NLP for 💰💰💰

Lisa Goerke and Valentin Kotov
SOW-MKI61, Cognitive Computational Modeling of Language and Web Interaction
May 27, 2019,
George Kachergis.
Email: l.goerke@student.ru.nl and v.kotov@student.ru.nl

Abstract

Stock market prediction is a very interesting problem because it requires capturing the dynamics behind investor decisions which collectively shape prices. In this project, we tried to develop a Long Short-Term Memory Neural Network architecture which predicts the Dow Jones Industrial Average based on news headlines from two sources. Our goal was to shed light on whether an architecture with two recurrent layers is superior to an architecture with only one. After an extensive amount of experiments that exhausted a space spanned by various structural and hyperparameters, we find that the investigated architectures cannot solve the task at hand. We contemplate on this, discussing possible methodological (choice of pre-processing, bad data) and economic (efficient market hypothesis) explanations and provide pointers for further research.

1. Introduction

Predicting the stock market has fascinated generations of speculators and investors at least since stocks of the Dutch East India Company, the first company in history to issue shares of stock to the general public, were traded on the Amsterdam Exchange in 1602. This fascination does not stop with those interested in investing capital, but today proves to be an interesting problem for methodological explorations. What concerns the latter, one will find that of all recent work on stock market prediction, around 75% is concerned with machine learning, the same amount holds also for natural language processing (NLP)[1].

Therefore, it is interesting to look at those fields of machine learning and NLP which have (to our knowledge) not yet been applied to stock market prediction. In [13], a successful approach to neural machine translation was presented, which beat previous approaches to a large extent thanks to using a deep neural network encoder-decoder architecture based on Long Short-Term Memory (LSTM) units. The authors attribute this performance increase among others to the chosen architecture, which raises the question if also other natural language machine learning problems aside from translation can profit from an encoder stage followed by what has been previously used for the respective task. In the case of stock market the prediction, the question would be more concretely if here, too, an encoder-classification architecture is able to compete with state-of-the-art models. Looking into this architecture combined with a target problem other than translation can shed further light on whether the benefits of the architecture can also generalize to other natural language problems, and could motivate further research on creatively transferring architectures across domains. Moreover, solving the problem of stock market prediction with a new machine learning algorithm is of interest in and of itself.

As basis for stock market prediction in this study, two news data sets will be used. The first consists of the top 25 news headings of the Reddit world news subforum. The second is made up of the headings and content of economic news paper articles. With this data, the possibilities of a two-stage architecture for stock market prediction from news articles are be explored.

In Section 2, an overview over relevant background material is given, followed by a more detailed description of the project in Section 3 as well as its outcome in Section 4. Section 5 contains a contemplation on the project as a whole and Section 6 gives some suggestions for future work. Section 7 gives a short summary and conclusion to our work.

[1] *Recent* here refers to the last 10 years. The following information as well as that stock market prediction has been research before 1917 is deduced from the Google Scholar database. For the percentage information, the number of articles found with both topics concatenated was divided by the number of articles only on stock market prediction.

2. Background

LSTMs, originally introduced in 1997 [9], are specialized in modeling temporal sequences and are able to capture long-distance dependencies — which makes them very suitable for language related tasks since language is both, a temporal sequence and sensitive to long-distance dependencies. Previous works already applied LSTMs on a variety of natural language processing tasks such as Named Entity Recognition [11], semantic relatedness and sentiment classification [14], and text classification on multiple data sets with a varying number of classes [16]. In all of those works, the LSTM approach scored competitive or higher results than previous or traditional methods.

In [13], Sutskever et al. present a novel approach to machine translation, an instance of sequence to sequence learning with deep neural networks. While previous approaches only work on sequences of a fixed length, which makes them rather unflexible and domain-dependent, Sutskever et al. show that an two-stage encoder-decoder architecture with LSTM networks can overcome this shortcoming by first obtaining a fixed-dimensional vector representation of the input (stage one) and then applying a language model to extract the output using this vector representation (stage two).

3. Project

3.1. Research Question

Based on the fact that the encoder-decoder architecture described in [13] is essentially performing so well because it is a *two-layer* LSTM-network that learns the transformation function, we wanted to investigate whether a two-layer LSTM-model will have any comparable advantages over one-layer LSTM-models in the problem of stock market prediction (while the former is many-to-many and the latter many-to-one, it might still be an improvement).

3.2. Design

3.2.1. Approach. To have a principled and meaningful comparison between these model families, the comparison needs to be as fair as possible. The goal was to find and compare the best representatives of each family. The search space, spanned by model hyperparameters, was to be traversed by hyperopt [5], a Python package implementing the theoretically optimal Bayesian Inference Search using Parzen Tree Estimators.

The search space consisted of learning rate, batch size, and the dropout at three (resp. five) locations: after the embedding and the recurrent layer, and on the recurrent connections.

To isolate the effect that is due to the architectural difference, model capacity expressed in the number of learnable parameters (weights) was kept constant (with a two-layer network twice as deep, but half as wide).

3.2.2. Data sets. We found a suitable starting point in an openly available data set on kaggle.com [1], which contained 8 years of news headlines crawled from news aggregator reddit.com, as well as the historical performance of the Dow Jones Industrial Average (DJIA) during the same time frame. We also used a second news data set because we suspected the former to carry too little signal. We also considered predicting other stock indices than then DJIA, but there exists a whole marketplace around data sets of this kind, which makes freely available ones scarce.

First data set The *Daily News for Stock Market Prediction* data set [1] contains 1989 entries with the 25 highest ranked news headlines from 2008 to 2016. The data set also already has the DJIA adjacent closing price change connected to each entry as a binary variable (0 for falling in comparison to the day before, 1 for staying constant or rising). 45.8% of the entries belong to the falling class and 54.2% to the other.

Second data set The *Economic News Article Tone and Relevance* data set [7] contains 8000 news articles of 6109 unique days including heading and text as well as human ratings considering the tone and relevance to the US economy and meta data about the rating process. For this study, only the headings, texts and relevance ratings were used. Since the target value of the DJIA change is only accessibly from 1985 until today while the news articles are collected from 1951 to 2014, the actual data set used for training consisted of the overlapping data points resulting in a size of 3696 entries, which possible had more than one headline for each day. The DJIA adjacent closing price for the selected days was converted into a binary variable indicating a rising price in comparison to the next day (1), or a constant or decreasing price (0). On 51.4% of considered days a rise could be found, on 48.6% the price stagnated or fell.

For sanity checking, the relevance rating (0 for not relevant, 1 for relevant for the US economy) contained in the data set was also extracted in order to serve as a target variable. The data set had 19.8% positive relevance ratings as opposed to 80.2% negative relevance ratings, which is a rather imbalanced target. Here, if

Data set	size training set	size validation set	size test set
Reddit	1,589	199	199
(Reddit: kaggle script)	1,233	378	378
Crowdflower	2,956	370	370
Crowdflower Relevance	4,164	521	521

Table 1: Split of data sets into training, validation and test sets.

there were several headlines for one day, they could be separated according to their individual relevance rating, so that the resulting data set was bigger (4164 entries).

3.2.3. Pre-processing. Several pre-processing methods were tested.

Characterwise encoding with padding The data set was cleaned from special characters with simple string search and replacement and converted into an integer vector on character level. The vectors were padded to the length of the longest vector.

Vectorization The input sequences were vectorized according to the TF-IDF values of each word. The resulting vector had a length of 258.

Tokenization The data set was tokenized with the words being frequency-ranked words and converted into integers according to their rank. The resulting embedding had a length of 200.

3.2.4. Training - Test split. For the split into training, validation and test set, one has to bear in mind that time series data — as the stockmarket — has to be splitted keeping the original order intact to have a meaningful result from the validation and testing [10, 3]. We did so with both data sets using a 80-10-10 split. In the also used kaggle script, the data was divided 65-17.5-17.5. The resulting number of entries can be found in table 1.

3.3. Experiments

There was one python script for an LSTM model available on kaggle. [2] To ensure our starting point had a rigorous foundation, we only used it as inspiration and built our own script from the ground up. We quickly arrived at our own pre-processing, network, and pipeline. However, our model always converged to a trivial classifier (same prediction for all inputs).

We investigated various possible causes, primarily:

Insufficient model capacity – *reasoning*: because of hardware constraints, our model currently was an order of magnitude smaller than the model on kaggle[3]. *Test*: switch to EC2 on AWS to train models as large as those specified on kaggle. *Result*: no change, continuing with small network, but staying on AWS.

Faulty pre-processing – *reasoning*: largest point of divergence from script on kaggle. *Solution*: switch to pre-processing of kaggle script *Result*: model finally learns a more complex transformation.

Also, we pivoted to using the second data set (headlines only), in the hope of it carrying more information by virtue of having more data points. From comparing the final and peak performances of 3000 runs of the kaggle script[4], supplemented with dropout of .2, .3 or .5 at all three candidate locations, dropout=.5 clearly yields the largest improvement. We adopted this change to our own script. To keep computational costs limited, we only ran hyperopt to tune the learning rate for now. We furthermore adapted the kaggle-script to use two separate test sets (see table 1), as in the original implementation, validation set was re-used as the final test set, yielding a strongly biased "test" performance that is a poor indicator of true generalization power.

The next section describes our final architecture in some detail.

3.3.1. Neural network. All architectures used were implemented in Python [15] with the help of the keras deep learning library [6]. We chose to use the Theano [2] backend because of a speed-up in training time of factor three.

A visualization of the model architecture we investigated can be found in figure 1.

The experiments were run on an machine with 4 CPUs and 7.5 GB of RAM in the Amazon Elastic Compute Cloud (EC2) [4], as GPU machines were unavailable to our account within the time frame of the project.

4. Results

We used our network on the second data set, and the kaggle script on the first data set.

[2] kaggle.com user lseiyjg, "Use News to predict Stock Markets", https://www.kaggle.com/lseiyjg/use-news-to-predict-stock-markets

[3] Kaggle: 128 units in both embedding and LSTM layer. Ours: 20 and 10, later 20 and 20.

[4] (down-sized to our size, i.e. 20 embedding neurons and 10 LSTM cells, in contrast to 128 and 128)

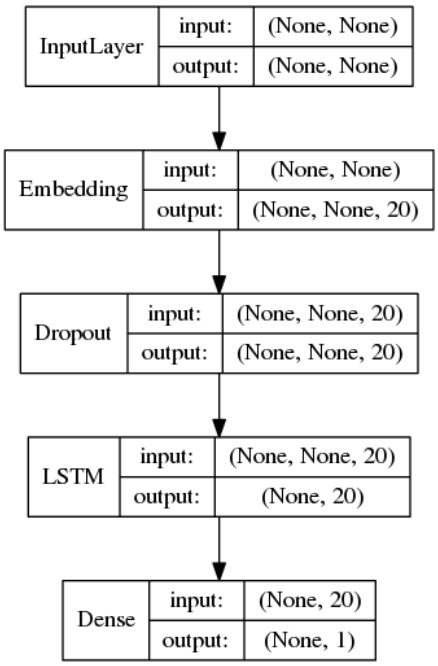

Figure 1: The one-layer basis of our final model architecture (the two-layer version has 10 units per LSTM layer). Note that there is also dropout within and after the LSTM layer(s) (this not being shown is an artifact of the `keras.layers` and `keras.utils.vis_utils` API).

At the end, our network was able to achieve a training accuracy of 99.5 when not using a validation set and 98 otherwise, while the validation accuracy never increased beyond chance level. To decrease this overfit, we looked into regularization through dropout (there are other methods, but dropout should always have an effect). Interestingly, we were not able to increase validation set performance beyond chance level at all, regardless of the level of dropout and the learning rate (learning rates need to be increased 10-100 fold when using dropout [12]).

The same picture appeared on the kaggle data: After adding a separate validation set to the kaggle script, it as well turned out to not be learning anything that provided better than chance performance on the separate test set.

So, to summarize, this bad performance occurs:

1. irrespective of brute force model capacity
2. irrespective of learning rate (0.0001-100) and dropout level (0-0.5)
3. regardless of which of the two data sets was used
4. regardless of the architecture (one level vs. two levels)[5]

On a more positive note, during our work we were able to achieve a markedly increased validation accuracy of 60.32 (compared to 56.6 reported in the original notebook) with the kaggle script, while using a one order of magnitude less complex model (only 10 LSTM and 20 embedding units instead of twice 128).

5. Discussion

The results of our experiments are both startling and fascinating.

We see a number of possible explanations for this failure to find a model picking up a signal present in both the training and the validation/test sets:

Wrong preprocessing: We still did not get any other preprocessing than the one we eventually shared with the kaggle script to yield any learning. As we tested the generalizability of the kaggle model very late, this would have been one of our next steps, as it might just be that this preprocessing is the reason why neither the kaggle-script nor our network generalize to unseen data.

Inherent difficulty: For methodological reasons, the validation and test partitions of the full data set comprise the most recent parts of the date range. In our domain, this standard approach from modeling historic time series is faced with an issue that is characteristic for this type of problem: Markets, especially stock markets, tend to be efficient.[6] Market participants have a very high incentive to find, exploit and thereby eliminate inefficiencies in the market (like those that we would hope to find using our model). It furthermore is likely that conclusions of such simple models as ours, trained on such obvious data sources as news headlines, are – at least today – *already reflected in market valuations on those younger date ranges*. Therefore, using these more recent ranges for testing out-of-sample performance effectively becomes a race against the whole market, as the dynamics which shaped prices in the past have always kept up with the most recent modeling approaches. Thus effectively, our test-set performance

[5]For the sake of completeness, we trained 200 instances of both the one-layer and the two-layer version of the above network (and 200 on variations of the two). In none of these 2 (4) variants, the validation (and test) set performance rose above chance level.

[6]This is sometimes called the 'efficient market hypothesis', an idea going back to Friedrich Hayek's conception of markets as the most effective tools for the aggregation of information dispersed among society's members [8].

is not actually out-of-sample performance, but out-of-problem performance. In consequence of this very dynamic nature of the problem, we may conclude that a model as simple[7] and as closely related to [13] as ours probably has no chance in winning against the post-2014 market (given 2014 as the year in which [13] was published and given the fact that quants are always at the bleeding edge of machine learning).

We could test this hypothesis by completely moving to much older date ranges, but we did not have such data readily available. Even then, good performance that only generalizes to a separate test set if that test set is also in the more distant past (but not to today) would have little to no practical utility.

While to us, these are good reasons to believe that the problem instance (predicting this index based on this particular data) is too intrinsically hard for simple approaches like ours to yield any results, for this to be the root cause (and a sufficient explanation) it would be required that *all* higher-than-chance test set performances reported in the forum accompanying this data set on kaggle.com would have to be due to errors in the submissions.[8] Based on our own cursory analysis of some of the other solutions, this seems to be very well possible (as most are of very poor quality, using unsound methods or assumptions).

6. Future Work

Future work could, if only for academic interest, target older date ranges. If predictions for today's world are sought, we recommend to explore other data sets for input, and to minimize the likelihood of the market appropriately reflecting the signal we hope to extract from this data by aiming for targets that have a lower likelihood of being targeted by market participants (maybe individual stocks, or indices of other, foreign markets, possibly in emerging or frontier economies as these have the reputation of being much less efficient than developed markets).

7. Conclusion

In this project, we tried to predict movements of the Dow Jones Industrial Average based on news headlines from two sources (reddit and CrowdFlower) using LSTM networks. We performed an extensive amount of experiments, exhausting a large hyperparameter search space, but in the end couldn't reach performance significantly better than random guessing on our test set containing the most recent available data points (mostly from the year 2016). We suspect that there are either faults in our method (in the implementation, but more likely in the approach), or, more likely, that market valuations at least in the date range we used for testing out-of-sample accuracy already account for the information added by models as simple as ours, negating the predictive effect that these would otherwise have had.

As noted in the introduction, we began this project with the assumption that this type of modelling has not yet been applied to stock market prediction. Looking at the above economic argument being the most likely explanation for our results, we have to admit that we were unaware of just how critical this assumption might have been.

[7] Typically, predictive models employed by Wall Street firms and hedge funds consider not one, but hundreds of public and non-public (and often obscure) data sources.

[8] Information leakage from validation/testing into training (leading them to report inflated performances). The best performing submissions have accuracies in the sixties, which is also the range reached by the validation accuracy of the original kaggle LSTM script, which incorrectly reported this accuracy as *test* accuracy in the public notebook).

References

[1] Aaron7sun. Daily news for stock market prediction. `https://www.kaggle.com/aaron7sun/stocknews`, 2016.

[2] Al-Rfou et al. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[3] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[4] Amazon Elastic Compute Cloud, Amazon Web Services `https://aws.amazon.com/ec2/instance-types/`

[5] Bergstra, J., Yamins, D., Cox, D. D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *Proc. of the 30th International Conference on Machine Learning (ICML 2013)*, 2013.

[6] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[7] CrowdFlower. Data for everyone. `https://www.crowdflower.com/data-for-everyone/`, 2017.

[8] Friedrich August von Hayek. The Use of Knowledge in Society. *American Economic Review*, 35(4):519–30, 1945

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Rob J. Hyndman. Why every statistician should know about cross-validation. `https://robjhyndman.com/hyndsight/crossvalidation/`, 2010.

[11] Nut Limsopatham and Nigel Collier. Bidirectional lstm for named entity recognition in twitter messages. *WNUT*

2016, page 145, 2016.

[12] Srivastava, Nitish, et al. Dropout: A simple way to prevent neural networks from overfitting. In *The Journal of Machine Learning Research*, volume 15.1, pages 1929–1958, 2014.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[14] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[15] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[16] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

Appendix

Our code can be found at our github repository. We worked to large extents with pair programming and pair paper formulation, exploiting the beauty of being able to look at the running network on one laptop while noting down its performance on the other. With two people and 5 pages, there are not many sentences only one person has had their hands on, we therefore think the author of each section are we both. A worklog can be found below (it shows the amount of hours per person, total working hours are twice these numbers).

Data cleaning (Reddit data set)	1:15
First network attempts	4:20
Data cleaning (Economic news data set)	2:10
Setting up three EC2 instances	4:35
Trying various things to find source of problem	25:20
Write-up	16:00
Sum	53:40