1 Introduction

Investors' perceptions of companies are influenced by the news that is released about them in the financial sphere.Financial economists have undertaken event studies to get insight into how markets react to fresh information about companies. These event studies assess the impact of a certain event on a company's worth. However, analysing the influence of specific events on the stock markets is a time-consuming task. Automatic techniques for detecting economic events in text have been proposed as identifying news published about certain events in a timely manner allows researchers in the field of event studies to process more data in less time.
The goal is to automatically assign the presence of a set of predetermined economic event categories in a sentence of a news article.

2 Data Description

We have 497 news articles(.txt file) and an annotated file(.ann file) for each one of them.These annotated files contain information about which sentences of the articles have economic-event span and the company involved with that event.Total 10 types of company-specific economic/financial events were identified in the corpus. These are:-

- Buy ratings
- Debt
- Dividend
- Merger acquisition
- Profit
- Quarterly results
- Sales volume
- Share repurchase
- Target price

Using these news articles and annotated files,we firstly obtained **sent.txt** and **tuple.txt**.

sent.txt: It contains only those sentences from the articles which have some contextual information of at least one of the event types mentioned above.

tuple.txt: Corresponding to each sentence in sent.txt, tuple.txt contains tuple in form of (event type;event span;companies name). If no companies are

involved for that event in the sentence we used 'UNK' token.Also if the same sentence has more than one type of events than we will have that no of tuples separated with ' | '.

Example:

```
Absa , the South African banking group , said its integration with new majority-owner
Barclays was proceeding well as it announced a jump in its interim profit .
The country 's biggest retail bank saw pre-tax profit for the six months ending
September 30 rise from R3.33bn to R4.12bn ( Pounds 360m ) .
The strong performance allowed Absa to lift its interim dividend by 68.4 per cent to
160 cents a share .
```

sent.txt

```
Profit;announced a jump in its interim profit;Absa
Profit;saw pre-tax profit for the six months ending September 30 rise from R3.33bn to
R4.12bn ( Pounds 360m );Absa
Dividend;to lift its interim dividend by 68.4 per cent to 160 cents a
share;Absa|Dividend;The strong performance allowed;UNK
```

tuple.txt

With these sent.txt and tuple.txt we obtained a **pointer5.txt** file. Also we added 'unused1'(which will be used if no company is present in that sentence) token before each sentence in sent.txt and created **sent5.txt** file. Now corresponding to each sentence in sent5.txt, pointer5.txt contains tuples separated with pip(' | ') . Each tuple has - (Event type ; event_start_index ; event_end_index ; comp_start_index ; comp_end_index) ;

Example:

```
unused1 Absa , the South African banking group , said its integration with new
majority-owner Barclays was proceeding well as it announced a jump in its interim
profit .
unused1 The country 's biggest retail bank saw pre-tax profit for the six months
ending September 30 rise from R3.33bn to R4.12bn ( Pounds 360m ) .
unused1 The strong performance allowed Absa to lift its interim dividend by 68.4 per
cent to 160 cents a share .
```

sent5.txt

```
Profit 21 27 1 1
Profit 7 25 0 0
Dividend 6 19 5 5 | Dividend 1 4 0 0
```

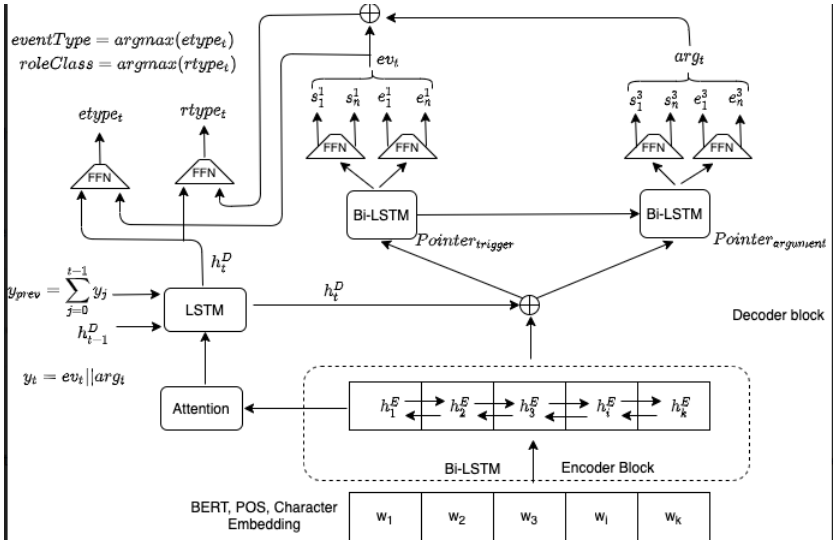
pointer5.txt

3 Preprocessing

To get train,dev and test files which can be fed to the model. Done in two steps:

- Creating sent5.txt and pointer5.txt
  It is generated using news articles and its annotated files. Description of these files is given above. Code and further explanation is present in - **Prep5.ipynb**
- Feeding sent5.txt and pointer5.txt to **helper_joint_ee.ipynb**. It tokenizes and preprocess the sent5.txt using pre-trained 'bert-base-cased' model.Finally we divided sent5 and pointer5 in train,dev and test files (70%-15%-15%) (code in **File_generation.ipynb**).Each train,test and dev has its own .sent and .pointer files which are final processed files and will be fed directly for training and testing.
  Final files that will be fed to the models are present in **'Dataset'** folder.It has following files:
  - Train.sent
  - Test.sent
  - Dev.sent
  - Train.pointer
  - Test.pointer
  - Dev.pointer
  - W2v.txt (click to view)
  - event_names.txt

4 Model Architecture



- As evident from the above pipeline ,the input the model consists of : the word embeddings , the corresponding mask , the character embeddings and the positional sequence.
- The Character and the Bert Embeddings are concatenated together for the better representation of the context.

$$enc\_inp\_size = bert\_base\_size + char\_feature\_size$$

$outputs = model(src\_words\_seq, bert\_words\_mask, src\_words\_mask, src\_chars\_seq, positional\_seq, max\_trg\_len, None, None, False)$

- The input is then fed into the Encoder part of the Seq2Seq Model which uses a BiLSTM to generate the context vector for the decoder block.
- The decoder first passes the inputs from the encoder to two consecutive layers of attention followed by a LSTM recurrent neural network.
- The decoder architecture is complex and effective in the sense that it has a separate LSTM for each of the pointers.
- The output from these LSTM's is then fed to a feed forward neural network.
- The decoder output's 5 instances corresponding to trig_s.unsqueeze(1), trig_e.unsqueeze(1), ent_s.unsqueeze(1), ent_e.unsqueeze(1), (hidden, cell_state), trig_et, ent_argt, event_types.unsqueeze(1)
- The trig_S,trig_e,ent_e,ent_S are obtained by applying softmax function on top of the tensor obtained by multiplying the corresponding pointer weights with the lstm output.
- The pointer weights in turn are obtained through normalizing the probability of each word index to be the strat index of arguments.
- The decoding then continues for a reasonable number of time_steps where the input for the current time step is obtained by concatenating the inputs from the previous time step and the input obtained from the first time step of decoding.

```
trig_s = torch.cat((trig_s, cur_trig_s), 1)
trig_e = torch.cat((trig_e, cur_trig_e), 1)
ent_s = torch.cat((ent_s, cur_ent_s), 1)
ent_e = torch.cat((ent_e, cur_ent_e), 1)
trg_type = torch.cat((trg_type, cur_trg_type),1)
```

- The decoder takes the context vector from the encoder along with the hidden layer obtained from the attention module.

$dec\_outs = self.decoder(prev\_tuples, dec\_hid, enc\_hs, src\_mask, trigger, entity, None, None, is\_training)$

Finally the loss is calculated by summing the total loss obtained from the event type criterion and the 4 pointer type criterion (entity start index entity stop index ,company start index and company end index).
Finally for evaluating the model, we use the dev.sent and the dev.pointer to predict the event corresponding to it.

5 Results

- The training model(code - **training.ipynb**) ran for around 20 epochs after encountering an early stop situation.
- It was able to predict the event type, the event and the event span correctly in around 55 out 300 predictions.
- Model was saved with the name- **model_bert_19_10.h5py** (click to view).
- For testing the model we used the test.sent and test.pointer files(code - **test.ipynb**)
- The test size was around 330.

```
340       408     51
no of correctly identified triggers= 80
no of correctly classified triggers= 70
```

- The model was able to correctly identify around 80 out 330 samples and correctly classify around 70 out of 330 samples.
- However, when both identification and classification of triggers is considered the model is able to do best at only 51 predictions out of the 330 predictions.
- F1 Score to identify event type: **0.214**

6 Conclusion

The model performed better than the expectations as there are around 10 classes and around uncountable possibilities for the pointers to be assigned.But, the model has an accuracy of around 25% when compared to a random accuracy to nearly null.So,the model has surely learnt reasonably in the 20 epochs.

7 Future Scopes

The accuracy and the F1 score could be further enhanced through the use of part of speech tag embeddings and the inclusion of the relations between the events with the entities.The model could also be extended to predict the roles of each event type in the corresponding span.
We leveraged glove word embeddings in the encoder and decoder side. Future works can explore BERT's contextualized representations which has proven success in a variety of NLP tasks. Additionally, we can attempt to directly generate triplets using sequence to sequence autoregressive denoising pre-trained models like BART, which are more suited for Encoder-Decoder architecture as compared to BERT due to its training objective. Finally, we can try out our model architecture on other parallel extraction tasks like Sentiment Triplet extraction or Relation Triplet extraction.

8 References

- PaulC.Tetlock.2007. Giving Content to Investor Sentiment: The Role of Media in the Stock Market. The Journal of Finance.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topi'c, Tomoko Ohta,Sophia Ananiadou,and Jun'ichi Tsujii. 2012. brat: a Web-based Tool for NLP-Assisted Text Annotation. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL'12)
- Effective Modeling of Encoder-Decoder Architecture for Joint Entity and Relation Extraction,Tapas Nayak, Hwee Tou Ng
- Economic Event Detection in Company-Specific NewsText, Gilles Jacobs, Els Lefever and´eroniqueHoste