Analyzing Sentiment of Reddit Comments for Content Creators And Users

Team member:
Ta-Wei Wang, Ting-Hsuan Chen, Ting-Ann Lu, Luke Hong

Table of contents

01

Introduction

Background And Motivation

About Reddit

- one of the largest forum community social networks on the internet
- over 430 million monthly active users and over 100,000 active communities
- users can comment on and upvote/downvote posts and comments

Motivation

From the perspective of content creators

understand the sentiment of Reddit comments on their posts to assess engagement and reach

Motivation

From the perspective of Reddit users

understand the sentiment of Reddit comments by classifying the emotions of the posts’ comments

02

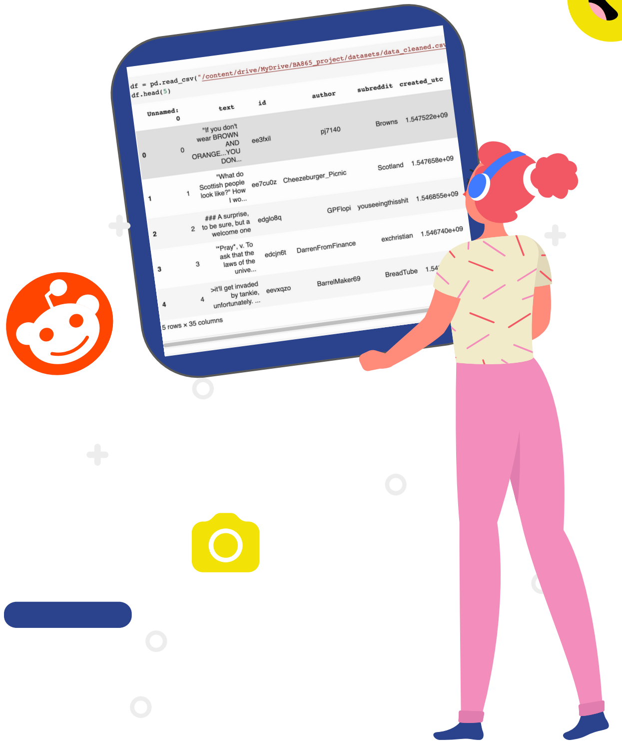
Dataset

Data Collection And Data Cleaning

Data source: [GoEmotions Dataset]

3 datasets:

1st: 70,000 rows / 37 columns
2nd: 70,000 rows / 37 columns
3rd: 71,225 rows / 37 columns



```
df = pd.read_csv("/content/drive/MyDrive/BA865_project/datasets/data_cleaned.csv
df.head(5)
```

	Unnamed: 0	text	id	author	subreddit	created_utc
0	0	"If you don't wear BROWN AND ORANGE...YOU DON...	ee3fxil	pj7140	Browns	1.547522e+09
1	1	"What do Scottish people look like?" How I wo...	ee7cu0z	Cheezeburger_Picnic	Scotland	1.547658e+09
2	2	### A surprise, to be sure, but a welcome one	edglo8q	GPFlopi	youseeingthisshit	1.546855e+09
3	3	"*Pray*, v. To ask that the laws of the unive...	edcjn6t	DarrenFromFinance	exchristian	1.546740e+09
4	4	>it'll get invaded by tankie, unfortunately. ...	eevxqzo	BarrelMaker69	BreadTube	1.54

5 rows × 35 columns

text	id	author
subreddit	link_id	parent_id
created_utc	rater_id	example_very_unclear

Positive		Negative		Ambiguous
admiration 👏	joy 😃	anger 😡	grief 😢	confusion 😕
amusement 😂	love ❤️	annoyance 😒	nervousness 😬	curiosity 🤔
approval 👍	optimism 🤞	disappointment	remorse 😔	realization 💡
caring 🤗	pride 😌	disapproval 👎	sadness 😞	surprise 😲
desire 😍	relief 😅	disgust 🤮		
excitement 🤩		embarrassment 😳		
gratitude 🙏		fear 😨		

Data Cleaning and preprocessing

	text	id	author	subreddit	created_utc	rater_id	anger	disgust	fear
18277	She’s horrible	ee1r7x8	HoldenCaulfield7	EDAnonymous	1.547476e+09	16	0	0	0
38338	She’s horrible	ee1r7x8	HoldenCaulfield7	EDAnonymous	1.547476e+09	1	0	0	1
50778	She’s horrible	ee1r7x8	HoldenCaulfield7	EDAnonymous	1.547476e+09	23	0	0	1
55497	She’s horrible	ee1r7x8	HoldenCaulfield7	EDAnonymous	1.547476e+09	11	0	0	0
63965	She’s horrible	ee1r7x8	HoldenCaulfield7	EDAnonymous	1.547476e+09	74	1	0	0

```
len(clean['text'].unique())
```

57732

Data Cleaning and preprocessing

Punctuation

Contraction

Possible misspelling

A surprise, to be sure, but a welcome one

a surprise to be sure but a welcome one

Lol! ♿♿♿

lol

03

Modeling

Own model And Pre-trained Model

Process

1. **Tokenize**
   Bert
   Keras

2. **Word Embedding**
   Bert
   Glove

3. **Model**
   Convolution
   Regression

4. **Outcome**
   Kfold

First Model

Preprocessing

Data Cleaning

Tokenization

Padding,
Attention Flag

Bert Modeling

Fit to BertModel

Modeling

Logistic
Regression

Outcome

Accuracy

Tokenization

- We used the 'BertTokenizer' function to tokenize the text
- Align the results with padded 0's and added an attention flag for BERT model
- Take the first 1500 rows as example:
  - Total number of words: 18740
  - Max length of a sentence: 41
  - 3705 unique words

Sentence

	Sentence
0	if you do not wear brown and orange you do not matter we need a tshirt with that on it asap
1	what do scottish people look like how i would love to have been there to take a swing at that softball
…	…
1499	they look like such fucking ocs designed them i do not know what you expected he phones in video games all the time

Tokenization & Padding

Sentence Tokenized

0	1045	2097	…	0	0
1	1998	2024	…	0	0
…	…	…	…	…	…
1499	1057	3501	…	0	0

BERT Model

- Run our sentences through BERT
  - BERT adds a token for classification at the beginning of every sentence.
  - The output corresponding to that token can be thought of as an embedding for the entire sentence.
- The results are returned as 'last_hidden_states' for calculations

Limitation:

- BERT requires huge quantity of CUDA memory.
- Though we managed to run our calculations on dedicated GPU, we are only able to import proportions of the dataset without exceeding the RAM limit.

Sentence Tokenized

0	1045	2097	...	0	0
1	1998	2024	...	0	0
...	...	...	...	...	...
1499	1057	3501	...	0	0

→ **Bert** →

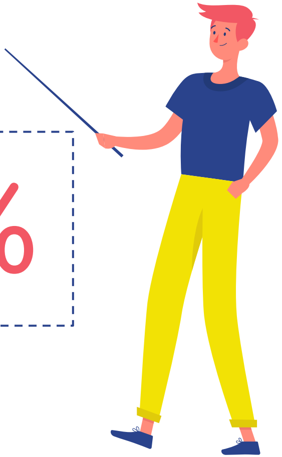
Sentence embeddings

0	6.2360e-02	3.9872e-01	...	4.4866e-01
1	2.1780e-01	2.3444e-01	...	3.0756e-01
...	...	...	...	...
1499	1.0603e-01	1.2828e-01	...	5.3088e-01

Testbench Evaluation

- Fit the embedded results to a logistic regression
- Evaluate the accuracy of each label.

90.45%



admiration	0.869	fear	0.974
amusement	0.935	gratitude	0.951
anger	0.882	grief	0.993
annoyance	0.794	joy	0.915
approval	0.752	love	0.951
caring	0.915	nervousness	0.984
confusion	0.902	optimism	0.895
curiosity	0.846	pride	0.98
desire	0.977	realization	0.873
disappointment	0.843	relief	0.99
disapproval	0.82	remorse	0.99
disgust	0.931	sadness	0.928
embarrassment	0.964	surprise	0.925
excitement	0.922	neutral	0.611

Second Model

Tokenization

Tensorflow.
Tokenize

Word Embedding

GloVe.6B
100 x 100

Define Model

Convolutional
Layers

Outcome

Kfold
K folds
N epochs

Overfitting

Plots

Tokenization

- We use tensorflow preprocessing tokenization
- And we got the data like…

```
array([[    0,     0,     0, ...,    21,     7,  2064],
       [    0,     0,     0, ...,    38,     8, 14900],
       [    0,     0,     0, ...,     5,   423,    46],
       ...,
       [    0,     0,     0, ...,    27,     5, 14831],
       [    0,     0,     0, ...,    18,  6805,   492],
       [    0,     0,     0, ...,    38,    12,   134]], dtype=int32)
```

Word Embedding

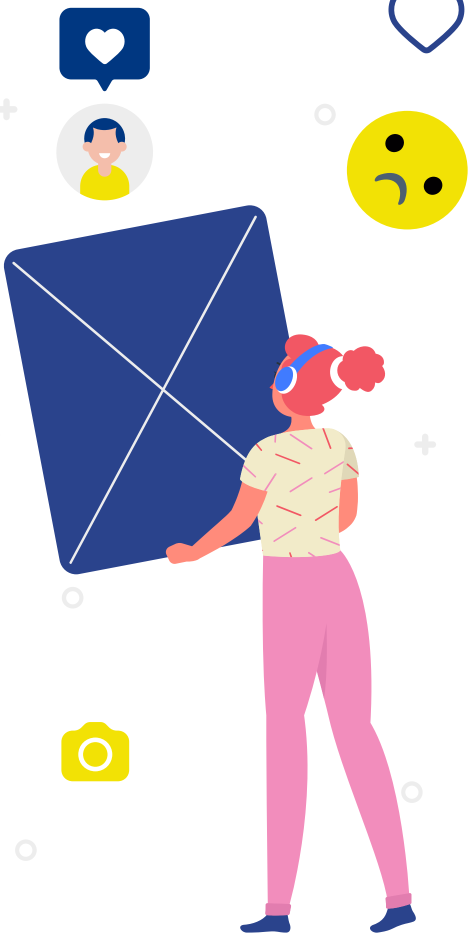
- We use GloVe.6b.100 x 100 pretrained model
- And take “feel” as instance

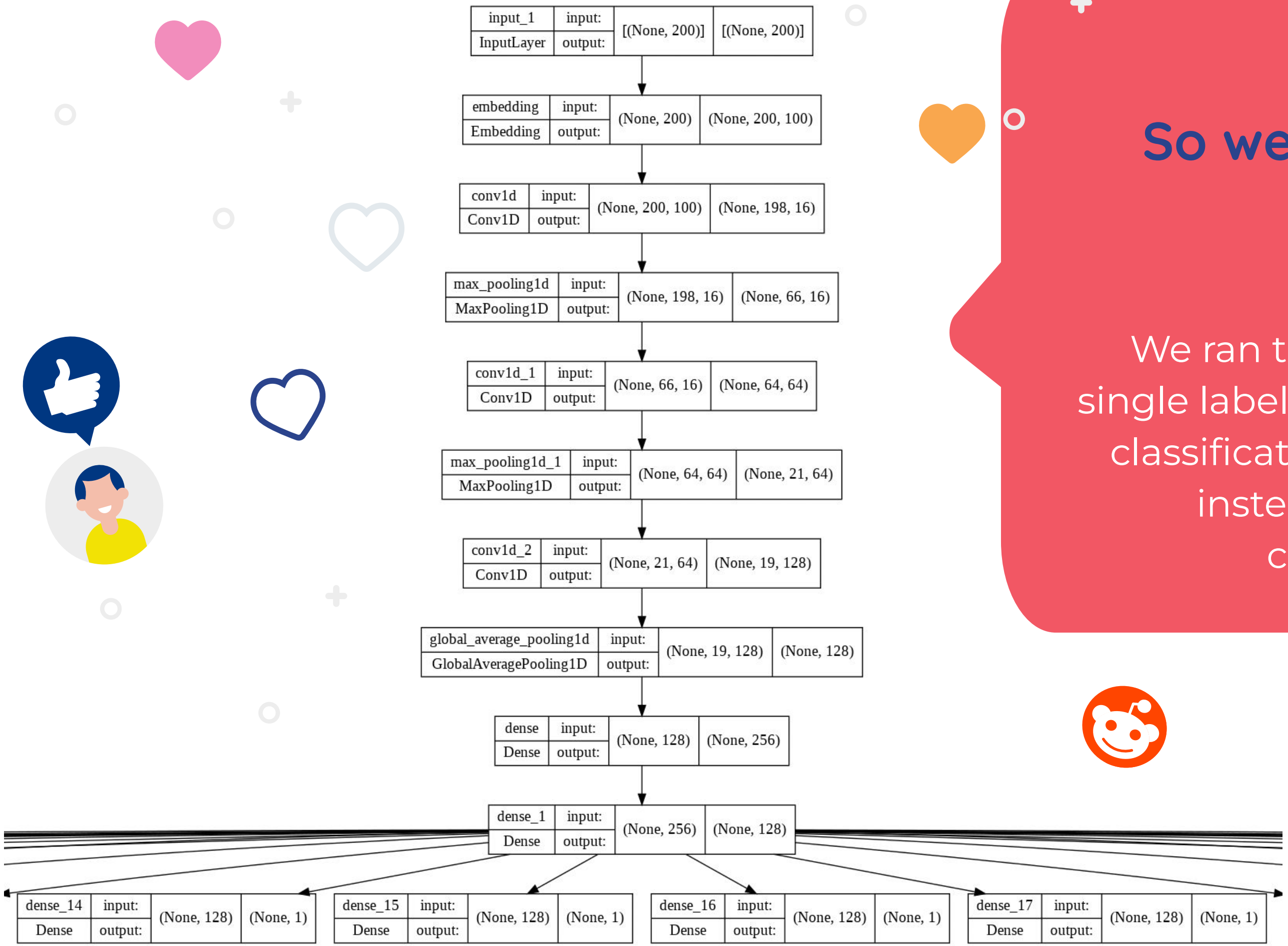
```
array([-4.2895e-02,  7.0671e-01,  7.5316e-01, -5.9963e-01, -8.0169e-01,
        6.0094e-01, -6.7419e-01,  2.3592e-01,  2.4005e-01, -3.6372e-01,
       -2.2720e-01,  3.6026e-02,  4.6591e-01, -3.6233e-01, -3.1325e-01,
       -3.6757e-01, -5.7255e-01,  3.0661e-01, -4.8606e-01,  6.3214e-01,
        2.8931e-01,  6.0741e-01, -8.6788e-03, -6.8206e-01, -1.8410e-01,
        2.4847e-01,  9.5902e-02, -6.9108e-01,  9.0869e-01, -3.9224e-01,
        1.4345e-01,  7.8112e-01, -1.1601e-01,  6.0312e-02,  4.6300e-01,
       -4.4191e-02,  1.4284e-01,  6.2234e-01,  2.3943e-01, -4.6815e-01,
       -2.2553e-01, -3.1920e-01,  3.1397e-01, -4.2308e-01, -4.0827e-01,
       -2.4659e-01,  5.1572e-01,  3.5494e-01,  3.1545e-01, -1.4043e+00,
       -1.6486e-01, -5.5508e-02,  1.0260e-01,  6.1515e-01,  1.2691e-01,
       -2.2379e+00,  6.9510e-01,  1.2609e-01,  1.0901e+00,  2.5953e-01,
        2.2974e-01,  1.1125e+00, -1.0156e+00, -4.1881e-01,  3.8853e-01,
       -4.8856e-02,  9.1775e-01,  1.9732e-03, -6.9162e-01, -5.8945e-01,
        3.5722e-01, -5.2180e-01, -1.2020e-01, -1.3826e-01,  2.1568e-01,
        5.7880e-01,  3.7060e-02, -5.7264e-02, -2.7097e-02, -2.5526e-01,
       -3.9219e-01,  3.7948e-02, -8.5041e-01,  4.6354e-01, -1.6462e+00,
       -2.7176e-01, -1.8171e-01, -3.8026e-02, -9.7623e-01, -5.1404e-01,
        6.9445e-02, -4.6013e-01,  2.5073e-01, -5.0782e-01, -1.5645e-01,
        1.2076e-01, -5.5868e-02, -5.4230e-01,  1.1809e-01,  3.9974e-01],
      dtype=float32)
```

0.3%

Accuracy

With a simple CNN model with 28 possible multiple outputs...





So we have another model...

We ran the model with every single labels, so we will have binary classification for every emotion instead of 28 multiple classifications.

We will have 28 denses in this layer

Performance

With 3 K-folds and 150 epochs, we have all the average accuracy for every emotion in every fold.

admiration	amusement	anger	annoyance	approval	caring	confusion	curiosity	desire	disappointment	disapproval	disgust	embarrassment	excitement
0.814617872	0.904684007	0.882320881	0.78298223	0.712211072	0.913683772	0.884434462	0.87748	0.941569507	0.857094169	0.820344985	0.923229039	0.966046214	0.919956386
0.849117041	0.923569918	0.896843255	0.808072567	0.744733095	0.913956523	0.892070651	0.894525	0.941092253	0.864798546	0.83357197	0.925615311	0.962841749	0.915797353
0.861652792	0.918723583	0.902495563	0.825855732	0.765239358	0.917087138	0.902972877	0.90495	0.948247671	0.879994571	0.854902506	0.923087418	0.965157509	0.921110034
fear	**gratitude**	**grief**	**joy**	**love**	**nervousness**	**optimism**	**pride**	**realization**	**relief**	**remorse**	**sadness**	**surprise**	**neutral**
0.95459193	0.928615272	0.990727484	0.885934412	0.935637832	0.970818818	0.870525658	0.981046	0.849662483	0.978591383	0.974227846	0.902502239	0.927728891	0.623099446
0.954796493	0.934819639	0.989363849	0.893911481	0.946342111	0.967478037	0.889411628	0.97975	0.856821418	0.979750454	0.974296033	0.91647917	0.932501554	0.688961625
0.958270848	0.943065584	0.989976823	0.902904689	0.951247811	0.96904403	0.905018389	0.980567	0.860152721	0.983567417	0.979476333	0.923769236	0.936792552	0.717168987

Also, we have the overall performance...

93.42%
Training

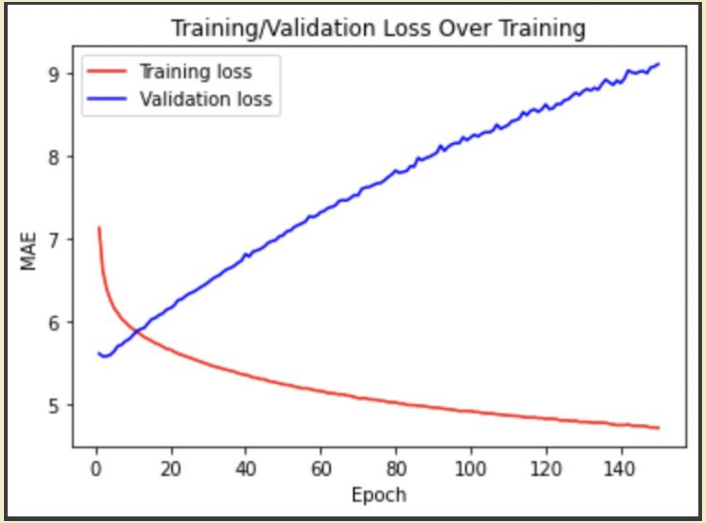
89.59%
Validation

Performance

93.42%

We have a great performance on accuracy!

However...



04

Conclusion

Sum of each label	
neutral	31449
approval	13235
annoyance	10024
admiration	9912
disapproval	8399
realization	7248
disappointment	6656
curiosity	6203
optimism	6200
joy	5688
anger	5644
confusion	5311
gratitude	5298
amusement	5180
sadness	4667
love	4349
excitement	4336
caring	4330
disgust	4053
surprise	3823
desire	2838
example_very_unclear	2731
fear	2136
embarrassment	2003
remorse	1663
nervousness	1556
pride	1127
relief	1085
grief	558

Conclusion

1. Own Accuracy(CNN) > Bert Logistic Regression Accuracy
   93.42% vs 90.45%
2. Overfitting Problem
3. Imbalanced Issue

Thanks

Credits: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik and illustrations by Storyset