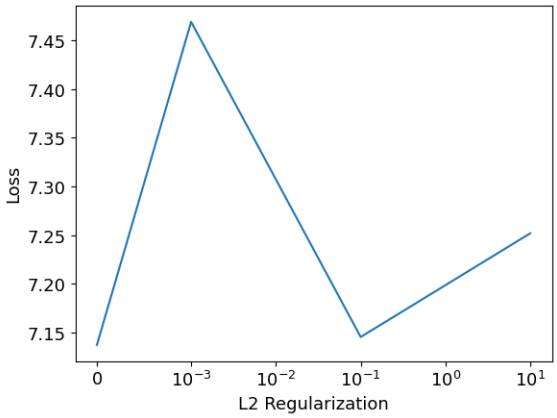


Figure 7: Loss depending on the regularization

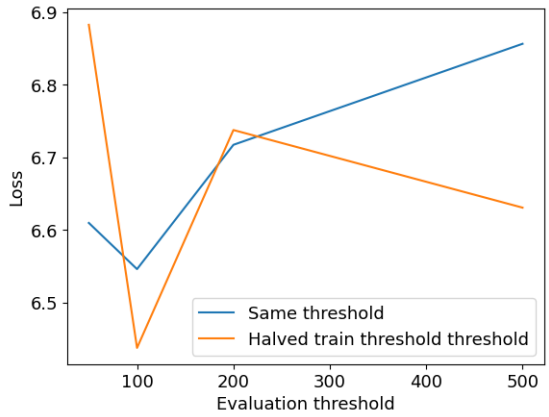


Figure 8: Loss depending on the threshold

Low threshold	High threshold	MAE
100	50000	7.9
100	10000	7.0
50	10000	**6.9**

Table 2: 3-phase parameter search