Chinese Restaurant Data Analysis

Ruyi Yan, Luwei Liang, Yilun Zhang

April 4, 2019

Overview

Review of Previous Work

Logistic regression based on word frequency

- Simple logistic regression: Y=review$stars

price	quick	lunch	nice	good	an	highly
2.3783480	1.9992634	1.7940573	1.7322821	1.5253039	1.5221779	1.4696290
on	very	rice	it's	many	favorite	this
1.3195870	1.0231201	1.0158467	0.9341363	0.8460795	0.8168955	0.7558957
they're	ma	peanut	tofu	portions	cheap	
0.7149960	0.6793929	0.5752047	0.5533853	0.4724339	0.2245626	

- Problem: already-known positive/negative words('nice', 'good'...)

Review of Previous Work

Logistic regression based on word frequency

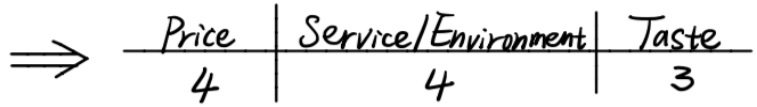
- Extract and delete these already-known positive/negative words
  - It's hard to identify/figure out the already-known positive/negative words
  - Even if we delete these words, the result for logistic regression is not so good, especially in small-sample cases.
- Using TF-IDF to optimize the word frequency matrix

Review of Previous Work

Scorecard of each review

- For each review, evaluate the score in Price, Service, Food.

"Decent fast inexpensive food. Large portions. Good value. It was cafeteria style so the service is fast. The lo mein was pretty good. The general zhos chicken was OK (small chicken, big breading). Pepper chicken and bbq chicken good. Not fancy but fills you up."



- Because of the complexity of language, it's hard to extract the 3 aspects from each review precisely.

Scoring for 'Food', 'Price', 'Service'

Overall thoughts

- Extract adjtives near specific words.

- If a restaurant has good service, the word 'service' will appear with high frequency and described by positive words mostly.

Scoring for 'Food', 'Price', 'Service'

Extract all adjtives from reviews

- Text processing1:
  - not great $\Longrightarrow$notgreat
  - not only quick but also friendly $\Longrightarrow$quick freiendly
  - fast food $\Longrightarrow$fastfood
  - high quality $\Longrightarrow$great
  - 5 stars $\Longrightarrow$fivestars
  - .....

Scoring for Food, Price, Service

Extract all adjectives from reviews

- Tagging and extracting adjectives with the help of nltk:

```
nltk.pos_tag(word_tokenize('It\'s a fivestars restaurant!'))
```

```
[('It', 'PRP'),
 ("'s", 'VBZ'),
 ('a', 'DT'),
 ('fivestars', 'JJ'),
 ('restaurant', 'NN'),
 ('!', '.')]
```

```
detact_adj('It\'s a fivestars restaurant!')
```

```
['fivestars']
```

- Only kept words appeared more than 100 times, finally got an adjective list with 1851 words.

Scoring for Food, Price, Service

Get the score of each adjective

- Text processing2:
  - Same work as in Text processing1
  - Delete punctuations and stopwords
  - Delete words 'pretty', 'very', 'always' (most of them are describing an adjective but could be regarded as adj by nltk), 'chinese'
  - For each review, only keep the adjectives from the adjList.

Scoring for Food, Price, Service

Get the score of each adjective

- Perform a logistic regression using reviews with only adjectives

$$Y = \begin{cases} 1, & star \geq 4 \\ 0, & star < 4 \end{cases},$$

$$X = \begin{array}{c} \begin{array}{cccc} nice & notgreat & bad & \ldots \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & \ldots \\ 0 & 1 & 0 & \ldots \\ 0 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix} \end{array} \tag{1}$$

  - Put no penalty on the regression coefficients to prevent over-fitting problem, because our goal here is to get the sentiment score for each adjective rather than precise prediction.

Scoring for Food, Price, Service

Get the score of each adjective

- Use the regression coefficients as the sentiment score for each adjectiv, and only keep words with the absolute value of coefficient ≥ 0.2
- Delete some weird words: 'bit', 'often'...
- Finally there're 975 words remaining, with 462 positive words and 513 negative words.
Words	...	notgood	speedy	friendly	oily	...
Score	...	-0.92	0.83	0.77	-0.51	...

Scoring for Food, Price, Service

Break Sentence

- Text processing3:
  - Same work as in Text processing1
  - Keep punctuations(,.?!) as sentence breaker
  - Some specific words like 'but', 'otherwise' should also be sentence breaker .
  - Example:

Price is reasonable but service is notgood, the food tastes okay.

$\Downarrow$

Price is reasonable ||service is notgood||the food tastes okay.

Scoring for Food, Price, Service

- With the scored adjList and sentence breaker, we can extract the scored adjs near specific words for each restaurant.

- 'price' for QQ Express:

it	gives	you	a	wider
0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
selection	of	items	at	the
0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
comparable	price			
0.5797432	0.0000000			

it's	also	a	really	good	price
0.000000	0.000000	0.000000	0.000000	0.665883	0.000000
for	the	amount	you	get	
0.000000	0.000000	0.000000	0.000000	0.000000	

Scoring for Food, Price, Service

- Calculate the average score of the adjs near 'price', then we get the score of price for QQ Express:

```
> score_summary(score_result[[1]], score_result[[2]])[[1]]
reasonable      worth       nice      solid       good
 0.8307113  0.6871596  0.4174705  1.2248443  0.6658830
comparable       good    average    average  excellent
 0.5797432  0.6658830  0.0000000  0.0000000  1.3567568
> score_summary(score_result[[1]], score_result[[2]])[[2]]
[1] 0.6428452
```

- We calculate the average scores near the words 'price', 'service' and 'food' for each restaurant:

QQ Express	Price	Service	Food
Score	0.64	0.26	0.14

Scoring for Food, Price, Service

- Restaurants with only one or two reviews with these specific words could be misleading

- To prevent those misleads, we introduced a **Factor of Credibility** to indicate how credible the score is:

$$Price_{res,new} = Price_{res,old} * C_{price,res}$$

$$C_{price,res} = \frac{\#\ 'price'}{\#\ reviews_{res}}$$

- 'price' for QQ Express Hong Kong Station

Score	Old	New
QQ	0.64	0.11

Score	Old	New
Hong Kong	0.65	0.04

Scoring for Food, Price, Service

- Then we calculated the new average score for each restaurant, and ranked them for each words
- Only keep restaurants with more than 20 reviews.
- Top 5 Restaurants for 'Price', 'Service' and 'Food' in Madison.

Price

Rank	Restaurant
1	QQ Express
2	Hong Kong Wok
3	Jade Garden
4	Happy Wok
5	Double 10 Mini Hot Pot

Service

Rank	Restaurant
1	Double 10 Mini Hot Pot
2	Tavernakaya
3	Umami Ramen & Dumpling Bar
4	Imperial Garden Chinese Restaurant
5	Great Wall

Food

Rank	Restaurant
1	ZenZen Taste
2	Chili King
3	Hong Kong Wok
4	Double 10 Mini Hot Pot
5	Double 10

Scoring for Food, Price, Service

- Which word should the business owner value most?
- We ranked all the restaurants in the U.S, and calculate the average stars it got from customers.

Restaurant	Price	Service	Food	Avg. Stars
...	...	...	...	...
King's Chef	16	153	338	4.19
China Passion	25	60	179	4.53
Tasty China	83	328	634	4.10
...	...	...	...	...
QQ Express	1126	1047	49	3.29
...	...	...	...	...

Scoring for Food, Price, Service

- From the rank table above, we could calculate the correlation coefficient between the rank in one word with the Stars it get, which indicates the importance of this aspect:

$$\rho_{food} = 0.69$$

$$\rho_{service} = 0.57$$

$$\rho_{price} = 0.17$$

- And the modified rank considering the importance of each aspect should be :

$$Rank_{price,new} = Rank_{price,old} * \rho_{price}$$

Scoring for Food, Price, Service

- After recalculation we can get the modified rank table. Summing the modified rank for each word, the restaurant with smallest rank-sum should be the best.

- Then we resorted the original rank table by the modified rank-sum result, here is the final result:

Scoring for Food, Price, Service

	food	service	price	final_rank
Double 10 Mini Hot Pot	4	1	5	1
Chili King	2	6	32	2
Double 10	5	11	14	3
Great Wall	6	5	32	4
Tavernakaya	8	2	37	5
Imperial Garden Chinese Restaurant	7	4	36	6
Hong Kong Wok	3	26	2	7
Chen's Dumpling House	11	12	21	8
Umami Ramen & Dumpling Bar	19	3	22	9
ZenZen Taste 四合院	1	23	32	10
DumplingHaus	15	8	28	11
Happy Wok	22	9	4	12
Noodles & Company	21	7	18	13
Ichiban Sichuan	9	20	23	14
VIP	14	17	13	15
Magic Wok	16	13	32	16
Orient House	20	19	7	17
QQ Express	24	16	1	18

Conclusion and Strong/Weak Points

- Conclusion:
  - For chinese restaurants, taste, service and price are all important. But it seems that taste and service weighs more than price in getting more stars from customers.
  - For each restaurant, it can get its rank in these three aspects as well as the final overall rank, indicating the business owner which aspect should be improved.

Conclusion and Strong/Weak Points

- Strong Points:
  - We have introduced the **Factor of Credibility** when calculating the average score for each words, which prevents the unreasonably high/low scores caused by only few reviews.
  - We calculate the correlation coefficients between the rank of the word and the stars, and use them to produced the final weighted rank table, which gives business owners an informative rank considering the importance of different aspects.
  - The business owner could not only get the rank of his/her restaurant, but also get more detailed information from the adjs/sentences about why the restaurant gets high/low rank in this aspects.

Conclusion and Strong/Weak Points

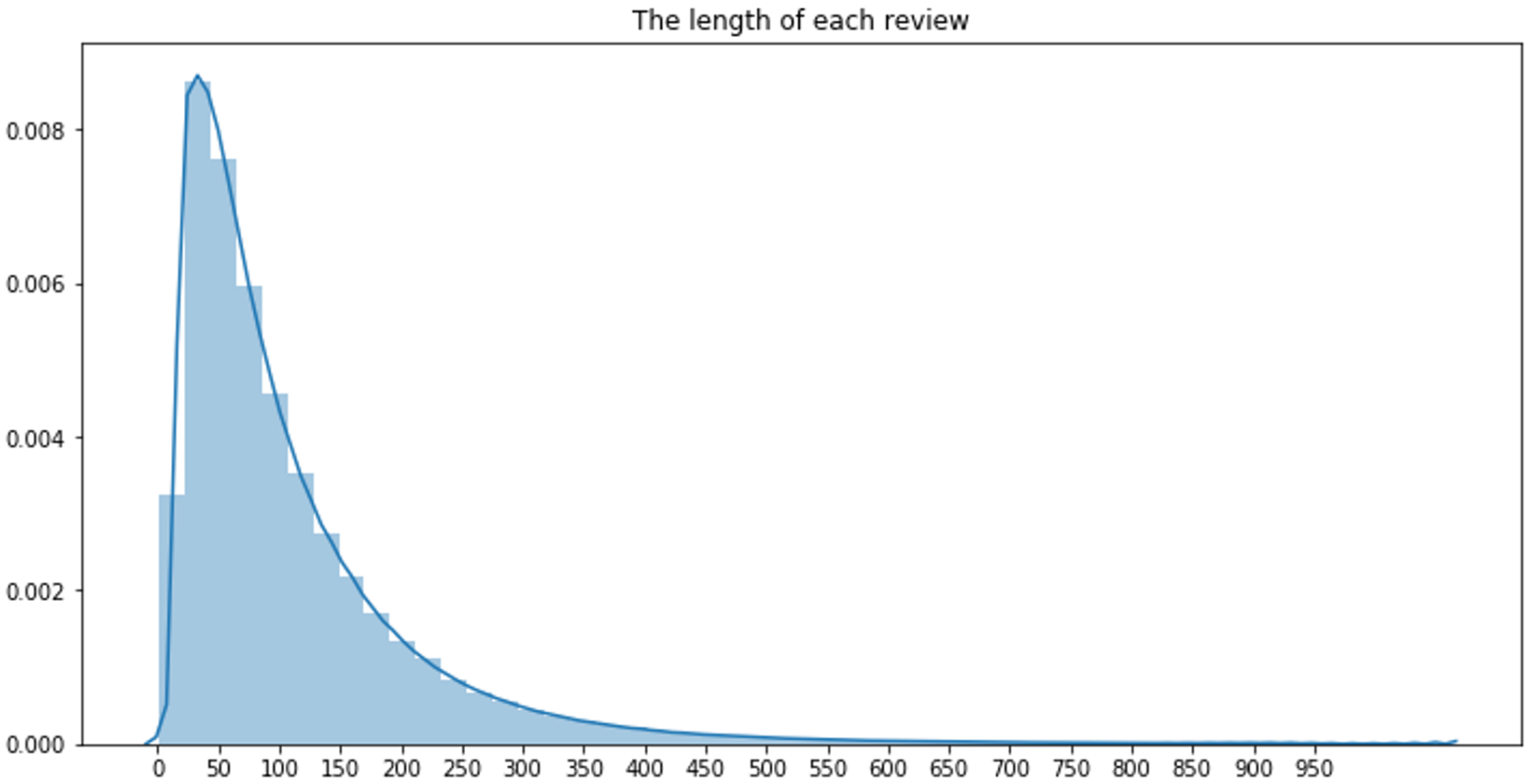
- Weak Points:
  - There do exist many other ways to describe the taste, service and price, which would be missed by the adj-extacting method.
  - There should be many other aspects which may affect the stars got, such as dim sum for Nani Restaurant, which were not included in our analysis.
  - Of course, the final scores we get from the adj-scoreList and sentence breakers are informative, but not the most accurate.

Prediction

Overall thoughts

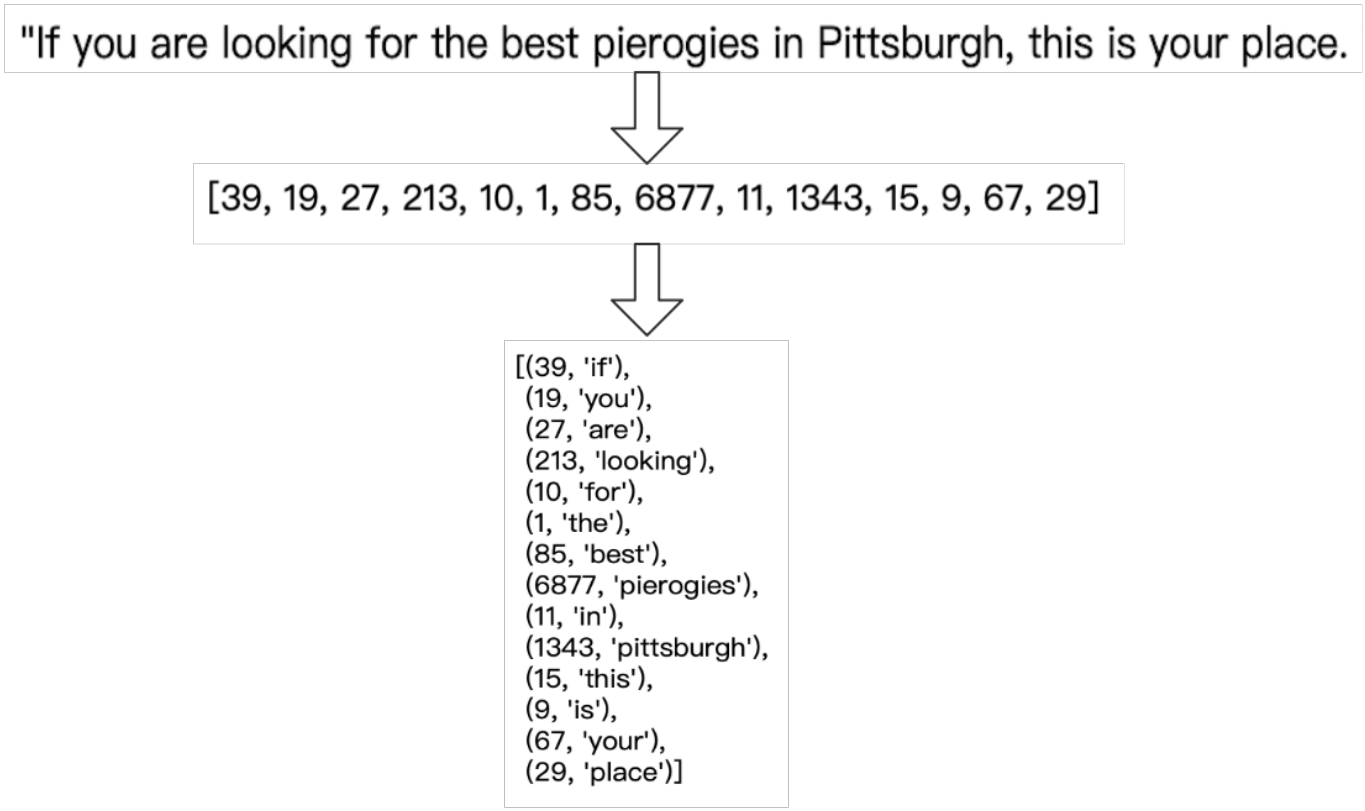
- Goal: Predict the rating of the reviews mainly based on the text data.
- Since it belongs to NLP problem, we choose LSTM model to solve this sentiment classification problem.

EDA



- From above, we can find most reviews are less than 200 words.

Tokenize



- Since the average adult knows 20000 to 35000 words, we choose 20000 top frequency words to construct a dictionary.

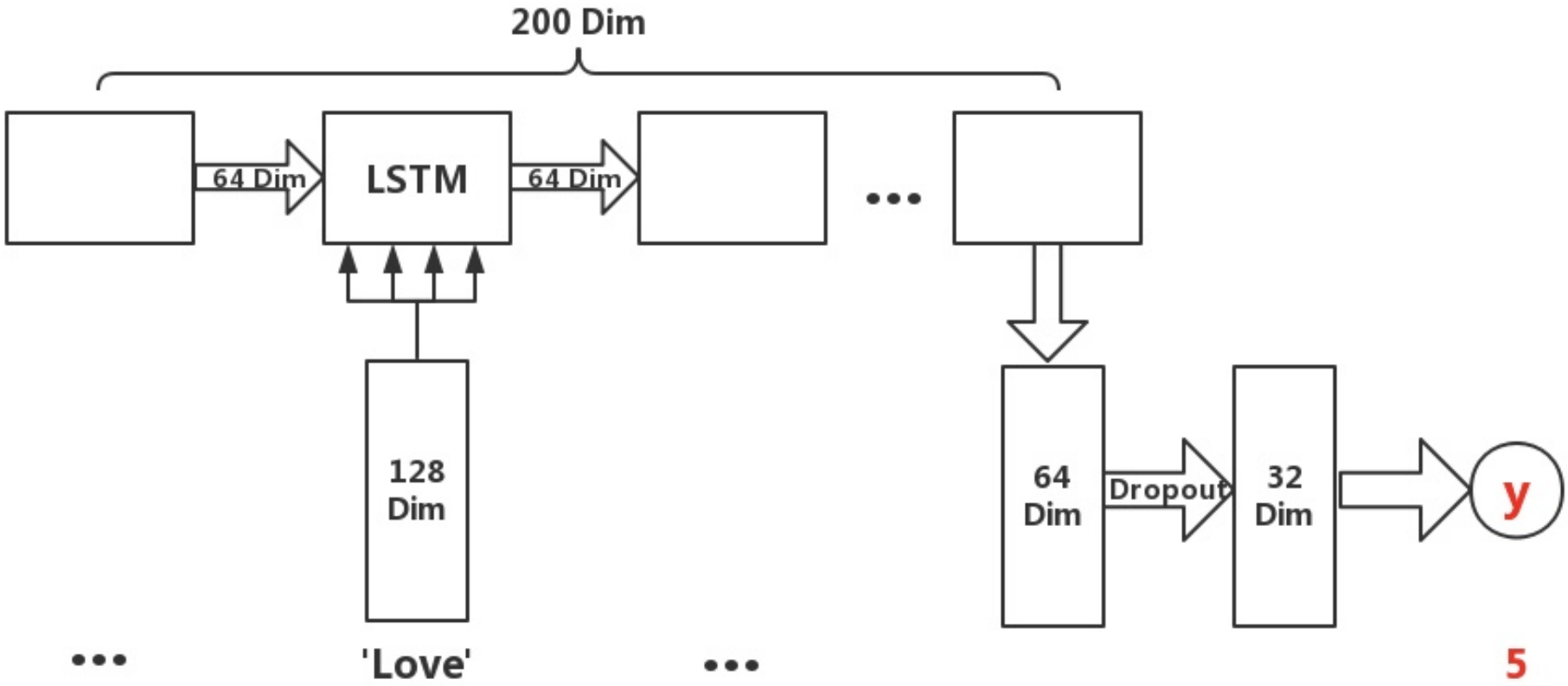
LSTM

- In general, LSTM is an improved version of RNN, which is used to solve the sequential problem and at the same time overcome the long-term dependencies defect.

- Our model constructed via four main layers. The Embedding layer is used to vectorize the chosen words in our corpus, and it overcomes the high dimension sparse problem caused by the one-hot encoder. The LSTM layer is used to solve the sequence problem and solve the vanishing gradient problem by the gate functions. The Dense layer aims to make the model more robust based on feature fusion. The Dropout layer can be applied to handle the overfitting problem.

LSTM

- Flow chart



LSTM

- MSE/CrossEntropy

Layer	Output Shape	Param #
Input	(None, 200)	0
Embedding	(None, 200, 128)	2560000
LSTM	(None, 64)	49408
Dense	(None, 32)	2080
Dropout	(None, 32)	0
Dense(linear)	(None, 1)	33

Layer	Output Shape	Param #
Input	(None, 200)	0
Embedding	(None, 200, 128)	2560000
LSTM	(None, 64)	49408
Dense	(None, 32)	2080
Dropout	(None, 32)	0
Dense(softmax)	(None, 5)	165

LSTM

- Final result

Method	Loss	Accuracy	Score On Kaggle
MSE	0.6525	0.5332	0.61528
CrossEntropy	0.7698	0.6810	0.69130

Thank you!