Problem statement:

The problem is to use financial information obtained from quarterly earning statements and identify and use the text information in the reports to identify trends in the prices of the stock.

As part of this we take two different approaches -

1. Explored how various neural network architectures can solve a problem similar to the above problem statement. This simplified problem is taken from a kaggle competition - Daily News for Stock Market Prediction
2. The idea for the model is based on Lee, Surdeanu, MacCartney, Jurafsky paper “On the Importance of Text Analysis for Stock Price Prediction”. There are three labels associated with the prediction viz., UP, DOWN and NEITHER.

Kaggle competition problem statement:

Given top 25 headlines for each day, the task is to predict -
- "1" when DJIA Adj Close value rose or stayed as the same;
- "0" when DJIA Adj Close value decreased.

Data:
- Data for 1611 days between 2008-08-08 and 2014-12-31 is taken as training set
- The following two years data for 378 days between 2015-01-02 and 2016-07-01 is taken as test set
- The test set has roughly equal distribution of both the labels “0” and “1”.

Models:

BoW model - sklearn Logistic Regression:

- This model is directly taken from kaggle discussion board.
- All 25 headlines for each day are concatenated and treated as one sentence. Used sklearn’s term frequency countvectorizer for implementing BoW model. Default logistic regression provided by sklearn is trained and tested on this BoW model.
- Able to replicate results that were mentioned in kaggle discussion board.
- Obtained below par test accuracy of 42% with unigram model, confusion matrix =

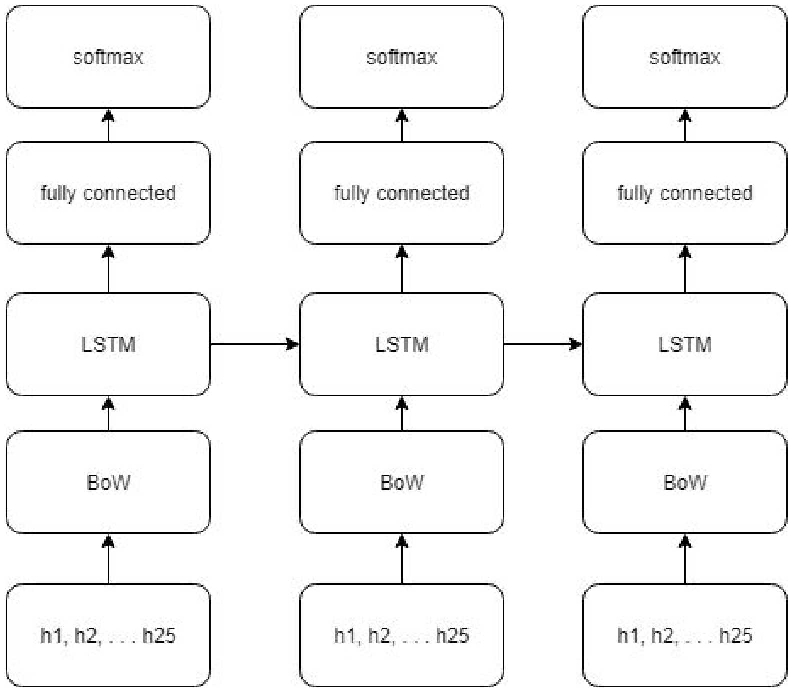
	0	1

0	61	125
1	92	100

- Obtained a better test accuracy of 57% with bigram model, confusion matrix =

	0	1
0	66	120
1	45	147

BoW model - LSTM:

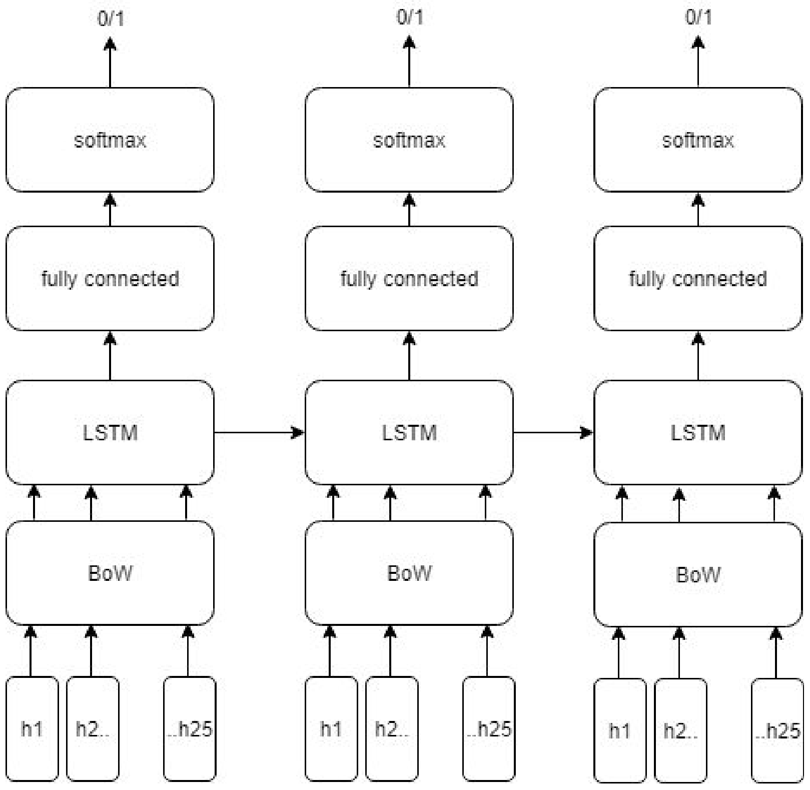


- The underlying modelling of the headlines is same as above but an LSTM layer is added on top of the vector encoding of the headlines and then a hidden layer followed by a softmax layer for label prediction.
- The idea is to capture the temporal information across the days.
- Tried varying hyperparameters - hidden cell size of LSTM, min_df parameter of CountVectorizer, number of epochs etc.
- Achieved a training accuracy of ~ 99.8% and a trivial test accuracy of 45.2%, confusion matrix =

	0	1
0	72	114
1	93	99

- Clearly this model is overfitting.

Variation:

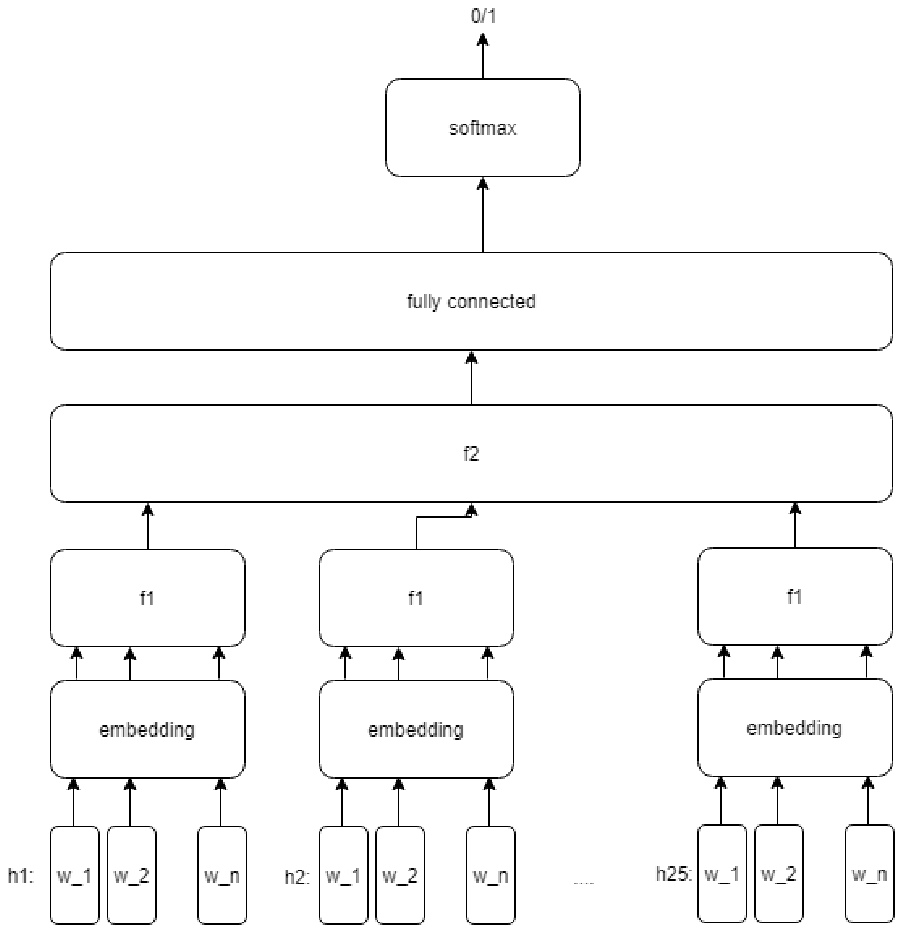


- Changed the modelling of the headlines. Instead of concatenating all the 25 headlines as one one BoW vector, created 25 BoW vectors for each of the headlines.
- All of these are passed to LSTM layer as one batch of input ie., batch size = 25.
- After hyperparameter tuning, obtained a similar training accuracy as before but a better non-trivial test accuracy of 52.3%, confusion matrix =

	0	1
0	55	131
1	49	143

- From these experiments, we concluded that the models are too complex for the data set and that they are simply memorizing the training data.

Pre-trained word embeddings (Glove) + logistic regression:

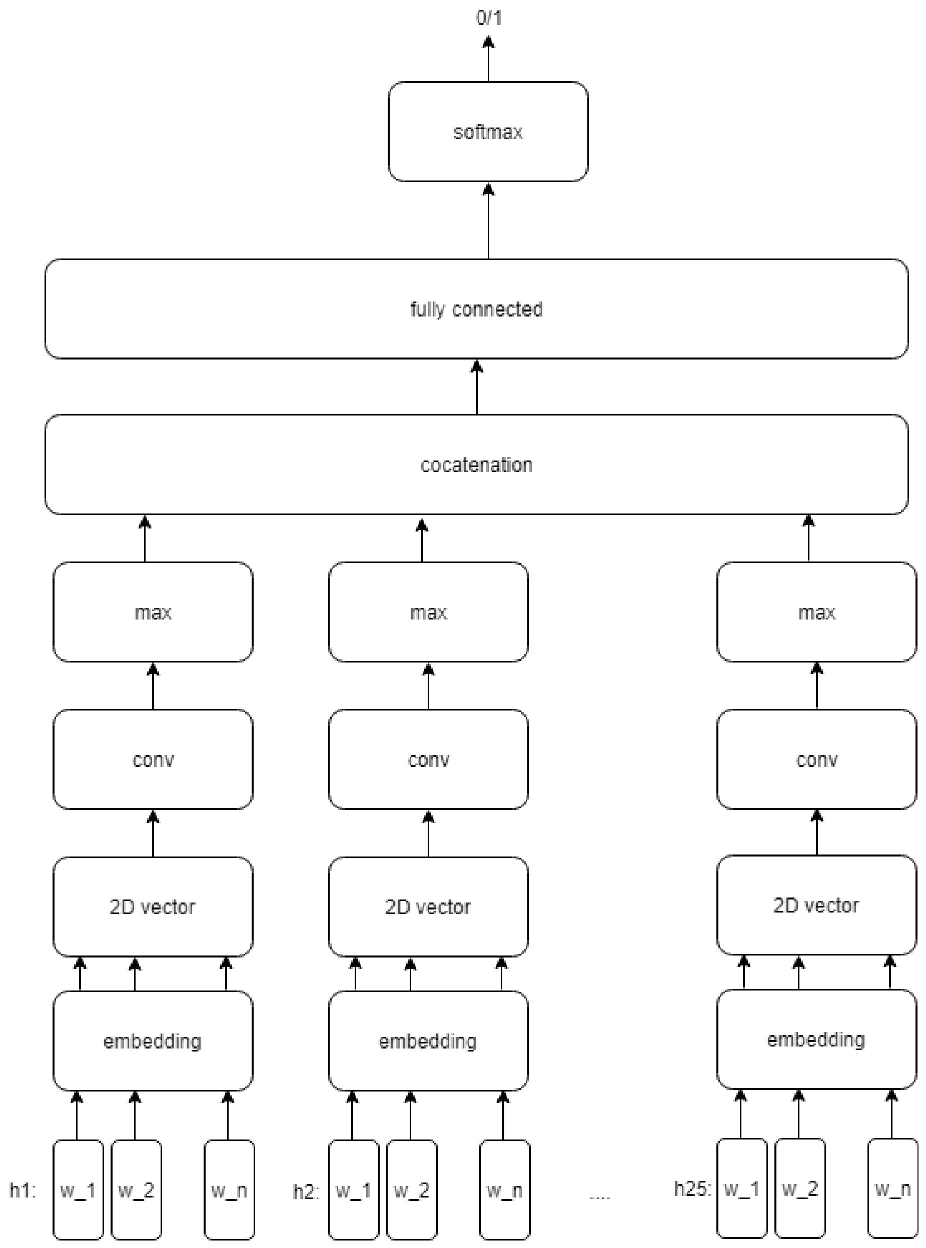


1. 25 headlines of each day are taken word-by-word as input by the model.
2. For each headline:
    - For each word: collect word embeddings.
    - Calculate *headline_encoding* as *f1*(embeddings) where *f1* is either sum or max.
3. Collect all *headline_encodings* and calculate *news_encoding* as *f2*(headline_encodings) where *f2* is either max or sum.
4. *news_encoding* vector is passed to a hidden layer followed by a softmax prediction layer.

- Tried varying number of epochs embedding size, number of hidden layers and combination of (f1 and f2) as (sum and max) or (max and sum).
- The highest test accuracy obtained was 51.5%, confusion matrix =

	0	1
0	71	115
1	68	124

CNNs



- Since we saw that a 2-gram models work better than unigram models, we tried to incorporate multi gram features using CNN model.
- Each headline is encoded as a 2D vector with first dimension as word sequence and second dimension as pre-trained word embeddings.
- This 2D vector is passed through a convolution layer followed by a max pooling layer.
- Kernel size: [4, embedding_size], this will make the output of convolution layer to be 1 dimensional
- The result of max pooling layer is concatenated for all the headlines and the concatenated vector is passed through one or two fully connected layers with dropout followed by a softmax layer.
- The highest test accuracy obtained was 50.5% and confusion matrix =

	0	1
0	50	136
1	51	141

Variation: Trainable word embeddings

- Adding a trainable embedding layer as the first layer with weights initialized by Glove word embeddings.
- This drastically increased the trainable parameters and increased the running time a lot. Because of time constraints, I could not experiment much with this model.
- The highest test accuracy obtained was 48.9% which is worse than the pretrained embedding model, confusion matrix

	0	1
0	75	111
1	82	110

Comparison of models:

Model	**Test accuracy**
Unigram + logistic	42
Bigram + logistic	57

BoW LSTM all headlines	45.2
BoW LSTM variation	52.3
Word embeddings	51.5
CNNs-Glove embeddings	50.5
CNNs-trainable embeddings	48.9

We decided to use CNNs for 8-K reports as CNNs can extract multigram features and is not as complex as LSTM model. Using an LSTM for 8-K reports has another disadvantage as it may memorize the entire 8-K dataset since it is small in size. Building a CNN model over time can help in extracting temporal information as well.

On a related note we have also in parallel worked on using 8-K financial reports from SEC which are filed whenever there is a significant business events like bankruptcies, layoffs, changes in senior management, change in credit etc. So, these reports serve as an important pulse of the organisation

Initial Model Baseline

The baseline model is based on one of the features used in the model viz., the earnings per share of the stock. The baseline consists of estimating the label based on the value of the earning per share. If it is positive then the prediction label will be UP, if negative then DOWN and if zero then it is neither.

Model 1 (Unigram Model)

The first model considered is a Random Forest with BoW input features from the text derived from the reports.

Model 2 (Using Temporal History)

The previous approach while significant did not incorporate any temporal information into the model As part of this model we try to incorporate the past history into the model in the way of historical changes in the price of the company say the last 5 price changes in the company data. We are currently working on the model we will report the initial results in a few days after this report

Model 3 (CNNs)

Applied the CNN model as explained before, for the 8-K dataset. Considered last 25 sentences (analogous to headlines) of event as input for CNN model as initial sentences of the event contain trivial information viz., company name, date etc. Since the model has large parameters, we ran it on first 1000 events with 80/20 split. Obtained test accuracy of 41.8% with confusion matrix:

	0	1	2
0	4	1	46
1	4	0	11
2	12	1	50

Model 4 (CNNs over time)
Since CNNs performed decently, we tried to incorporate temporal information into the model as explained before by passing the output of the concatenation layer at time step (t-1) as input to the the convolutional layer at time step t. The model constructed in this way should ideally encode the past history of the price trends as input to the current stock. The model however currently does not converge and we are currently investigating to see what might be the issue.

Model	Paper baseline	Our baseline
One feature(EPS)	49.4	36.4
Random Forest on text features(Unigram model)	54.4	38.5
CNNs	Not used	41.8

As we can see based on the baseline data the CNN tends to perform slightly better than the random forest unigram model.

Future work:

1. Complete the temporal model with Random forest taking features of the last few historical price changes.
2. Integrate unlabelled data to expand the size of the dataset in order to try more complex RNNs