Recording

https://www.youtube.com/watch?v=hUn7ZKXFfiU

MidasSense Stock Market Prediction with Advanced NN Architectures CNN, LSTM, BiDirectional LSTM and BERT

Siqi Zuo (zuosiqi@stanford.edu), Yiyang Shen (yiyangs@stanford.edu), Bo Yang (hiyangbo@stanford.edu)
Stanford University CS221 Project Poster

MidasSense Introduction

In this project we seek to do stock market prediction based on news, we believe this model will be helpful for not only individuals but also policy makers to make better decisions from world and domestic news

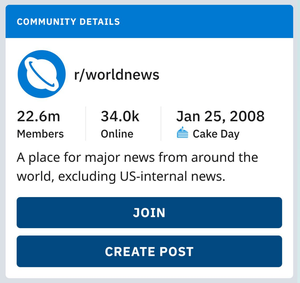
Input is a list of news from the present day

- Picked based on upvotes
- 25 selected per day

Output will be a binary indicator of stock market the following day values of Dow Jones Industrial Average (DJIA) to generate the following labels

- Closing values comparison
- Open values comparison
- Close values vs open values

The task of the model to is predict this binary indicator based on input text.



Data and Tools

- Data scraping tools
  - Scrapy, reddit api PRAW, bigquery
- Datasets used
  - bert_en_uncased_L-12_H-768_A-12 model by Google research is used for preinitialize our BERT model to be fine tuned for our classification task
  - Cnn daily mail non-anonymized summarization dataset is used for text encoding
- Model training and development libraries used
  - tensorflow keras, keras-bert
  - pytorch huggingface transformers

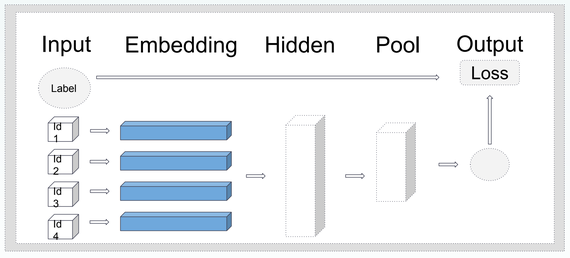
Input Processing

In order to feed our string input to our model we need to represent them as numerical inputs. We achieve this via three steps

- Text tokenization
  - every piece of string is tokenized to a list of unigrams
- Text encoding & padding
  - each unigram is assigned an id, in this project we use “cnn_dailymail/subwords32k”, a CNN/DailyMail non-anonymized summarization dataset for this encoding
  - force each list of unigram to be equal length by padding missing positions
- Embedding lookup
  - each id corresponding to a fixed length numeric vector, this id-to-vector mapping is trainable in our case
  - a sentence now becomes a list of fixed dimension vectors

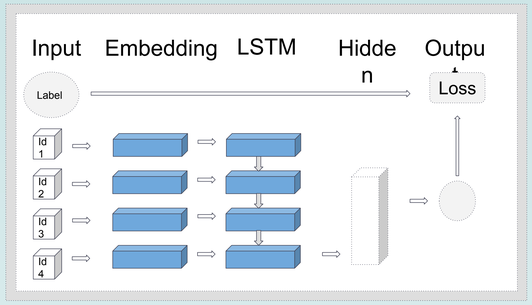
MidasSense v1 -- CNN

Convolutional neural networks are some of the most classical neural networks. They are known to be used for many classification tasks. So our first deep architecture explores CNN.



In output layer we represent each label in a one-hot manner. For example a label of 1 is [0,1] and a label of 0 is [1, 0]. We then apply binary cross entropy loss between output layer and input label.

MidasSense v2 -- LSTM



Since news text by itself has some inherent recurrent structure, it would be useful for us to exploit such substructures using recurrent neural network structures. We use the same embedding lookup as our CNN version as the input to LSTM layer.

MidasSense v3 -- Bi-Directional LSTM

In regular LSTM, cells only have memory of the past. This can be fundamentally limiting as we also care about the future. Hence combining forward and backward structure will be better at capturing context than regular LSTM.

For example consider the sentence “Mark Zuckerberg to give away 99 percent of his networth to charity”. A regular forward LSTM cell at “away” only has information about “Mark Zuckerberg to give away” and a backward LSTM cell at that position only has information “percent of his networth to charity”.

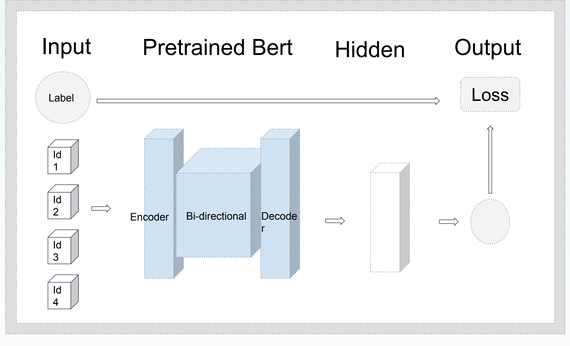
MidasSense v4 -- Fine-Tuned BERT

BERT stands for Bidirectional Encoder Representation for Transformers. BERT in essence consists of an encoder-decoder component, as well as a bidirectional recurrent component.

- The encode decode structure enables unsupervised learning from context
- The bidirectional structure enables learning from context from both left and right

Thus defined BERT learning is usually a 2 step process

- starts with an unsupervised task to learn randomly masked tokens from the remaining tokens
- use the learned model and fine tune it for specific task such as classification task in out model



We obtained our trained model from publicly available data set trained on wikipedia corpus with uncased tokens on English language. We initialize our model with trained parameters and added another layer on top for our classification task.

One caveat is that we no longer use cnn news vocabulary in this design, instead we use our news vocabulary intersect pretrained model vocabulary.

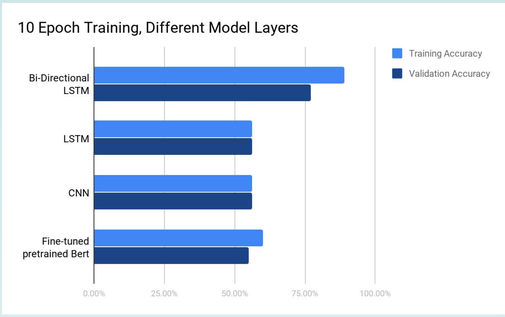
Results

We have compared the performance we have recorded the performance of each individual model below in this bar chart. As we can see bidirectional LSTM is the winner, despite being less complicated than fine-tuned pretrained BERT model.

We use ⅔ of our data as training data, and the remaining as validation set. In order to fairly present the results from a comparison amongst multiple different models. We limit each model to the same configuration of

- 10 epochs
- batch size of 64
- hidden layer size of 64

We also applied the same label across the comparison

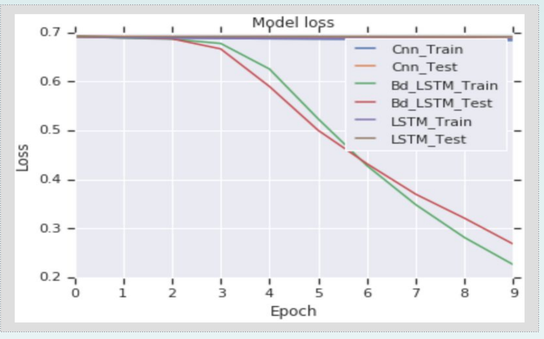


Concluding Analysis

Observation 0
BiDirectional LSTM is the winner

- training set convergence
- validation set convergence
- training set loss
- validation set loss

Also training time and resources required for this model is also moderate compared to that of pretrained BERT.



Observation 1 BERT does not work well

Architecture wise, pretrained BERT model makes most sense. We have come up with a couple hypotheses on why our BERT implementation fails for our project

- When we experimented on a very small subset of data, after 10 epochs we observe convergence of loss. However as we increase the training sample size, we no longer observer convergence. This could mean that we might have not trained our model enough with the epoch constraint we imposed.
- Our corpus is trained on wikipedia corpus. However the distribution of semantic and syntactic relations is much more different than those in reddit news. Therefore the learned model may not transfer well. Therefore initialization from this pretrained model is not much closer to actual optimum.

Observation 2 Bidirectional LSTM has a sizable gap between training error and test error

Bidirectional structure as mentioned can capture more relations between sub-sentence tokens. However this can also mean that they would tend to overfit more given the more complicated structure.

Observation 3 BERT requires more training time and resources

Regarding **training time,** a side remark is that BERT model is the most complicated to train. On the same data set our pretrained BERT fine tuned model takes 4 times as much time to train as the bidirectional LSTM model, even when GPU training is enabled. BERT model also consumes a large amount of memory with relatively moderate batch size. While simple convolutional neural network can be trained at the least amount of time, it does not perform well. So in practice for fast development cycle, moderately complicated bidirectional LSTMs is our best choice for this project.