Team 7

Restaurant Hunter

109034023 林哲宇
109034032 林宣霆
109034063 林東擇

OUTLINE


1
Problem Identification & Target users
System framework
2
3
Details in our model pipeline
Chatbot
4
5
The result

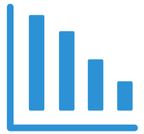
Problem identification

Server

How do users find the most representative strengths and weaknesses of a restaurant from the reviews on google map

Comment analysis

When the number of comments is very large, it becomes difficult to find the disadvantages that have the most guest responses



Visualization

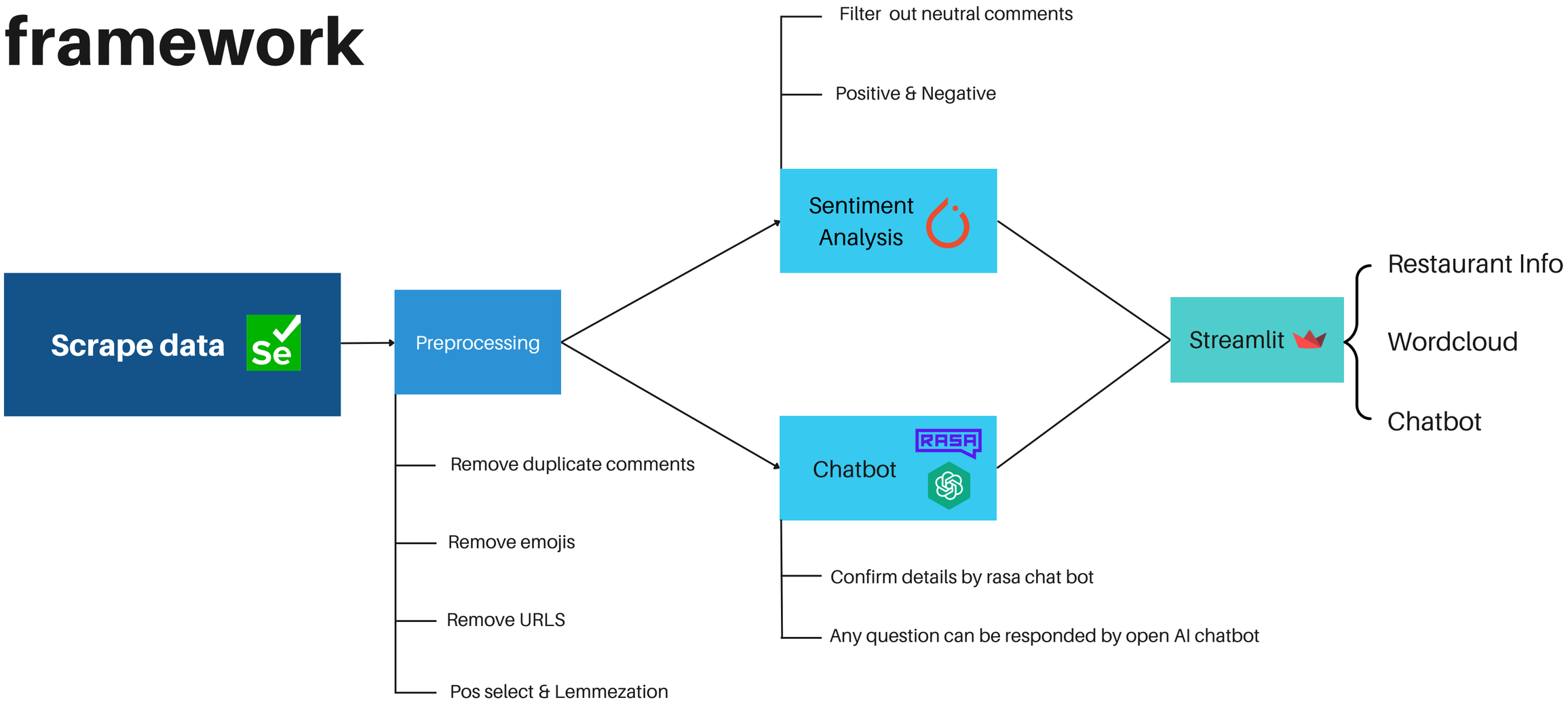
Our app will speed up the process and visualize the results

Target users

Restaurants who want to quickly identify strengths and weaknesses and make improvements

Consumers who want to find specific advantages or avoid specific disadvantages

System framework



Scrape data

Search restaurant
in
google map

Scraped comments,
address, location, phone
number

```
def WebScrape(restaurant_name, comments_num, progress_bar):
    chrome_options = Options()
    chrome_options.add_argument("--lang=en-US")
    chrome_options.add_argument('--headless')

    driver = webdriver.Chrome("桌面\chromedriver.exe",
options=chrome_options)

    search_url =
"https://www.google.com/maps/@39.1111036,-105.0715591,5.37z?hl=en"
    driver.get(search_url)
    time.sleep(2)

    # Search by restaurant name
    Search_box = WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.ID, "searchboxinput")))
    Search_box.clear()
    Search_box.send_keys(restaurant_name)
    Search_box.send_keys(Keys.ENTER)
    time.sleep(2)
    .
    .
    .
```

Preprocssing

--- Remove duplicate comments

Scraped comments

↓ Remove duplicate reviews

Scraped comments (Clean)

```
# Remove duplicate reviews
unique_comment_set = set(google_comment_scrape['comment'])
unique_comments = list(unique_comment_set)

google_comment_df =
pd.DataFrame({"comment":unique_comments})
google_comment_df['grade'] = 0
google_comment_df
```

Preprocssing

--- Remove emojis

Scraped comments

Scraped comments without emojis

```python
# Remove Emojis
def remove_emojis(text):
    # Remove emojis using regular expressions
    emoji_pattern = re.compile("["
                    u"\U0001F600-\U0001F64F"  # emoticons
                    u"\U0001F300-\U0001F5FF"
                    u"\U0001F680-\U0001F6FF"
                    u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                    u"\U00002500-\U00002BEF"  # chinese char
                    u"\U00002702-\U000027B0"
                    u"\U00002702-\U000027B0"
                    u"\U000024C2-\U0001F251"
                    u"\U0001f926-\U0001f937"
                    u"\U00010000-\U0010ffff"
                    u"\u2640-\u2642"
                    u"\u2600-\u2B55"
                    u"\u200d"
                    u"\u23cf"
                    u"\u23e9"
                    u"\u231a"
                    "\ufe0f"  # dingbats
                    "\u3030"
                    "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
```

Preprocssing

--- Remove URLS

Scraped comments
without
emojis

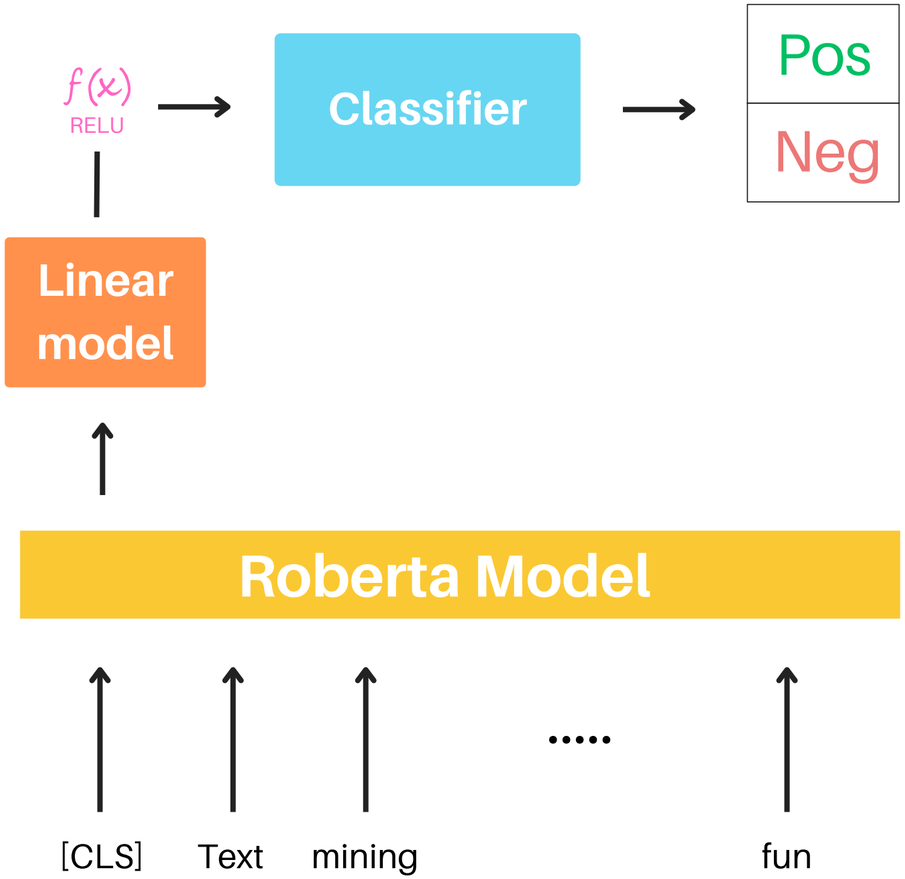
~~https://...~~

Scraped comments
without
URLS

```
# Remove URL
def remove_urls(text):
    url_pattern =
re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)
```

Sentiment analysis

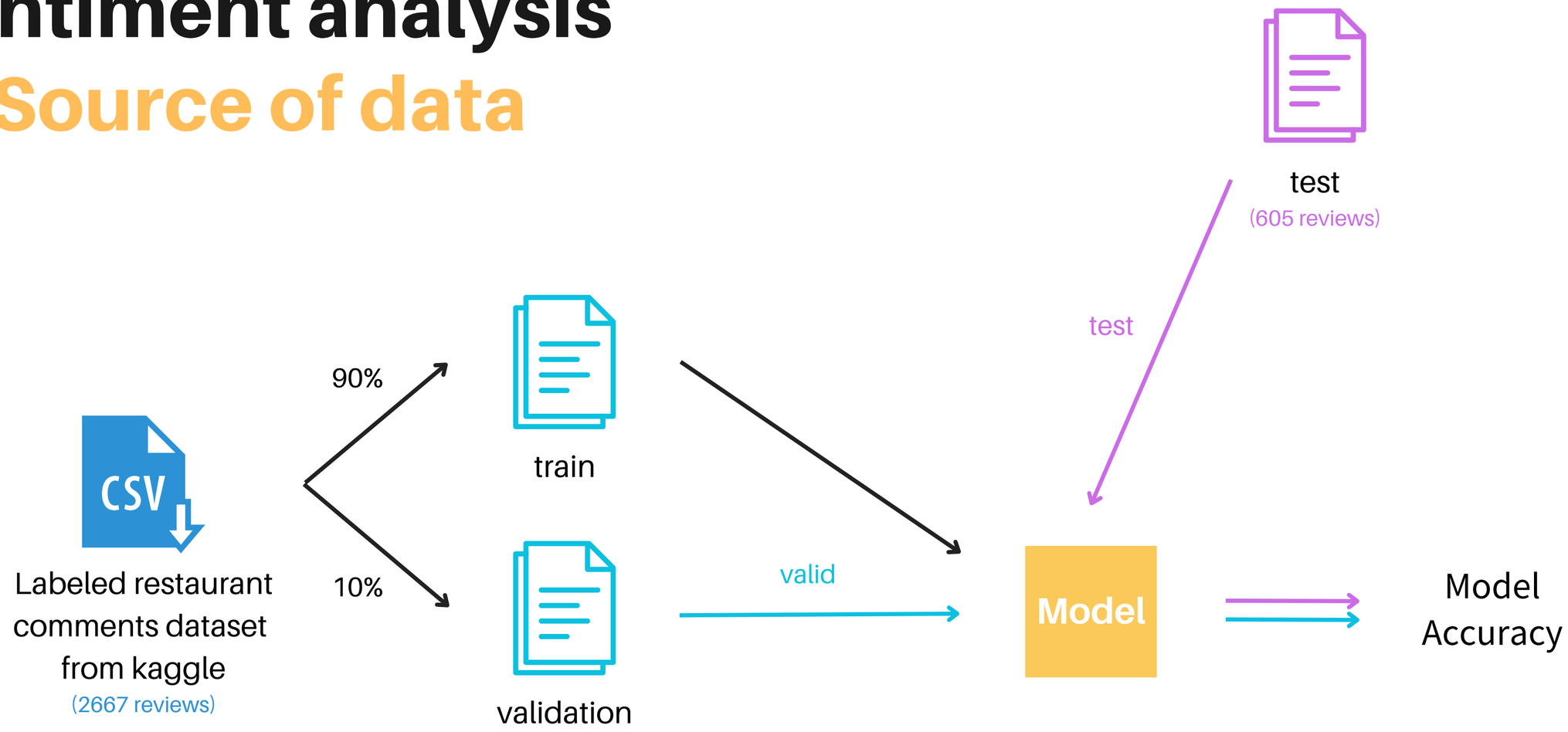
--- Model




```python
import torch

class Roberta_Model(nn.Module):
    def __init__(self):
        super(Roberta_Model, self).__init__()
        self.roberta_model = RobertaModel.from_pretrained("roberta-base")
        self.pre_classifier = nn.Linear(768, 50)
        self.dropout = nn.Dropout(0.3)
        self.classifier = nn.Linear(50, 2) # Output positive, negative

    def forward(self, input_ids, attention_mask, token_type_ids):
        raw_output = self.roberta_model(input_ids=input_ids, attention_mask=attention_mask, token_type_ids=token_type_ids)
        pooler = raw_output["pooler_output"]
        pooler = self.pre_classifier(pooler)
        pooler = nn.ReLU()(pooler)
        pooler = self.dropout(pooler)
        output = self.classifier(pooler)
        return output
```

Sentiment analysis

--- Source of data

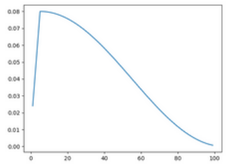


(Source of data: https://drive.google.com/file/d/1_mYqaco9oUrUQJmVXdFzpDwV1v6VmEkd/view?usp=sharing)

Sentiment analysis
--- Fine tune model

Optimizer - AdamW

Learning rate - Warmup



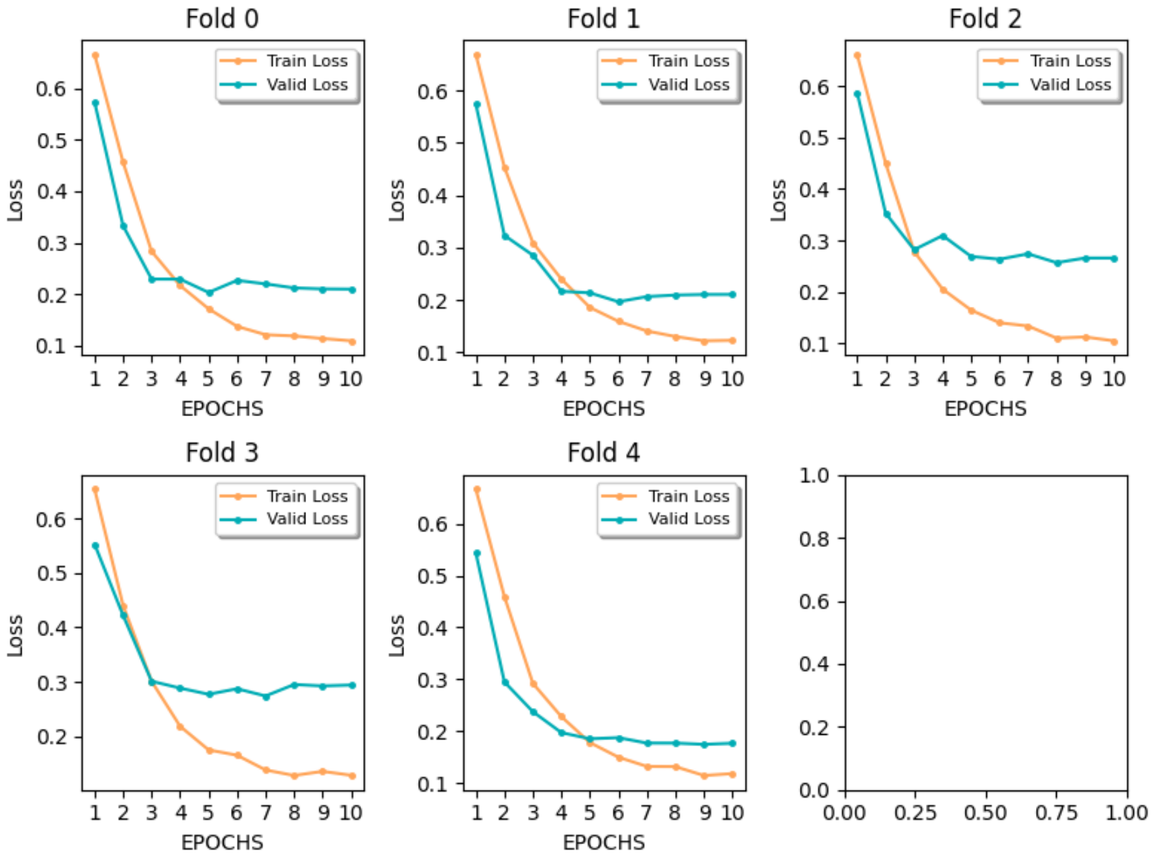
Loss function - CrossEntropy

```python
def train_fn(step, data_loader, model, optimizer, device, scheduler):
    model.train() # Switch the model to train mode

    logging_step = 5
    train_losses = []

    for batch in data_loader:
        ids = batch["ids"].to(device, dtype=torch.long)
        masks = batch["masks"].to(device, dtype=torch.long)
        token_type_ids = batch['token_type_ids'].to(device, dtype = torch.long)
        targets = batch["targets"].to(device, dtype=torch.long)

        optimizer.zero_grad() # Set gradient to zero

        outputs = model(ids, masks, token_type_ids).squeeze(-1) # Predictions from 1 batch of data

        loss = loss_fn(outputs, targets)
        train_losses.append(loss.item())

        loss.backward() # Compute gradient(backpropagation)
        optimizer.step() # Update parameters
        scheduler.step() # Update learning rate

        step += 1
        if step % logging_step == 0:
            print(f"Step {step} | loss = {loss}")

    return train_losses
```

Sentiment analysis

--- Model performance

In each fold of our validation, loss has been trending down



Sentiment analysis

--- Model performance

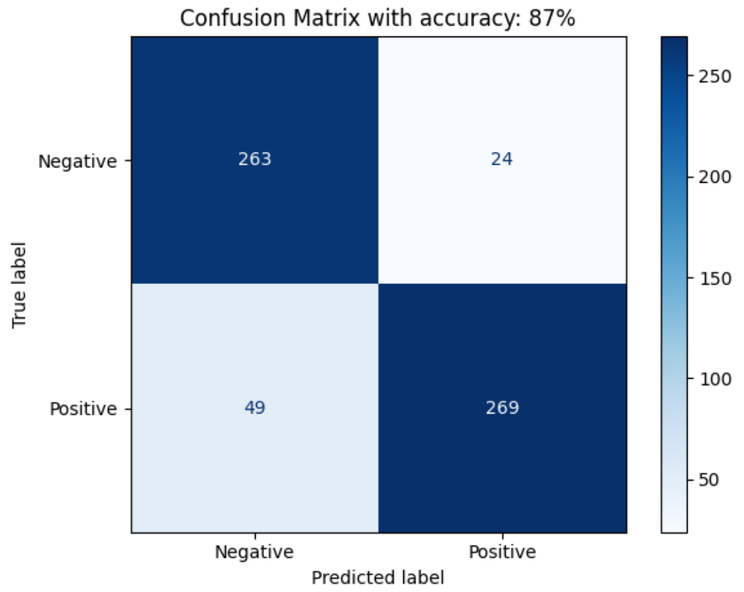
😭 Mid-Project

- Valid acc 78%
- test acc 72%

🤧 Final-Project

- Valid acc 93%
- test acc 87%

```
Validation Accuracy Epoch: 93.43339587242026
```



Sentiment analysis
--- Filter neutral comments

Positive & neutral & negative comments

lemmatization & pos select

vader algorithm

Positive & negative comments

```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer

sia = SentimentIntensityAnalyzer()

score_dic = {}
neg_score_list = []
neu_score_list = []
pos_score_list = []

for comment in google_negative_comment['comment']:
    score_dic = sia.polarity_scores(comment)
    neg_score_list.append(score_dic['compound'])

for comment in google_positive_comment['comment']:
    score_dic = sia.polarity_scores(comment)
    pos_score_list.append(score_dic['compound'])

google_negative_comment['compound'] = neg_score_list
google_positive_comment['compound'] = pos_score_list
.
.
.
```

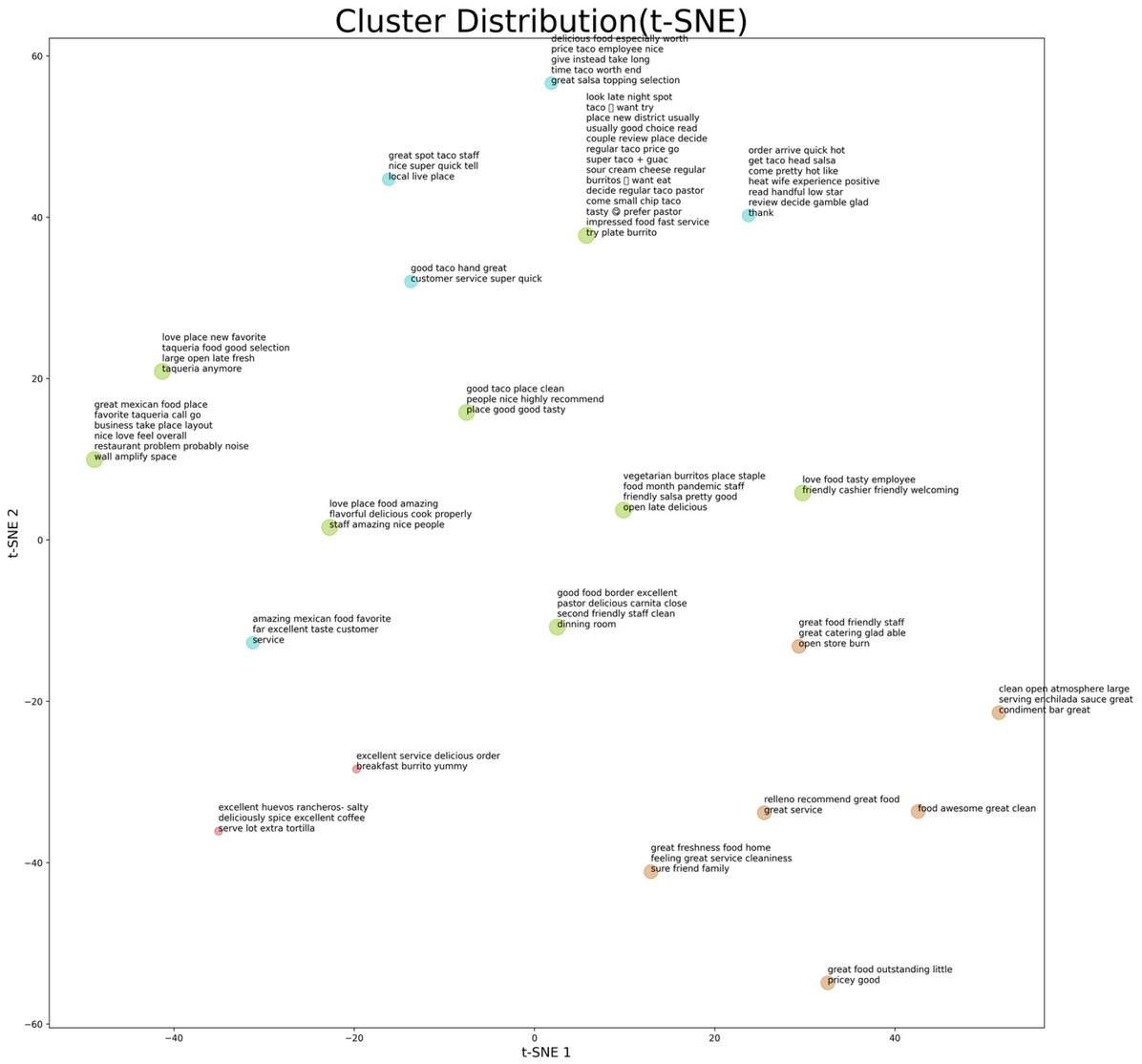
Sentiment analysis
--- Visualization

Positive & negative comments

↓

Caculate distance between comments (tfidf)

Visualize Positive & negative comments by T-SNE

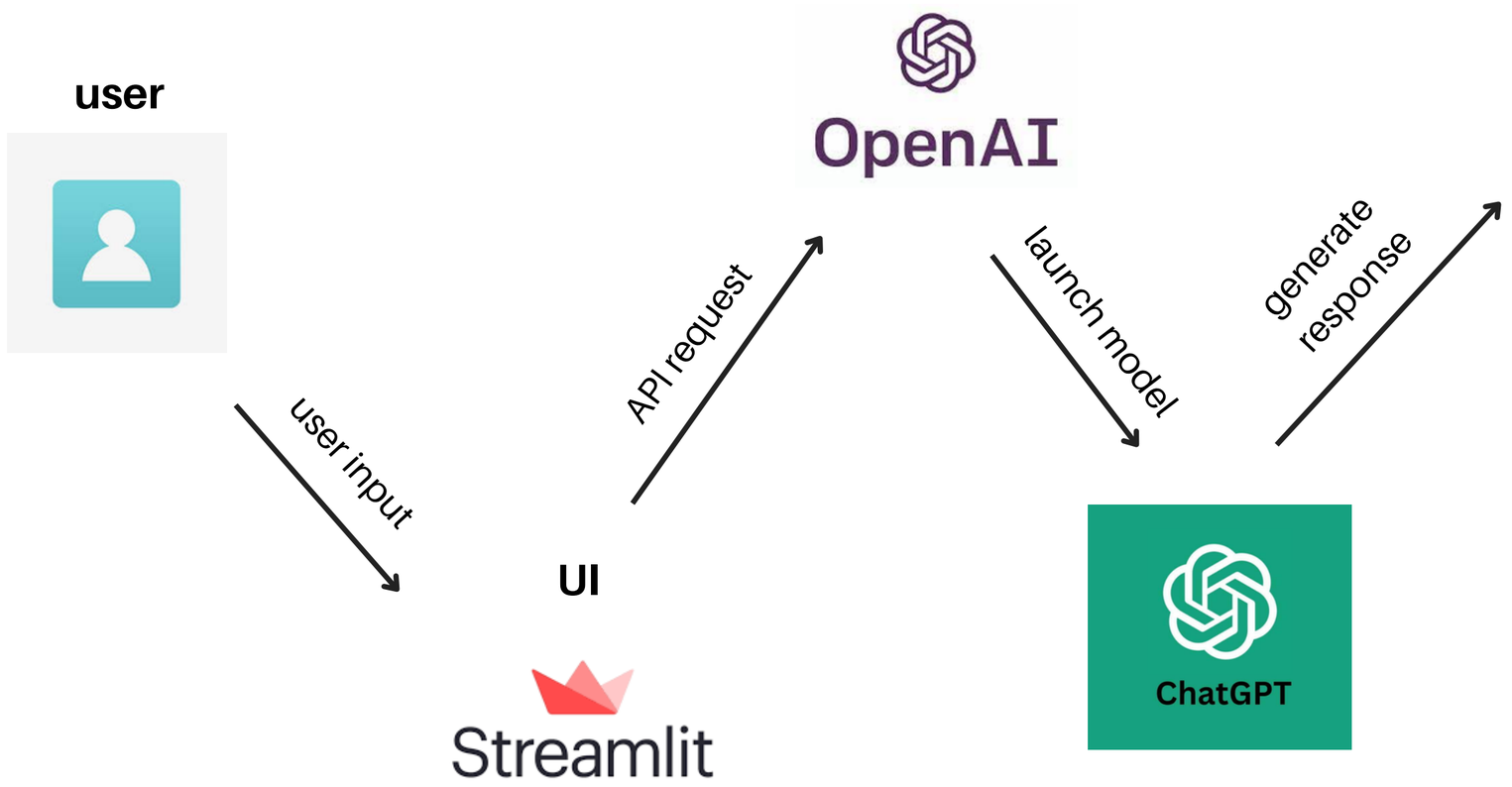


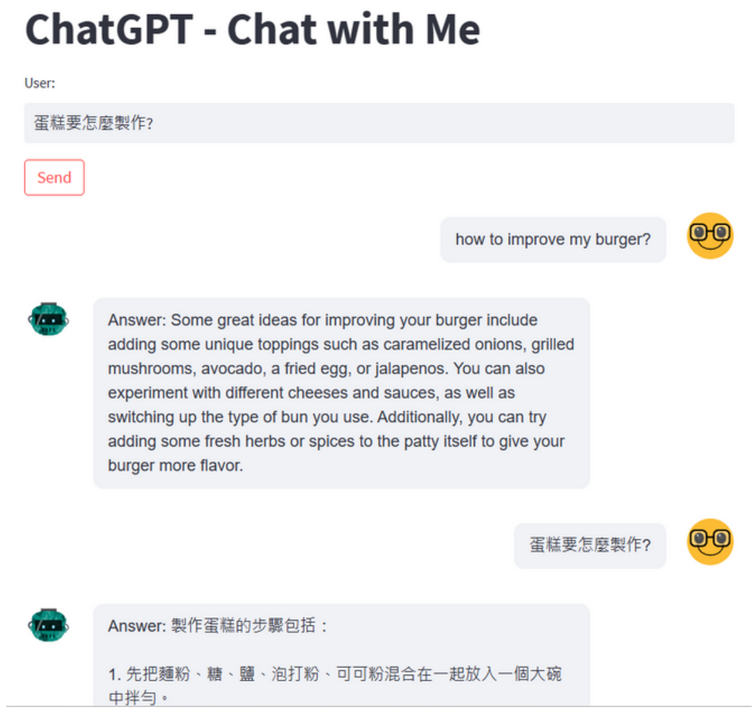
Sentiment analysis
--- Visualization

Easy visualize what food or service need to improve

Chatbot

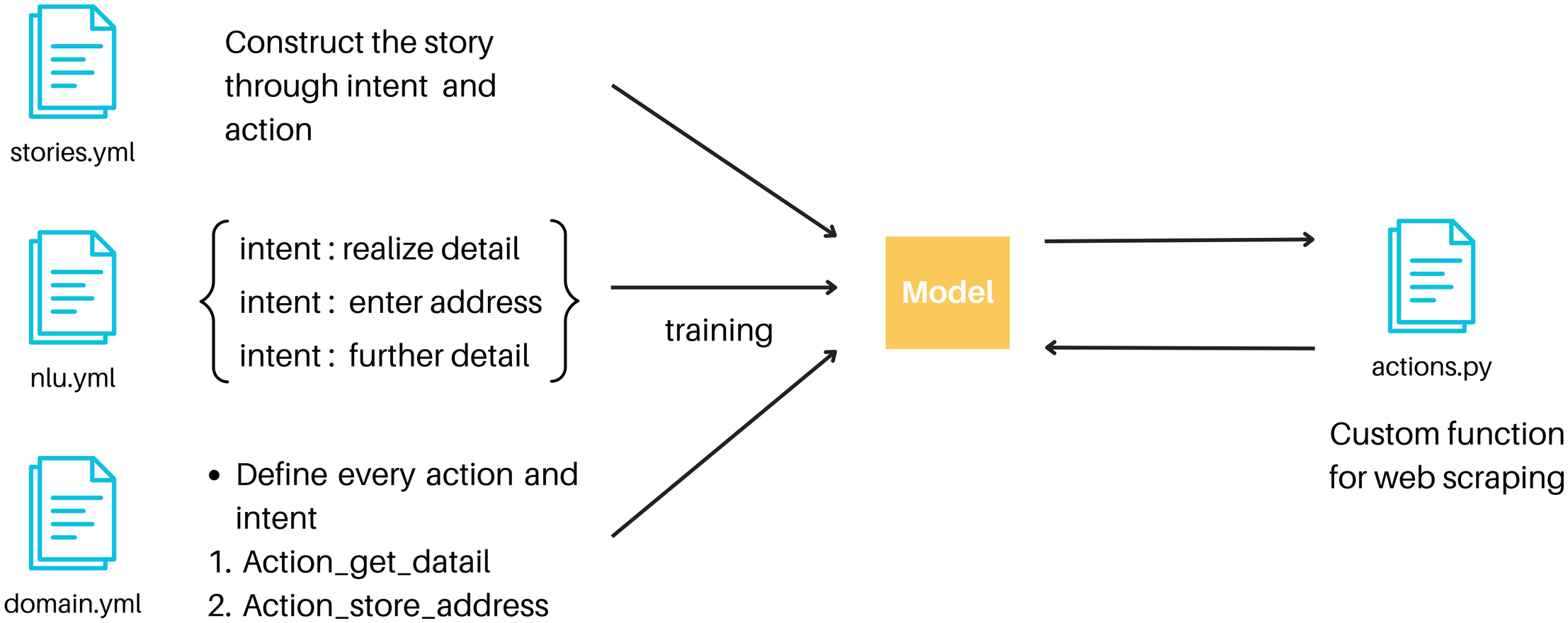
--- OpenAI





Chatbot

--- Rasa



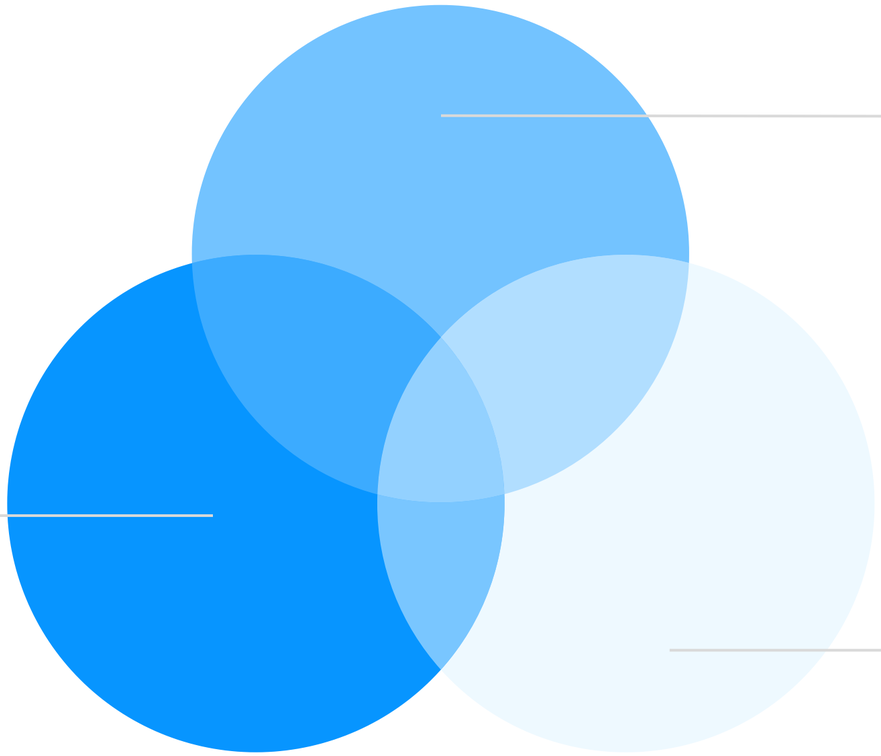
Results
--- Benifits

Customer

Customers can know the biggest advantages and disadvantages of the restaurant in advance before actually going to the restaurant.

Use Chatbot

If there is a need to further realize what customers' feeling, they can use rasa or openAI chatbot to check the comments with specific keyword.



Restaurant

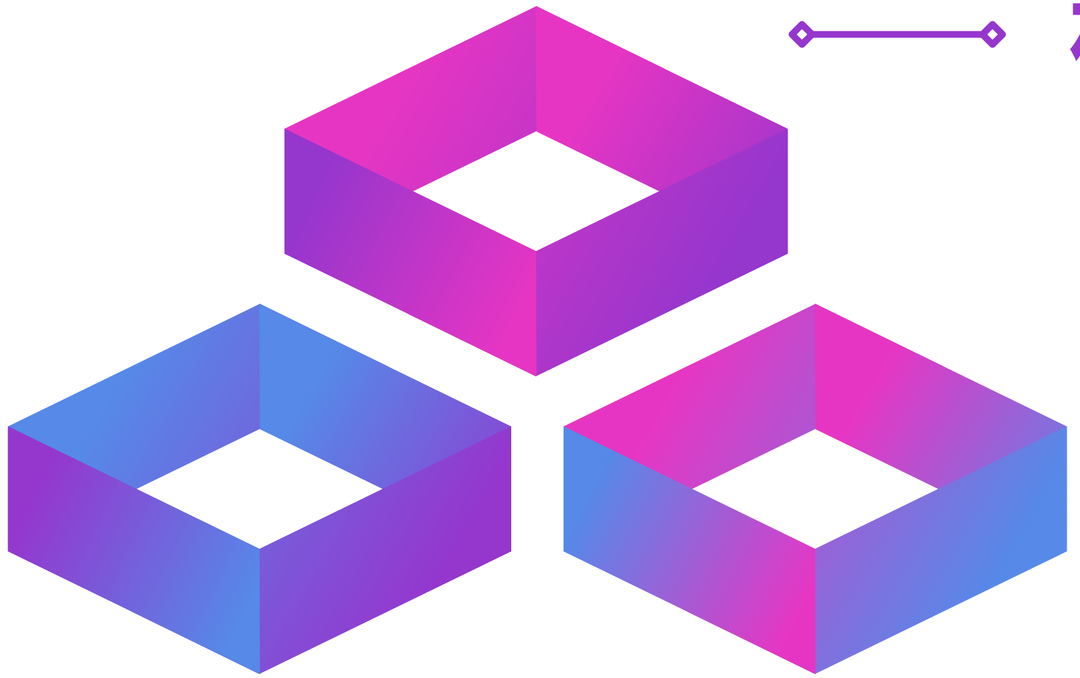
Restaurants can discover their biggest strengths and weaknesses through this app.

Results

--- Weakness

The execution speed is too slow, maybe we can change the code to make the execution speed faster and make the user experience better.

Division of work



林哲宇

- User interface construction
- Roberta fine tuning
- Web scraping (first page of user interface)

林宣霆

- RASA chatbot
- Second page of interface
- Final project video

林東擇

- OpenAI chatbot
- Second page of interface

Thank you for listening!