Extracting Numeral Relations in Financial Tweets

Jon Ryfetten
Computer Science / 4th year
jonry@stud.ntnu.no

Abstract

This report provides a model for detecting numeral attachments given a target entity. Understanding the relationship between entities and numerals in a natural language plays a crucial role in many domains. It is especially important in health records, financial data, and geographic documents. The model presented here is a part of the FinNum2 challenge, which is a task for fine-grained numeral understanding in financial social media data. We take a look at how one-way word-by-word attention and Recurrent Neural Networks can be used for encoding and classifying the relationships between stock symbols and numerals.

1 Introduction

Extracting relations from unstructured sources is important to understand and analyze the data. Most of the data in organizations is unstructured data (Shilakes (1998)) and require sophisticated methods in order to extract information. Today, we are facing increasing importance of understanding of such information in areas of such is clinical records, voice assistance, and in the finance industry. For many of these areas, it is required to understand the information in depth. Only understanding the meanings of numerals is not enough for practical uses, we need a fine-grained numeral understanding, in order to analyze the data.

FinNum (Chen et al. (2018)) is a shared-task for fine-grained numeral understanding in financial social media data. Because of the informal writing style in social media, the data is often considered more challenging than analyzing news and official documents. In 2019, FinNum-2 (Chen et al. (2019)) was released and describe the task of identifying linking between a stock symbol and a numeral in a financial tweet. Given a target numeral and a stock symbol, the problem is a binary classification problem to tell whether or not the given numeral is related to the given stock symbol. An example given here:

```
$RIOT $MARA $SRAX all are going to come/find $DPW .. lowest market cap
low float ready to rip above 1.
```

In the example, the target numeral is *1* and the target stock symbol is *$DPW*. As we can understand, the example has to be labeled *true*. Below is an example of a tweet and a relation labeled *false*. The relation is between *$PLUG* and *15*.

```
$BLDP $FCEL $PLUG - Some 15 million fuel cell vehicles could be on
the road by 2030 http://www.hydrogenfuelnews.com/some-15-million-fuel-
cell-vehicles-could-be-on-the-road-by-2030/8533490/
```

The contributions this report presents is proposing a one-way word-by-word attention model for entity extraction to the FinNum2 challenge. The model presented here achieved a 5.12% better F1-macro result than the Capsule-based model presented by the team behind FinNum2. It should be noted different test dataset was used, as the official is yet not released. The test dataset used in this report was built using 10% of the data from the currently released FinNum2 dataset.

2 Background

2.1 Gated Recurrent Unit

Recurrent Neural Networks (RNN) allows us to map time into state. They are designed to take a series of input, without a fixed limit, and output a sequence. Gated Recurrent Unit, GRU, (Cho et al. (2014)) is a RNN which includes an improved ability for long term memory by including gates to regulate the flow of information. The idea is that the gates learn which data in a sequence that is important. GRU is a simpler than the more popular RNN, Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber (1997)) but has proven similar performance. The GRU contains a single "update gate" (Eq. 2). The model decide how much to forget in the reset gate (Eq. 3). The current memory content is calculated in Eq. 4, and the final memory for the timestep is calculated in Eq. 5

$$\mathbf{H} = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} \tag{1}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{H}) \tag{2}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{H}) \tag{3}$$

$$\tilde{\mathbf{h}}_t = tanh(\mathbf{W} \cdot [\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t]) \tag{4}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{5}$$

Where $\mathbf{W} \in \mathbb{R}^{2k \times k}$ is weight matrix. σ denotes the element-wise application of the sigmoid function. The $\odot$-sign denotes element-wise multiplication of two vectors.

2.2 Attention

Attention (Bahdanau et al. (2014)) is inspired by how human visual attention allows us to have a focus on different regions when looking. In deep learning, attention could be used on both images and text. In the context of a GRU, attention allows the GRU to focus on different parts of the previously encoded outputs and improve its ability for long term memory. Word-by-word attention is about applying attention to word tokens. In this report, we will use attention not to generate words, but to obtain an encoding. Attention will be discussed future under the architecture, (section 4).

2.3 Word embedding

Neural networks do not understand words or characters. To represent tokens (words or characters), we build a vocabulary. The vocabulary contains all tokens that match a frequency threshold requirement. For every entity in the vocabulary, we one-hot encode the integer value of the entity. We then train a neural layer (embedding layer) to capture word relations in a more compressed vector space. The layer can be trained with the model, but can also be pre-trained used transfer learning.

2.4 F1-score

F1-score is known as the balanced F-score, which is used to measure test accuracy. It is the harmonic mean of precision and recall. It can be interpreted as a weighted average of the precision and recall. The formulas is listed below.

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{6}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{7}$$

$$F_{score} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8}$$

When computing the average of the F1-score, there are multiple ways of achieving it. In this report, we are going to look at both Macro-average and Micro-average. Micro-average aggregates the contributions of all classes and compute the average. Macro-average computes the metrics independently for each class and then take the average between the classes while treating the classes equally.

3 Related Work

The team behind the FinNum2 challenge has released the paper "Numeral Attachment with Auxiliary Tasks". In the paper, they describe one technique to solve FinNum2. They also include an auxiliary task not mention in FinNum2 about finding the reason type for the mentioned numeral. They present an embedding technique to consider words and numeral information simultaneously and also a design for a joint learning model with a capsule network to accomplish the task. The joint learning model contains a Convolutional Neural Network to extract features for the language model. The feature matrix is then processed in a capsule network and they use a bidirectional-GRU to capture the features from the capsule network. The model can complete both of the tasks and include a clever position-based approach for the embedding of the tokens. The model has a macro F1-score of 73.46% on their test dataset. The approach the paper uses to solve the problem describes an interesting path to solving FinNum2 using modern state-of-art methods. It is quite different from the approach proposed in this report and might yield better performance for generic numeral relation extraction, due to the clever embedding process. A small similarity is the choice of GRU over LSTM. They do not particularly describe the reason why they choose GRU, but we know that GRU exposes the full hidden content and has performance on par with LSTM while being more computationally efficient.

Another related paper was released in 2015 by Google DeepMind in collaboration with the University Of Oxford and University College London. The paper is called "Reasoning about Entailment with Neural Attention" (Rocktäschel et al. (2015)). Motivated behind the paper is about achieving Recognizing Textual Entailment (RTE) systems without heavily rely on engineered natural language processing (NLP) pipelines, extensive manual creation of features, as well as various external resources and specialized sub-components such as negation detection. In the paper, they presented a state-of-art generic end-to-end system for textual entailment. The model takes two texts as input, a premise, and a hypothesis. They then perform conditional encoding using two LSTMs and a Multi-Layer Perceptron (MLP) as a classifier. In the paper, they implement a two-way word-by-word attention mechanism between the encoders. The paper was found after developing the later presented model in this report, but the two models share many similarities. It should be clarified that the model here is not based on the paper. The model later presented here contains one-way word-by-word attention, with two encoders and a separate classifier module. The major difference between the models is the classifier, different RNN, and one-way attention. In the paper from Google DeepMind, under the analysis section, they discuss that the two-way attention did not seem to improve the performance. They suspect that is due to entailment being an asymmetric relation, and that it might then lead to noise in the training signal. The paper proposing a generic RTE system, while the model proposed here is specialized for binary entity extraction, a subbranch within RTE systems. For the model presented in the next section, the relation will also be asymmetric, and two-way attention would be unnecessary here as well.

4 Architecture

At a high-level, the model contains two encoders and one classifier. The first encoder, encodes the possible relation, while the second encoder, encodes the tweet. When encoding the tweet, the model has a one-way attention mechanism between the encoders, to give the model focus on the relation while encoding the tweet. We then grab the encoded hidden state and last output and feed it into the classifier. The classifier decodes the information using a neural layer. The high-level architecture of the model is displayed in Figure 1.

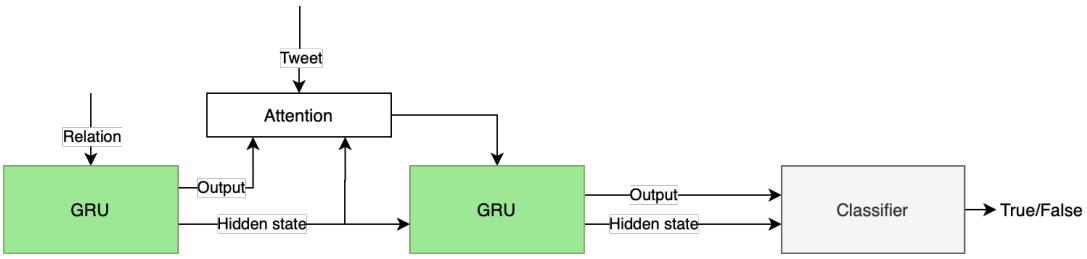

Figure 1: Architecture of the model

4.1 Word embedding

The model process tokens at a word level, instead of for example characters-by-character. The vocabulary threshold is set to 10, which means word with frequent less than 10 is converted to a common token ("unk"). The embedding is trained with the model.

4.2 Pre-processing

To able to extract information from the tweets, all numbers are replaced with a num<index> token. This limits the model to only understand sentences with a fixed amount of numbers, as the embedding only learn the fixed numbers. The same is done with the tickers.

An alternative approach could be to always assign a given token to the target number and ticker, and assign the rest of the number and ticker a given token respectively, eg "num" and "ticker". This means that we are dropping the information on the index of the numbers. Experiments show that including the index in the number has yield better results than without.

4.3 Attention mechanism

Attention mechanisms is often found in Seq2Seq models (Sutskever et al. (2014)). Normally, you would have an attention decoder with teacher enforcing. In the presented model here, the attention mechanism is connected to a second encoder, the tweet encoder. We therefore does not include teacher enforcing as the encoder does not output a sentence, but an internal encoding. The reason to still include the attention mechanism is to make the network able to focus on the relation. The attention weights are calculated with a feed-forward layer, attn, as seen in Figure 2. The figure describe one iteration of a word token. The initial hidden state is from the relation encoder, before the tweet encoder produces the next. After attention weights are calculated, a softmax function is applied and we get the final attention-weights. We then perform a batch matrix-matrix product between the attention-weights and the output from the relation encoder. After the attention has been applied, the matrix is combined with the embedded input token and a ReLU function is applied before the matrix is sent to the GRU.

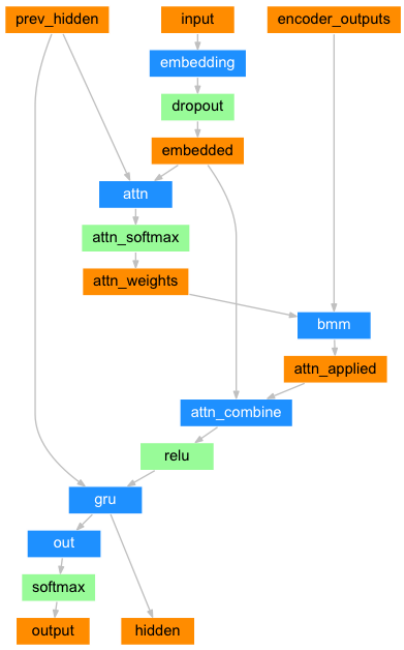

Figure 2: Attention architecture. The attention weights and encoder outputs is combined with a batch matrix-matrix product function (bmm). Image is adapted from Robertson (2017)

4.4 Classifier

The classifier is a neural network layer, which takes as input the last output vector and the hidden feature vector from the tweet encoder. The two vectors are combined to a matrix and forward-passed to the classifier. The model then outputs a binary classification.

5 Experiments and Results

5.1 Experimental Setup

For the final results, the embedding dimension was set to 256. The hidden size of the two GRUs was set to 512. Adam optimizer was used, with a learning rate set to 0.00005. Since the dataset is highly unbalanced, a weighted loss was applied. The weights were set to 0.65 for false labels and 0.35 for true labels.

When building the vocabulary from the dataset, the threshold for keeping a word was set to 10. The dataset was split (before developing the model) in training, validation, and test dataset, each respectively at a size of 80%, 10%, and 10%. The original dataset contained 7187 entities, which is a relatively small dataset.

5.2 Experimental Results

After the hyperparameters were found, the model was trained and quickly started to overfit after around 4 epochs, where the final model was saved. The performance results on the test dataset can be displayed in Table 1 below. Since the model is a binary classifier, the accuracy of the model is the same as the F_1-micro score.

Table 1: Results

F_1-micro	F_1-macro
89.15%	78.58%

The loss is displayed in the graphs below, see Figure 3 for the final model. For reference, a run with 10 epochs is displayed in 4.

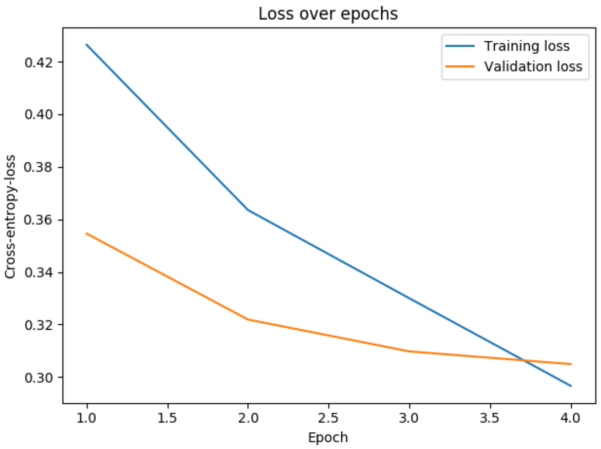

Figure 3: Cross-entropy loss over 4 epochs, learning rate set to 0.0005. Final model used.

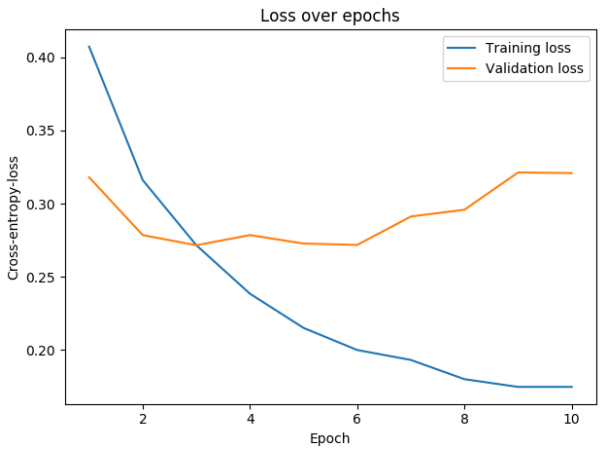

Figure 4: Another model, cross-entropy loss over 10 epochs. The model overfits early after 4 epochs.

6 Evaluation and Discussion

The model achieved a 5.12% improved F_1-macro score over the proposed capsule-based model from the team behind FinNum2. At our first glance might look promising. As described in section 1, the official test dataset is not yet released and the results here are based on a 10% split of the released dataset.

When evaluating the performance of this model, we need to evaluate the performance in the context of the dataset. The dataset used here is the training dataset provided by FinNum2. It is a highly unbalanced dataset, which makes looking at the F_1-micro score less important. In our case the same as the accuracy. Therefore, looking at the accuracy does display the full picture, and increase the importance of the F_1-macro score. In our validation dataset, 81.64% of the data was found labeled as a true relation. To combat this problem, a weighted loss function was used while training the model, details are mentioned in the experimental setup 5.1 section. Using weighted loss helped to improve the F_1-score.

Looking deeper at the dataset. An issue with the dataset is not only that is highly unbalanced, but also the examples labeled false contains different characteristics than the data labeled true. Most false labeled data had sentences with a range of different stock symbols and numbers. Many of the sentences with a true label only had one ticker and often 1-2 numbers. This has been an issue for the model proposed here since the model here depends on the learning of a distribution. When distribution does not reflect the real world, the model will neither perform well when using real data. Of course, this also points out a weakness of the model, as the model might be too sensitive for factors such as this and struggle to learn the correct features.

An example of data the model will struggle to understand is displayed below.

```
$AAPL $MSFT $TSLA all up over 1% today, $FB down 2%.
```

Given a set of different relations, the model will label this sentence false in most cases, as most of the sentences with these characteristics are labeled false in the dataset.

In general, it is more difficult to process tweets than formal data, as the data contains a lot of different styles of expression. Using a small embedding size for the vocabulary made the model perform better, that might merged the related words (made the distance between the words in the vector space smaller).

When we look at the loss graphs, in Figure 3 and 4, we see that the model starts to overfit very early. Normally, this should not be the case and is a sign that something is wrong. There could be several reasons. I believe the most likely cause is that the model itself is not optimally designed given the FinNum2 dataset, or the quality of the dataset is not good enough. Most likely, it is a combination of both. In general, the more data we have, the better the model performs, and the result could have been better. In our case, the dataset was quite small, and a larger dataset would have been beneficial.

The embedding on numbers and tickers also puts constraints on the model, by limiting the number of tickers and numbers in a tweet to a fixed size. While this works fine for FinNum2, a different approach to solve this should be considered for generic tweets.

The model itself might scores relatively high on the FinNum2 challenge, depending on the yet to be released test dataset. We do not have enough information to know if the model perform achieves the goal of generic fine-grained numeral extraction in financial tweets, due to the weaknesses pointed out with the FinNum2 dataset.

7 Conclusion and Future Work

The report presents a deep learning model to do extraction of numeral relations in connection to financial tweets. A key mechanism in the model was the use of attention to encode the data. The overall performance of the presented model is good on the FinNum2 dataset. An important question to raise is if the dataset provided by FinNum2 is a good reflection of relations in real-world financial tweets. Future work would be to improve the dataset used, improve the pre-processing, and change the embedding of words. Encoding of the relation is an important part of the model, and this report just presents one way of doing it.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Chung-Chi Chen, Hen-Hsen Huang, Yow-Ting Shiue, and Hsin-Hsi Chen. Numeral understanding in financial tweets for fine-grained crowd-based forecasting. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 136–143. IEEE, 2018.

Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. Numeral attachment with auxiliary tasks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1161–1164, 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

Sean Robertson. Nlp from scratch: Translation with a sequence to sequence network and attention. `https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`, 2017. Accessed: 30-05-2020.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

Julie Shilakes, Christopher C.; Tylman. Enterprise information portals. archived from the original (pdf) on 24 july 2011. `https://web.archive.org/web/20110724175845/http://ikt.hia.no/perep/eip_ind.pdf`, 1998. Accessed: 30-05-2020.

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.