NLP Project 1

Fine Grained Sentiment Analysis on Financial Microblogs

0. Team Members

- b03902089 資工四 林良翰
    - model training & testing
    - model fine-tunning
    - report
- b03902093 資工四 張庭維
    - preprocess: clean up corpus, word index, word embedding
- b03902065 資工四 陳奕先
    - preprocess: self-defined features
    - report

1. Preprocessing

1.1 Clean Corpus

- Remove some hashtags or urls to lower the size of embedding matrix and noises.

```
for i, data in enumerate(corpus):
    data = re.sub(r'\$[A-Za-z0-9]*[ ,]?', '', data) # remove $ target
    data = re.sub(r'@[a-zA-Z0-9]*', '', data) # remove @ tag
    data = re.sub(r'http.*[a-zA-Z0-9]?', '', data) # remove url
    data = re.sub(r'&#39;', '\'', data) # fix '
    data = re.sub(r'[0-9.,]*[0-9]+', '', data) # remove numbers
    data = re.sub(r'(~?&[a-z]*;)', '', data) # remove Latex
    data = re.sub(r'["$%&()*+\-/:;<=>@\^_`{|}~…—\n\t•\[\]]|[.]+\.' , '
', data)
    data = re.sub(r'#', ' #', data) # split continuous hashtag
    data = re.sub(r' +', ' ', data) # remove space redundancy
    corpus[i] = data
```

1.2 Tokenize Texts

- Generate word index and tokenize texts (tweets) into sequence of numbers.

```
{'the': 1,
 'to': 2,
 'a': 3,
 'of': 4,
 'in': 5,
 ...}
```

- Pad the sequence with prefix zeros to equalize the sequence length.

1.3 Construct Word Embedding Matrix

- Selected features: bearish / bullish cdidf, word sentiments, word vectors

```
num_words = len(word_index) + 1
emb_dim = 300 + 3
embedding_matrix = np.zeros((num_words, emb_dim), dtype=np.float32)

for (word, index) in word_index.items():
    try:
        if word[0] == '#':
            content = hashtag_dict.loc[word[1:]]
        else:
            content = word_dict.loc[word]

        bear = content['bear_cfidf'] / 100
        bull = content['bull_cfidf'] / 100
        sentiment = content['market_sentiment']
        word_vec = content['word_vec']
        embedding_matrix[index] = np.asarray(word_vec + [bear, bull,
sentiment])
    except:
        continue
```

1.4 Self Defined Feature

- Multiply the word sentiments for each sentence in tweet, and sum up the sentiments of the tweet.

```
S = []
for data in corpus:
    S_data = 0.0
    sentences = re.split('[.,!?]', re.sub(r'#[A-Za-z]*', '', data))

    for sentence in sentences:
        S_sentence, v = 1., 0.
        words = [w for w in re.split(' ', sentence) if w != '']
```

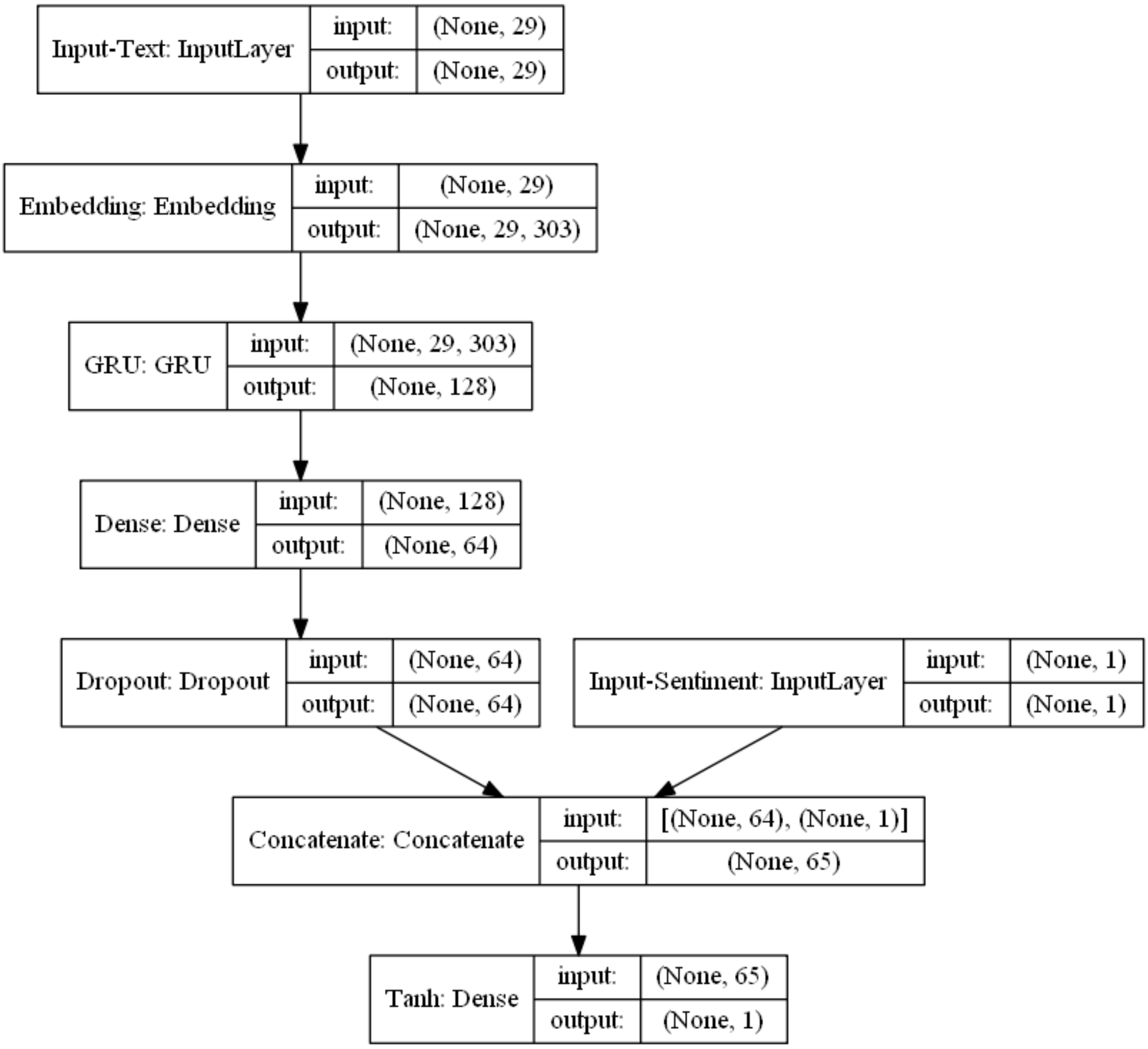
```
        for word in words:
            try:
                s = word_dict.loc[word.lower()]['market_sentiment']
                S_sentence *= s
                v += 1
            except:
                pass

        if v > 0:
        S_data += np.sign(S_sentence) * np.abs(S_sentence) ** (1/v)

    S.append(S_data)
```

2 Models

2.1 GRU (Gated Recurrent Network)

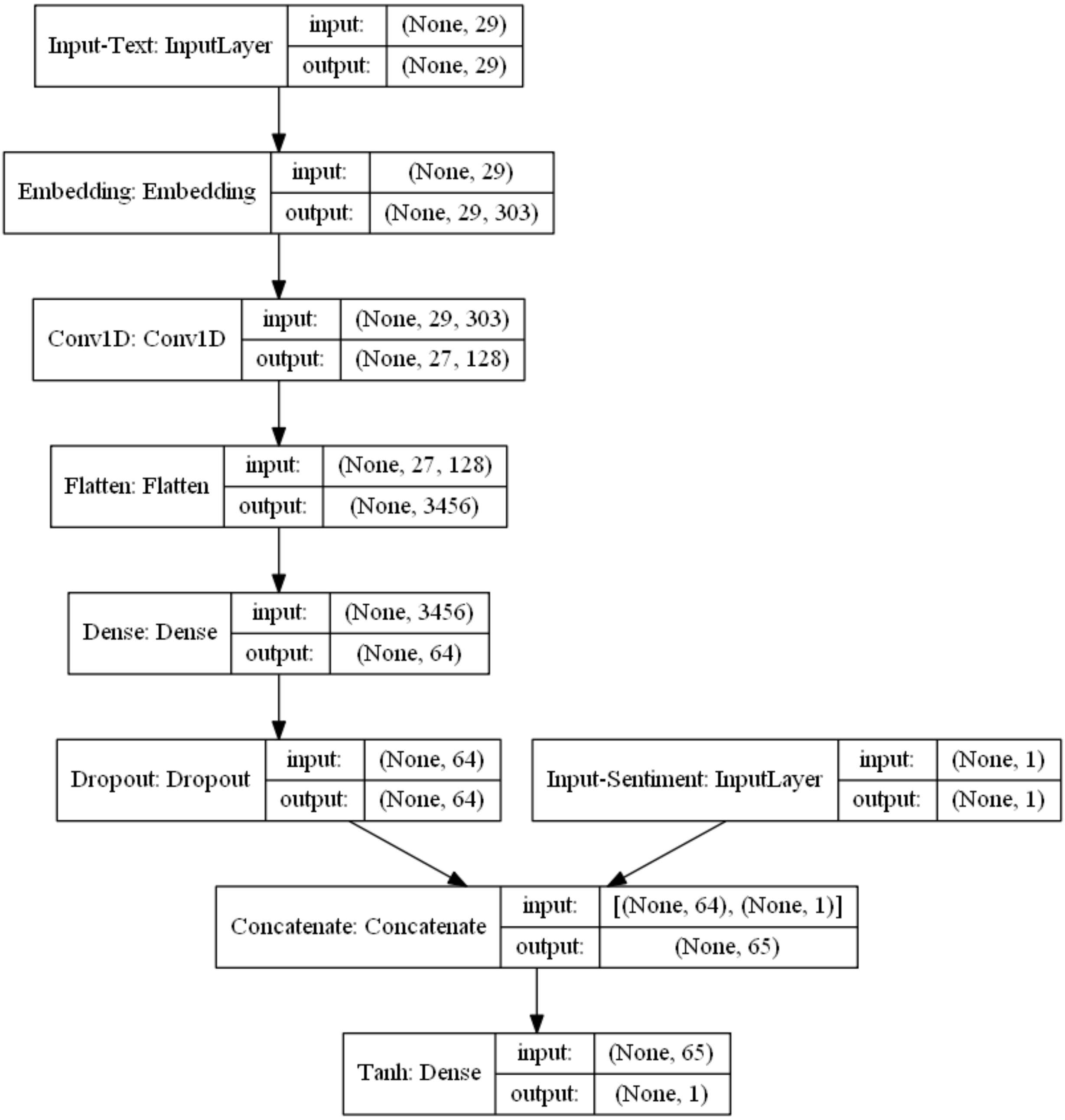


- Input-Sentiment is self-dedined feature.

2.2 LSTM (Long-Short Term Memory)

- The technique of the LSTM is similar to the GRU.
- Generally, performance of the LSTM model would be slightly diffetent from that of the GRU model.
- Therefore, we just replace the GRU units with the LSTM units for the LSTM model.

2.3 Conv1D (Convolution 1D)



3. Evaluation

3.1 Metrics

- Mean Square Error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2$$

- Accuracy

$$ACC = \frac{1}{N}\left(\sum(true\ positive) + \sum(true\ negative)\right)$$

- F1 Score
  - Recall & Precision

$$\begin{cases} recall = \frac{\sum(truepositive)}{\sum(conditionpositive)} \\ precision = \frac{\sum(truepositive)}{\sum(predictedconditionpositive)} \end{cases}$$

  - F1

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$

3.2 Result without self-defined feature

Metric \ Model	GRU	LSTM	Conv1D	GRU + LSTM + Conv1D
MSE	0.0784	0.0718	0.0738	0.067645
Accuracy	80.60%	82.18%	81.39%	82.97%
F1 Score	0.8575	0.8719	0.8626	0.8765

3.3 Result with self-defined feature

Metric \ Model	GRU	LSTM	Conv1D	GRU + LSTM + Conv1D
MSE	0.0757	0.0716	0.0753	0.067557
Accuracy	82.02%	82.97%	80.13%	83.28%
F1 Score	0.8680	0.8779	0.8527	0.8797

4. Discussion & Conclusion

- Cleaning up the corpus can improve the performance. (Accuracy $78\% \rightarrow 80\%$)
- LSTM has best performance among three models (GRU, LSTM, Conv1D).
- Ensembling models can produce better results than single LSTM.
- Adding self-defined feature can improve MSE slightly, and increase the accuracy by average 0.32%.