GOOGLE PLAY REVIEWS SENTIMENT ANALYSIS

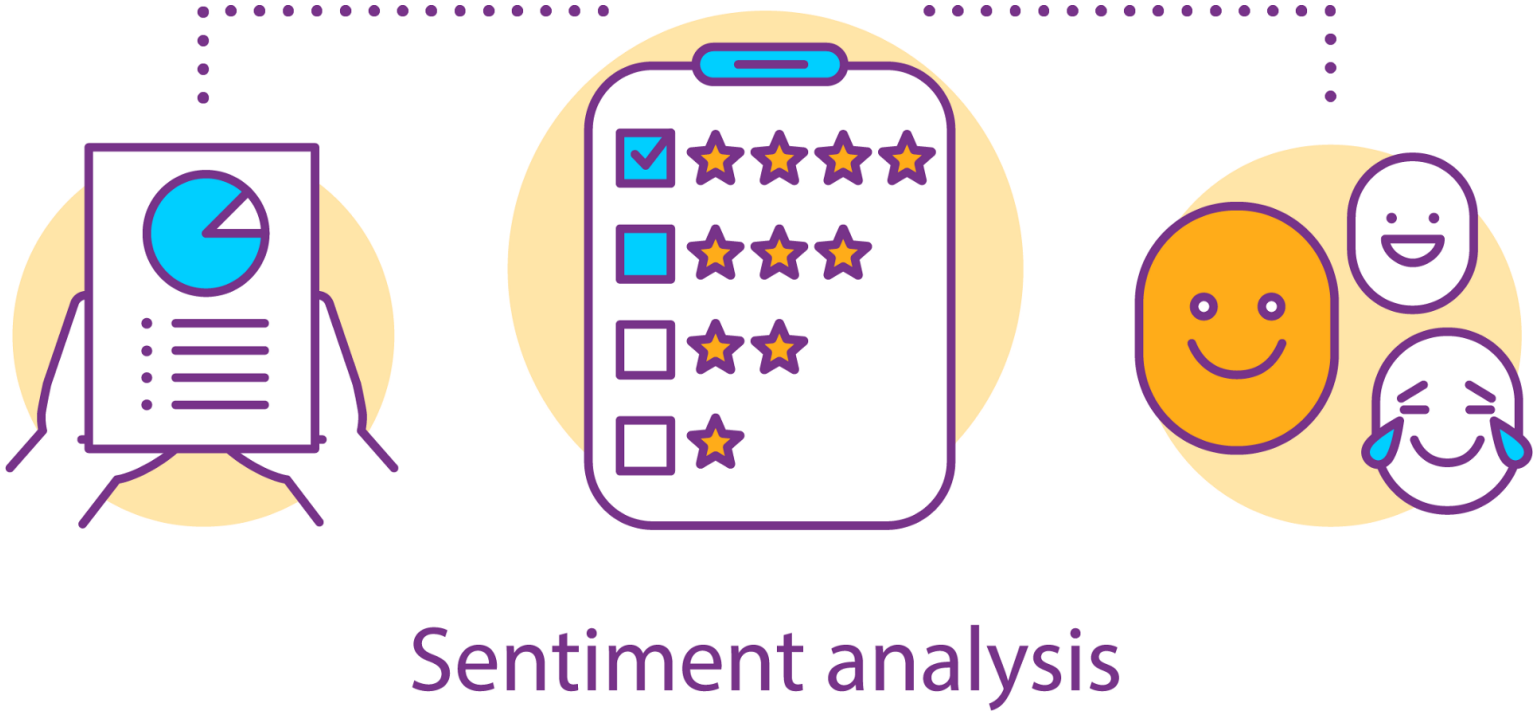
Agenda

- INTRODUCTION
- PIPELINE
- DATA COLLECTION
- DATA ANALYSIS
- DATA PREPROCESSING
- TEXT REPRESENTATION
- MODELS
- API
- DEPLOYMENT
- FUTURE WORK
- RESOURCES

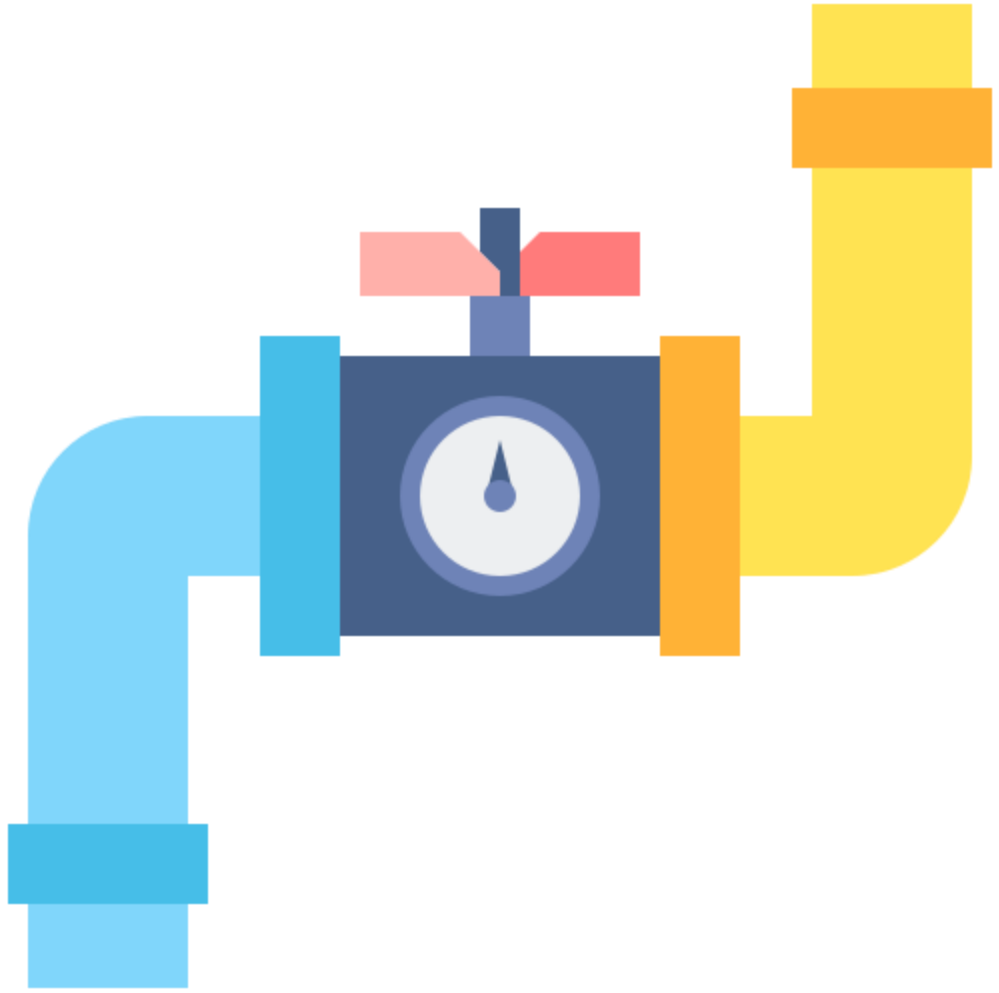
Introduction

Introduction

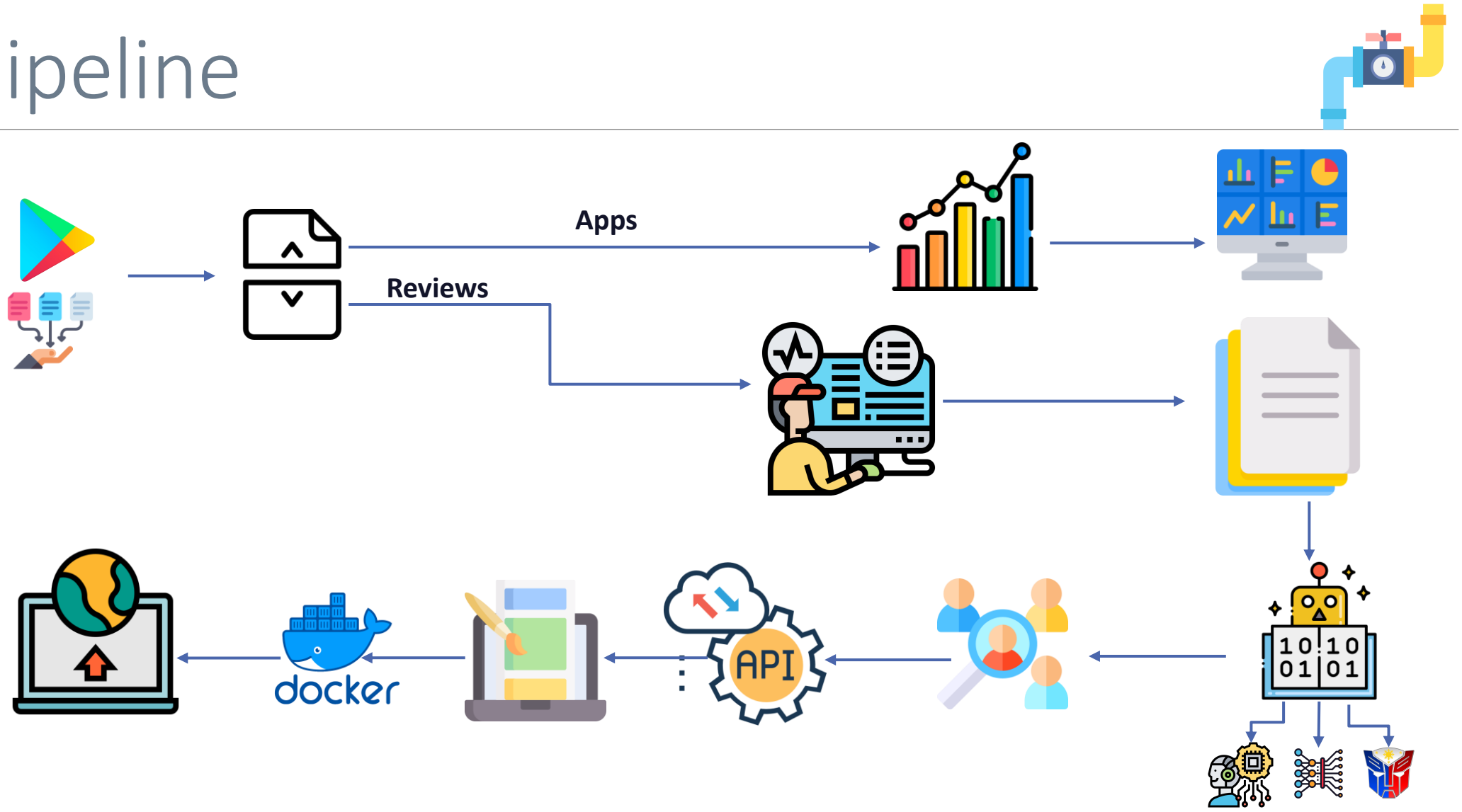
- What is Sentiment Analysis?
- Business Domain



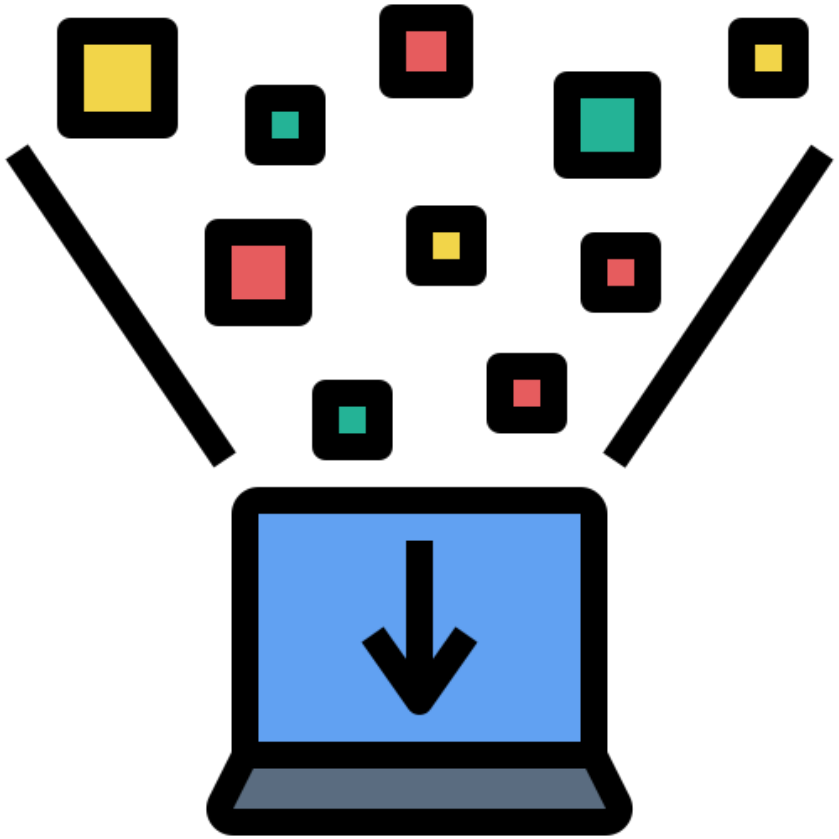
Pipeline



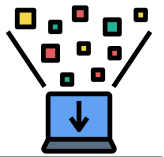
Pipeline



Data Collection

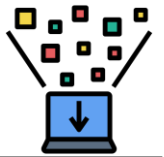


Data Collection

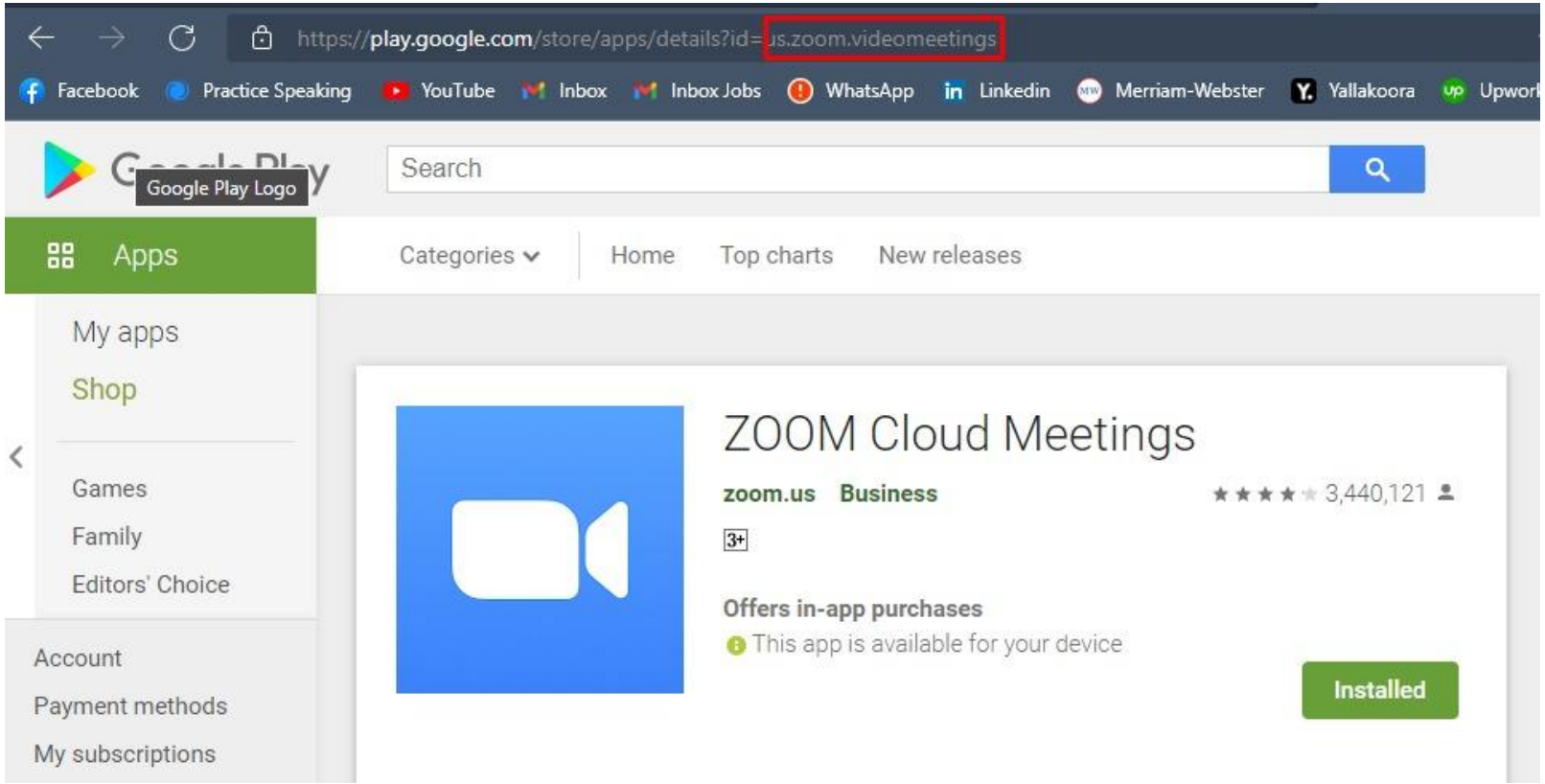


1. Scrap Tool Definition:

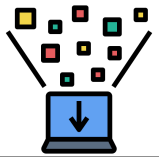
Data Collection



2. Scraping Input:



Data Collection



3. Scraping Output


```
{
    "title": "Pokémon GO",
    "appId": "com.nianticlabs.pokemongo",
    "url": "https://play.google.com/store/apps/details?id=com.nianticlabs.p
    "description": "NEW! It’s time to interact with your Pokémon like never
    "descriptionHTML": "NEW! It’s time to interact with your Pokémon like n
    "summary": "Step outside and catch Pokémon in the real world! Collect &
    "summaryHTML": "Step outside and catch Pokémon in the real world! Colle
    "installs": "100,000,000+",
    "minInstalls": 100000000,
    "score": 4.126332,
    "ratings": 12001843,
    "reviews": 4772986,
    "originalPrice": None,
    "sale": False,
    "saleText": None,
    "saleTime": None,
```

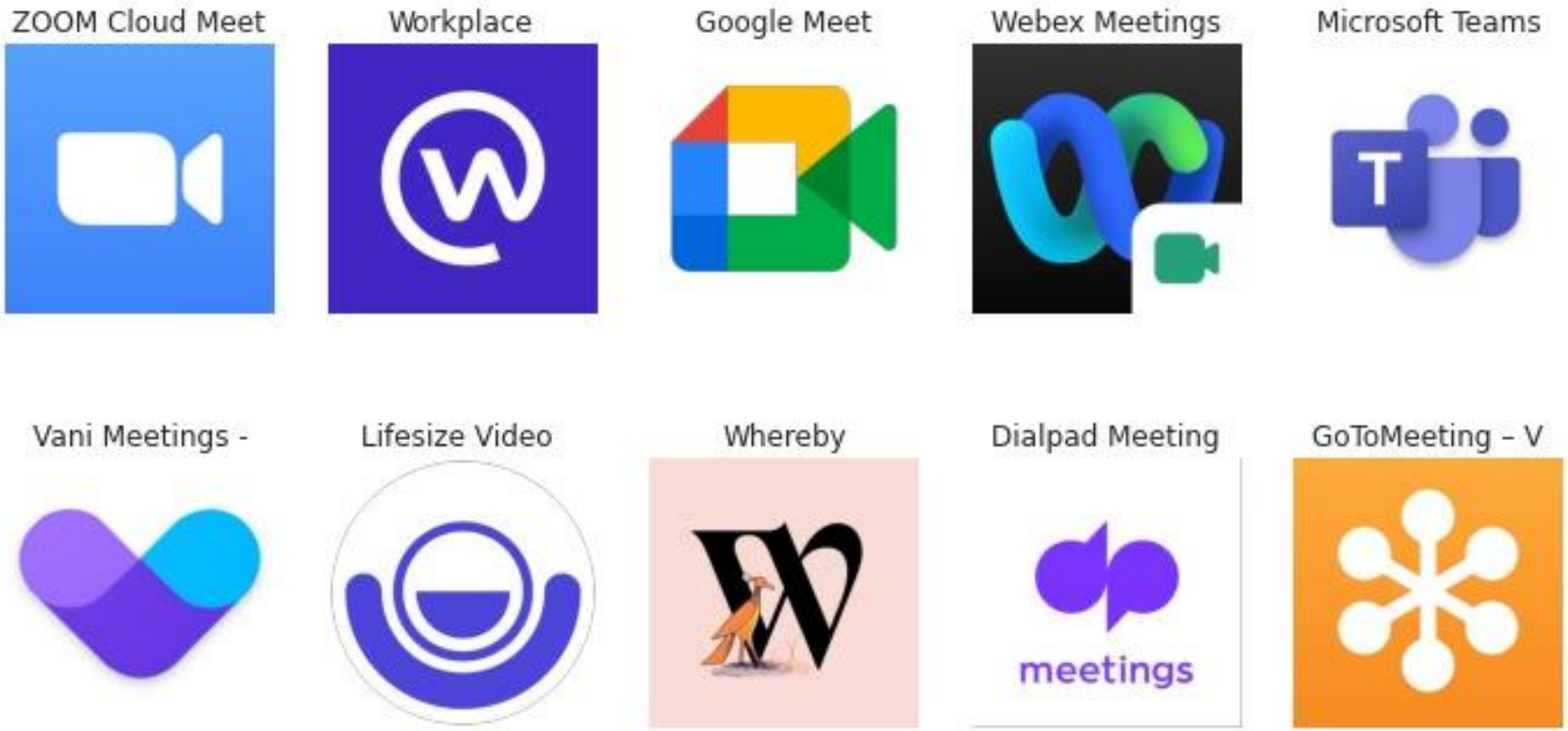

Apps


```
[
    {
        "userName": "Alyssa Williams",
        "userImage": "https://lh3.googleusercontent.com/-cVEHKr7mzv8/AAAAAA
        "content": "This is literally the best idle game I have ever played
        "score": 5,
        "thumbsUpCount": 54,
        "reviewCreatedVersion": "1.16",
        "at": datetime.datetime(2020, 2, 24, 17, 19, 34),
        "replyContent": "Hello, We will gradually improve the various syste
        "repliedAt": datetime.datetime(2020, 2, 24, 18, 30, 42),
        "reviewId": "gp:AOqpTOE0Iy5S9Je1F8W1BgCl6l_TCFP_QN4qGtRATX3PeB5VV9a
```


Reviews

Data Collection

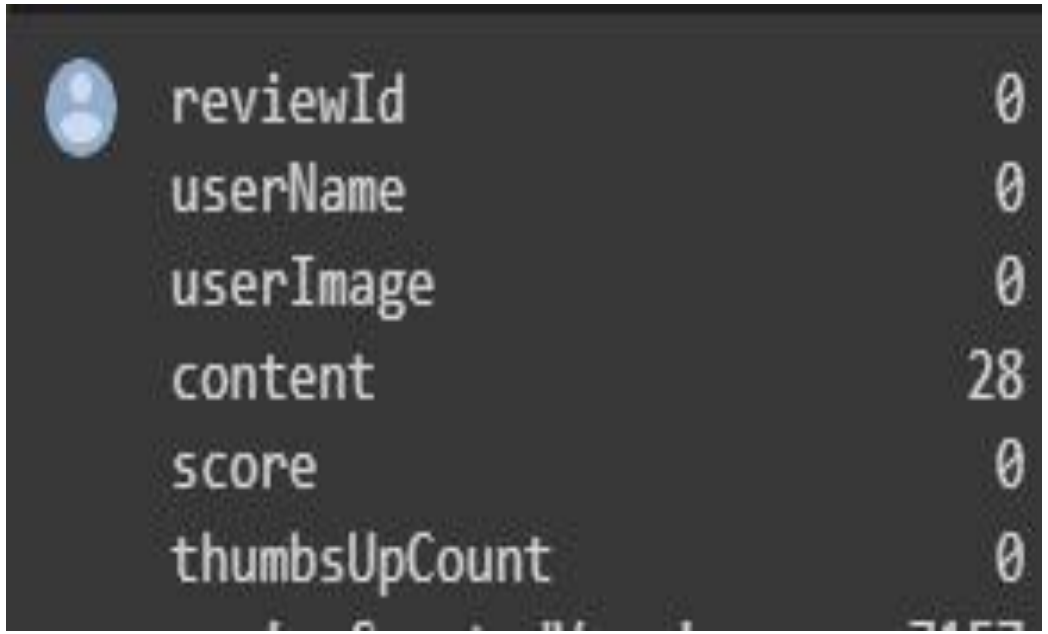
4. Applications We Scraped:



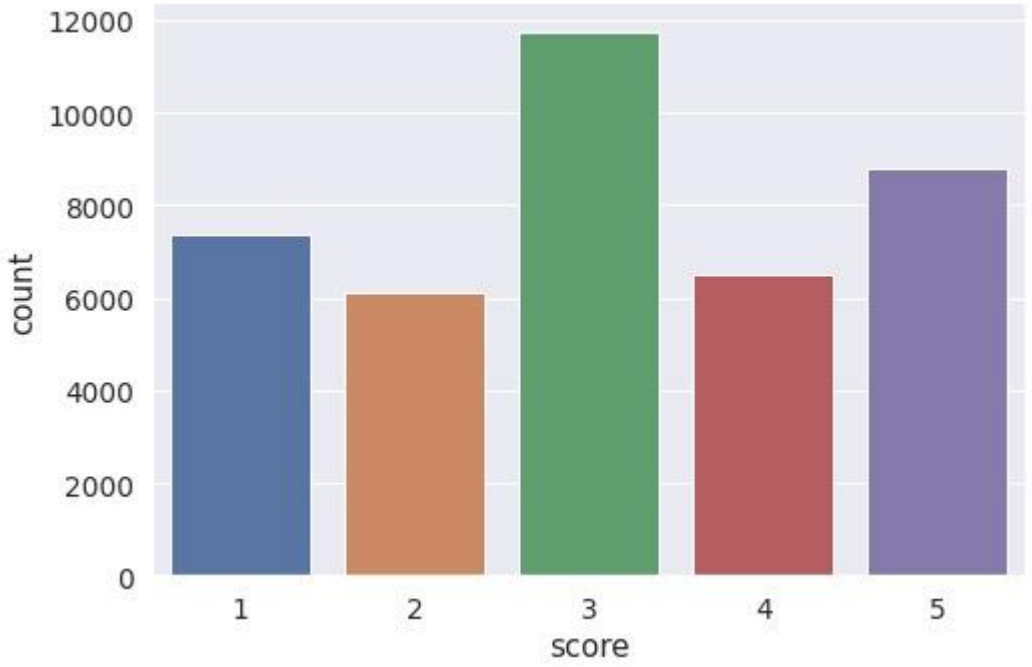
Data Analysis

Data Analysis

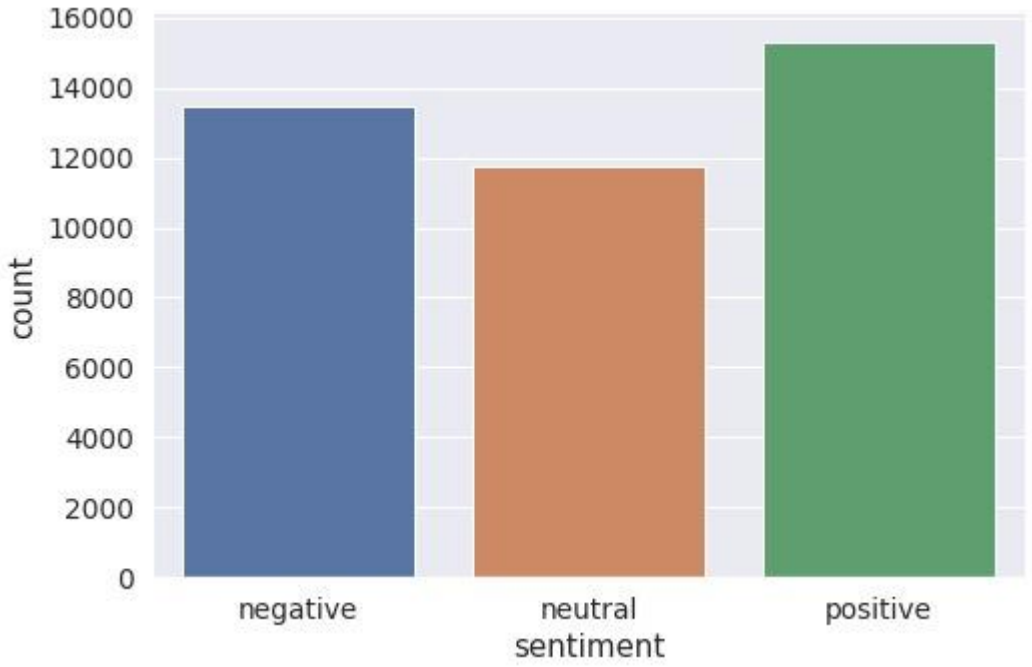
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40546 entries, 0 to 40545
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   reviewId              40546 non-null  object
 1   userName              40546 non-null  object
 2   userImage             40546 non-null  object
 3   content               40518 non-null  object
 4   score                 40546 non-null  int64
 5   thumbsUpCount         40546 non-null  int64
 6   reviewCreatedVersion  33389 non-null  object
 7   at                    40546 non-null  object
 8   replyContent          7943 non-null   object
 9   repliedAt             7943 non-null   object
 10  sortOrder             40546 non-null  object
 11  appId                 40546 non-null  object
dtypes: int64(2), object(10)
memory usage: 3.7+ MB
```



Data Analysis



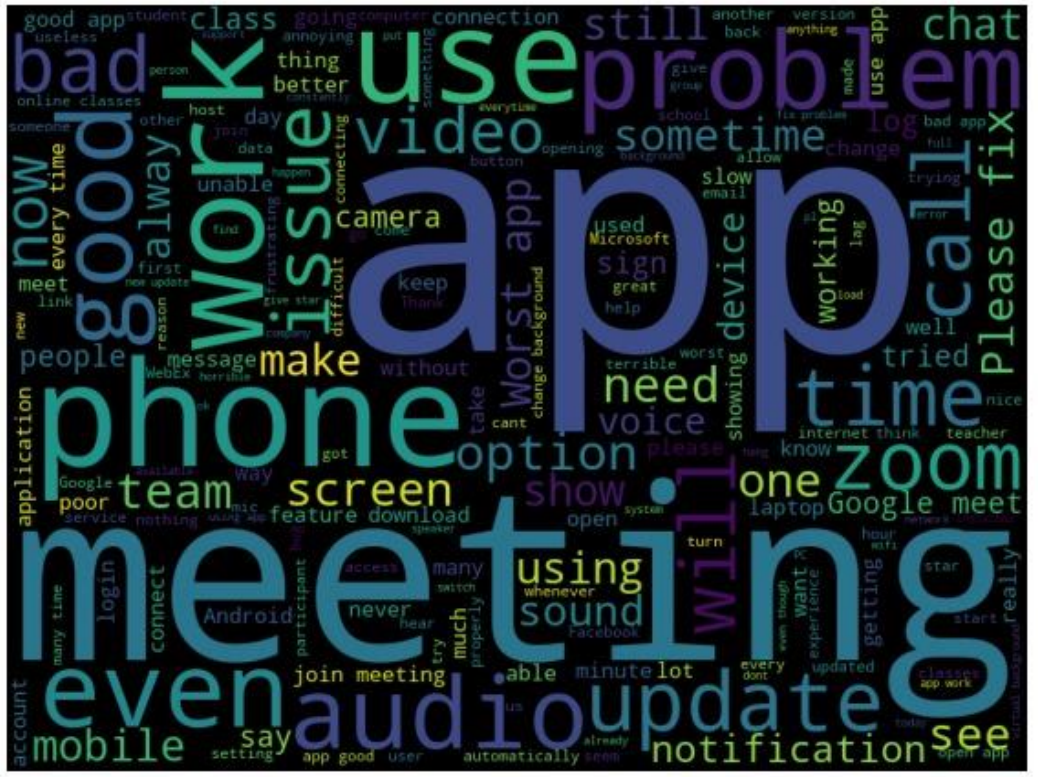
Before



After

Data Analysis

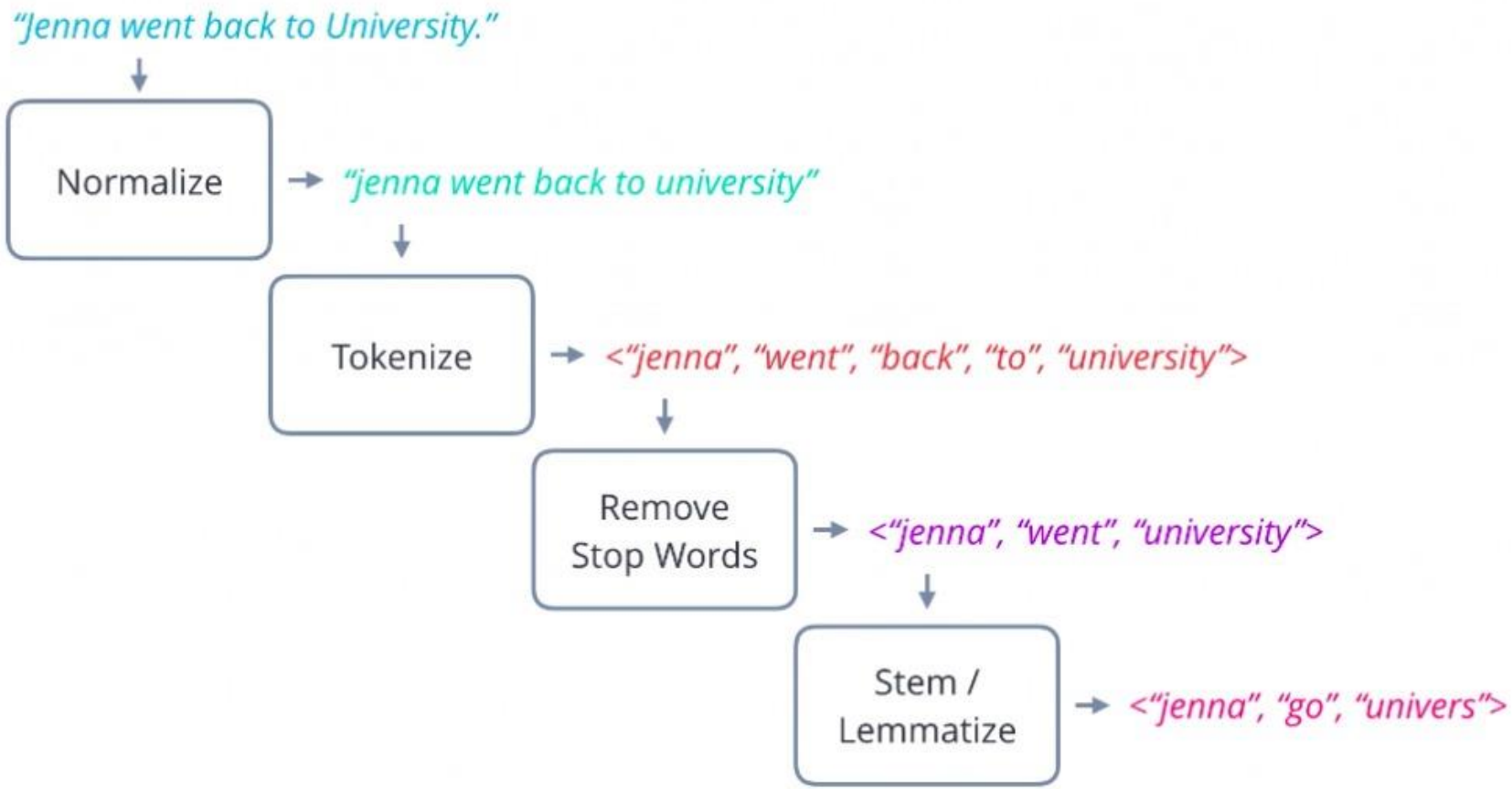
Positive



Negative

Data Preprocessing

Data Preprocessing

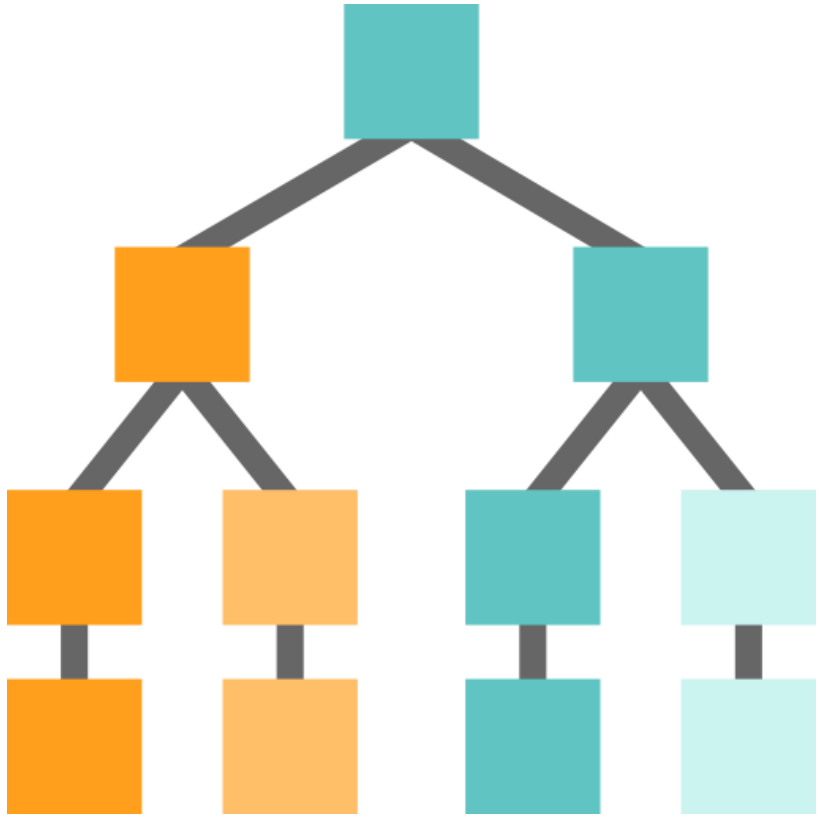


Data Preprocessing

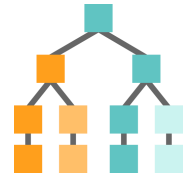
```
1 print(f"Orignal Text : {data.review[13]}")
2 print()
3 print(f"Preprocessed Text : {preprocess_text(data.review[13])}")
```

```
Orignal Text : I like the mobility the app provides. however, it does tend to be a bit glitchy

Preprocessed Text : ['i', 'like', 'mobil', 'app', 'provid', 'howev', 'tend', 'bit', 'glitchi']
```

Text Representation



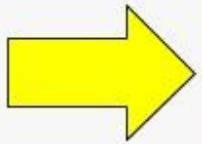
Text Representation



Count Vectorizer:

- Count the occurrences of each word in the corpus
- Doesn't care about order
- Easy to generate and understand

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Text Representation

TF-IDF:

- Focus on unique words in the corpus
- Give words/n-gram a weight based on it's frequency
- very useful in many domains

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

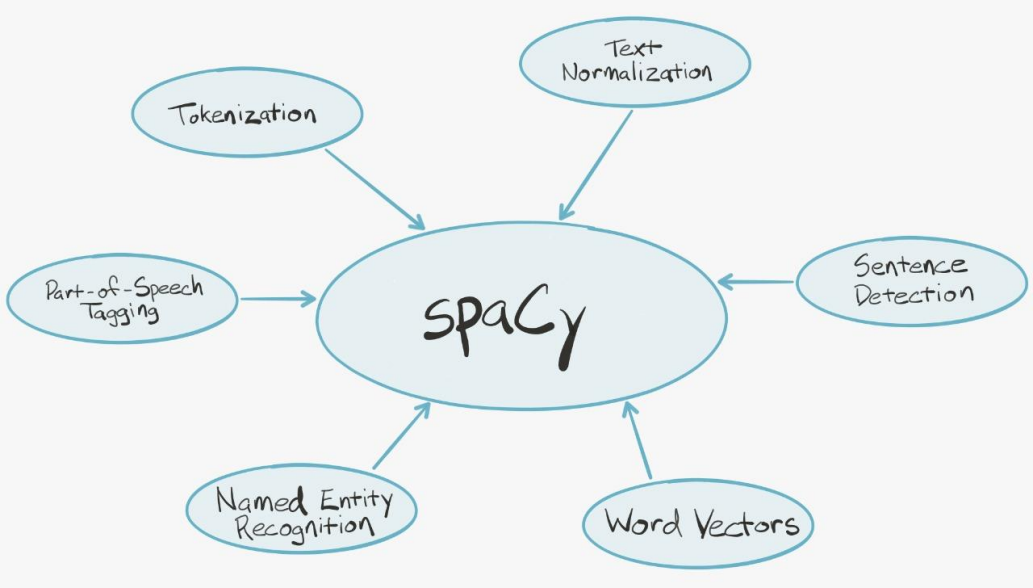
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

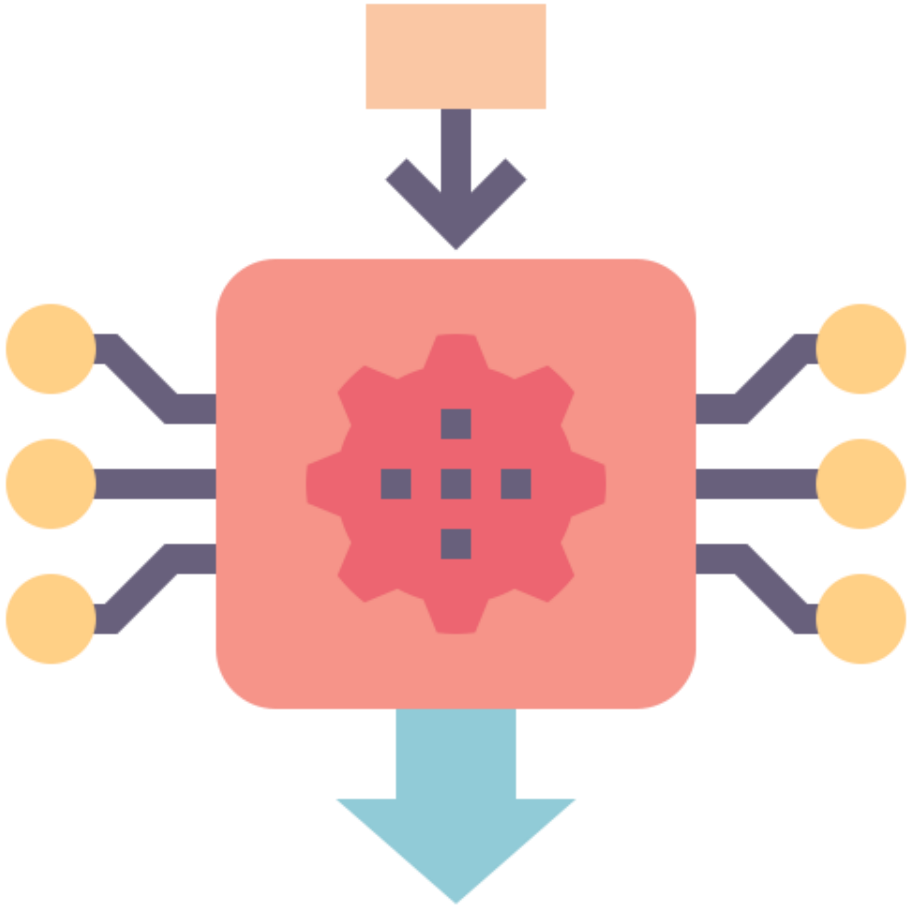
Text Representation

Word2vec(pretrained) :

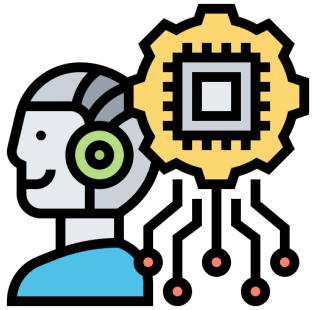



```
>>> import spacy
>>> nlp = spacy.load("en_core_web_lg")
doc >>> text = "the fat sister was running towards the plate."
>>> doc = nlp(text)
>>> doc.vector
array([ 2.57911207e-03,  4.20577750e-02, -1.24354653e-01, -2.02225317e-04,
        8.28643292e-02,  1.30202264e-01, -3.06084417e-02, -8.31593126e-02,
        1.11833781e-01,  2.22333312e+00, -1.55609339e-01,  2.40942210e-01,
        1.28558666e-01, -1.36541585e-02, -8.07588920e-02, -5.62237687e-02,
       -8.09930265e-02,  1.19106889e+00, -1.87391460e-01, -1.03011444e-01,
       -8.48544668e-03, -1.92224622e-01, -5.69729991e-02,  5.63547760e-02,
        3.60591151e-02, -1.56987756e-02, -1.30819798e-01,  6.16077660e-03,
       -2.13672787e-01, -2.12041158e-02, -1.37466148e-01,  9.74319279e-02,
        3.38687822e-02,  5.22094332e-02,  1.76095337e-01, -1.10202663e-01,
       -1.69056684e-01  1.56344213e-02  6.59077987e-02 -2.02557780e-02
```

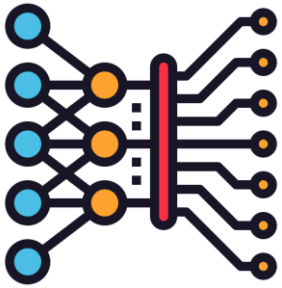
Models



Models



Traditional Machine Learning

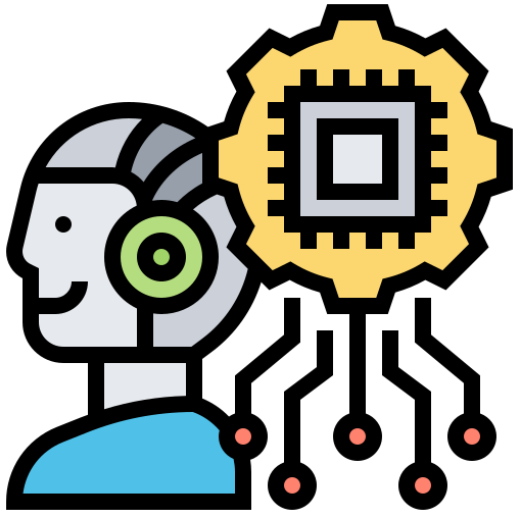


Deep Learning From Scratch

Transformers(Transfer Learning)

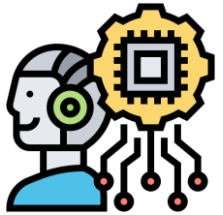
Models

- logistic regression
- SVM
- Naïve Bayes

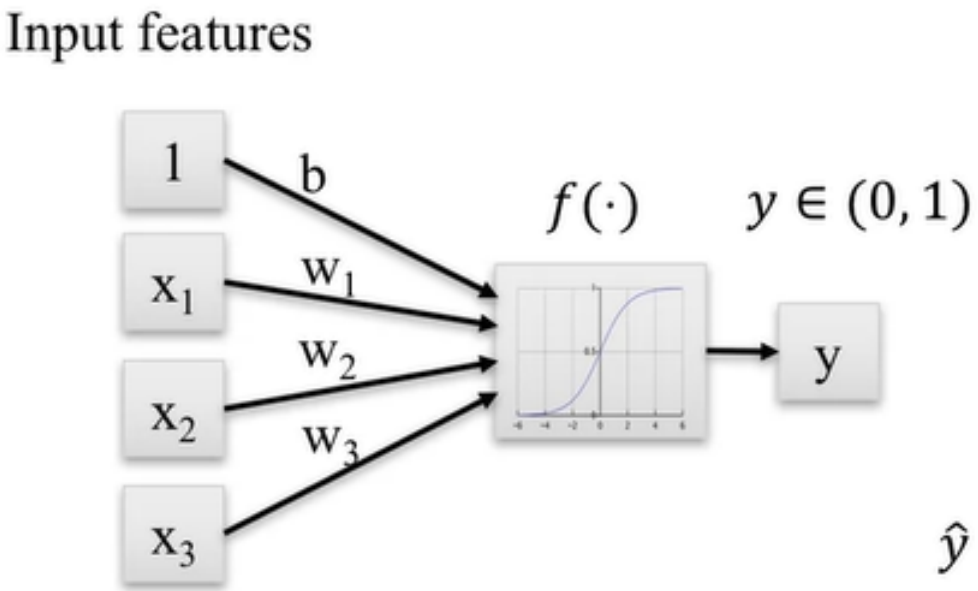


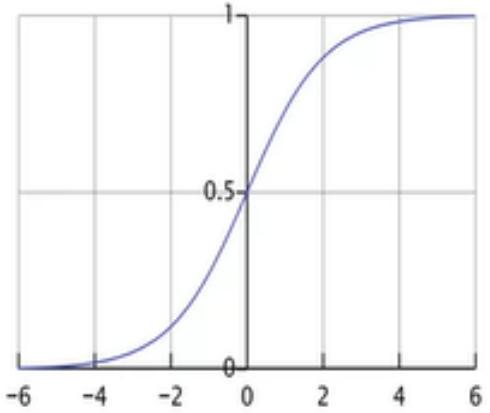
Machine Learning

Models **Machine Learning:**



logistic regression(Architecture):

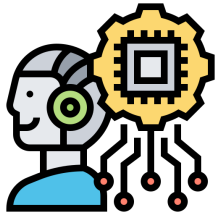




$$\hat{y} = \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)$$

$$= \frac{1}{1 + \exp\left[-(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)\right]}$$

Models **Machine Learning:**

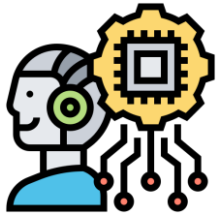


logistic regression (Evaluation):

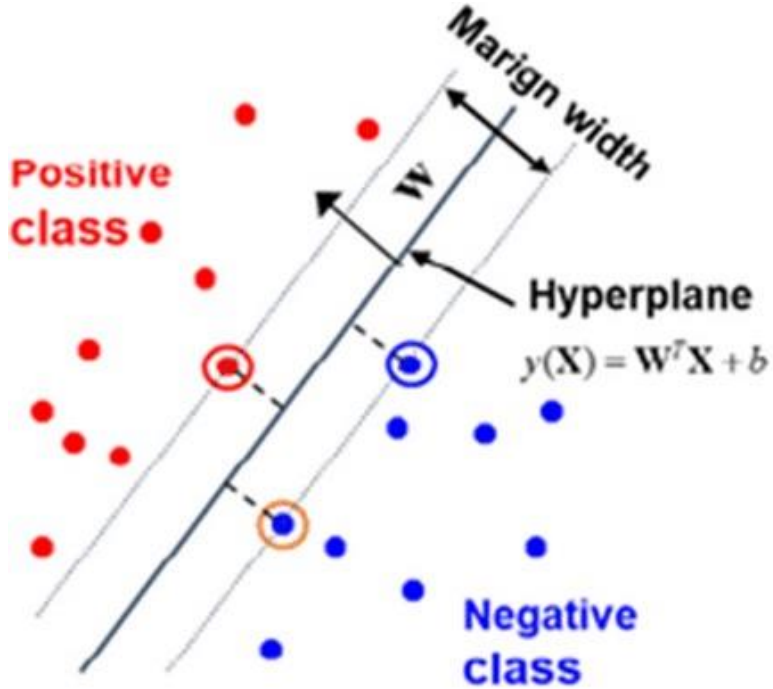
	precision	recall	f1-score	support
0	0.67	0.66	0.66	2693
1	0.52	0.38	0.44	2350
2	0.66	0.80	0.72	3061
accuracy			0.63	8104
macro avg	0.62	0.61	0.61	8104
weighted avg	0.62	0.63	0.62	8104

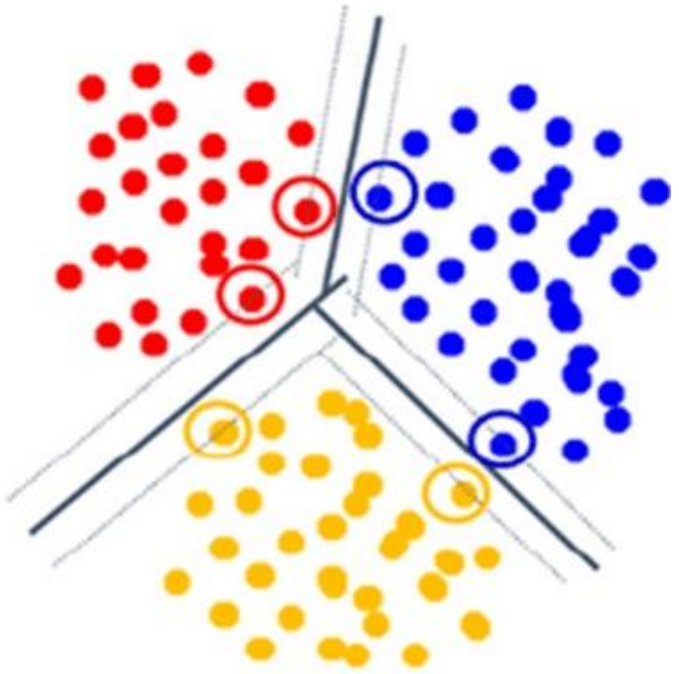
accuracy: 0.632

Models **Machine Learning:**

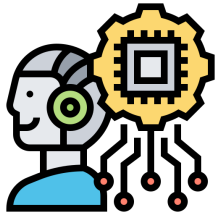


SVM(Architecture):





Models **Machine Learning:**



SVM(Evaluation):

	precision	recall	f1-score	support
0	0.59	0.72	0.65	2693
1	0.47	0.25	0.32	2350
2	0.64	0.76	0.70	3061
accuracy			0.60	8104
macro avg	0.57	0.57	0.56	8104
weighted avg	0.57	0.60	0.57	8104

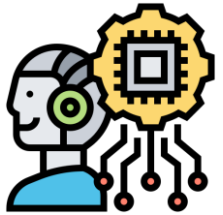
SVM with word2vec

	precision	recall	f1-score	support
0	0.70	0.72	0.71	2693
1	0.59	0.48	0.53	2350
2	0.71	0.79	0.75	3061
accuracy			0.68	8104
macro avg	0.67	0.67	0.66	8104
weighted avg	0.67	0.68	0.67	8104

accuracy: 0.680

SVM with TF-IDF

Models **Machine Learning:**

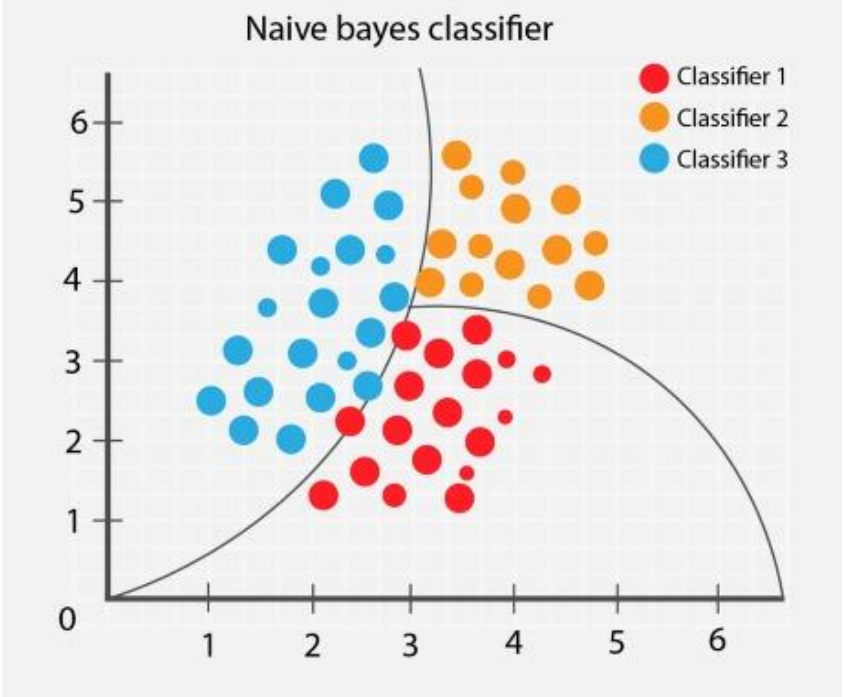


Naïve Bayes (Architecture):

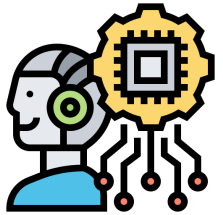
$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$



Models **Machine Learning:**



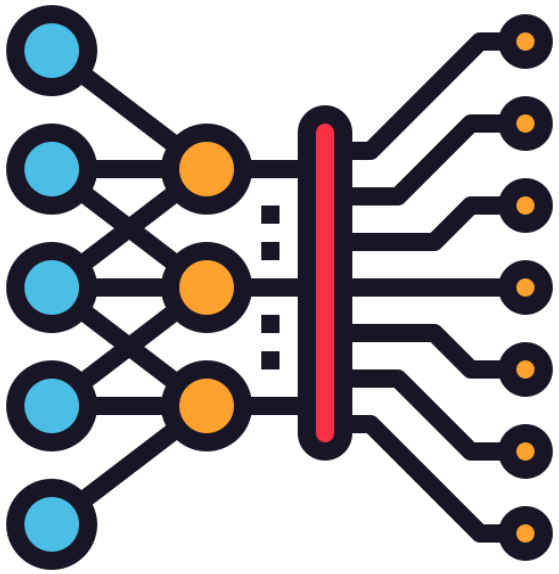
Naïve Bayes (Evaluation):

Naive Bayes REPORT.

	precision	recall	f1-score	support
0	0.73	0.61	0.66	4790
1	0.30	0.56	0.39	1943
2	0.78	0.66	0.71	5423
accuracy			0.62	12156
macro avg	0.60	0.61	0.59	12156
weighted avg	0.68	0.62	0.64	12156

Models

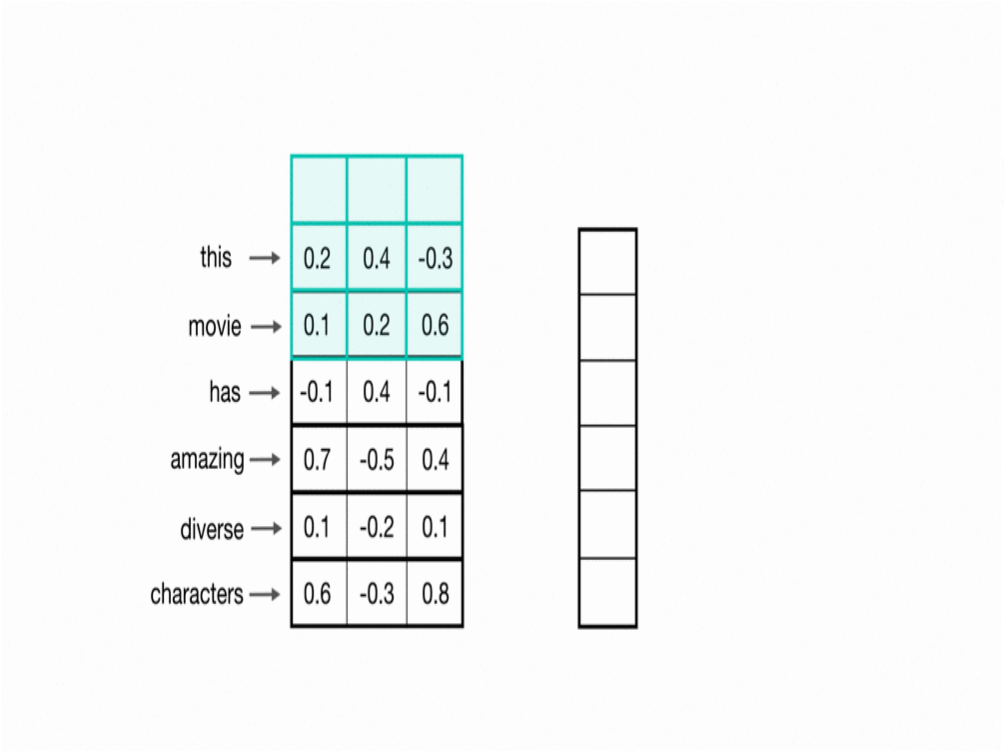
- CNN
- GRU
- LSTM

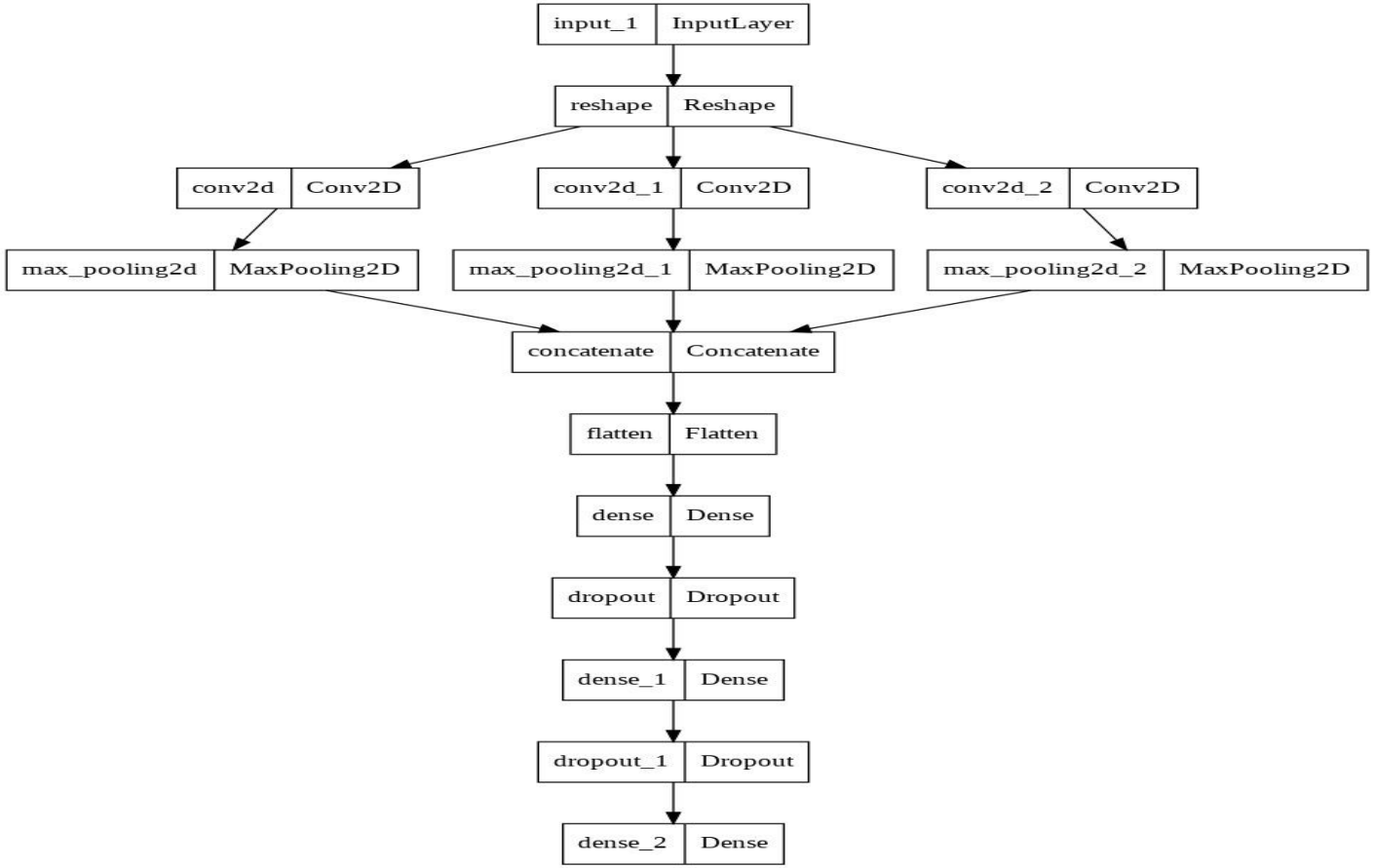


Deep Learning

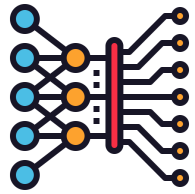
Models **Deep Learning:**

CNN (Architecture):





Models **Deep Learning:**



CNN (Evaluation):

	precision	recall	f1-score	support
0	0.56	0.67	0.61	2219
1	0.41	0.41	0.41	2381
2	0.74	0.65	0.69	3504
accuracy			0.58	8104
macro avg	0.57	0.58	0.57	8104
weighted avg	0.59	0.58	0.59	8104

Models **Deep Learning:**

GRU (Architecture):

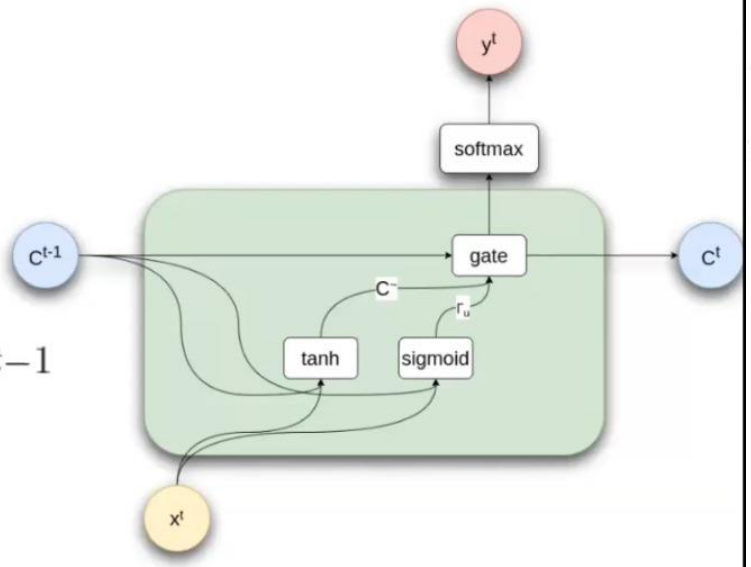
GRU

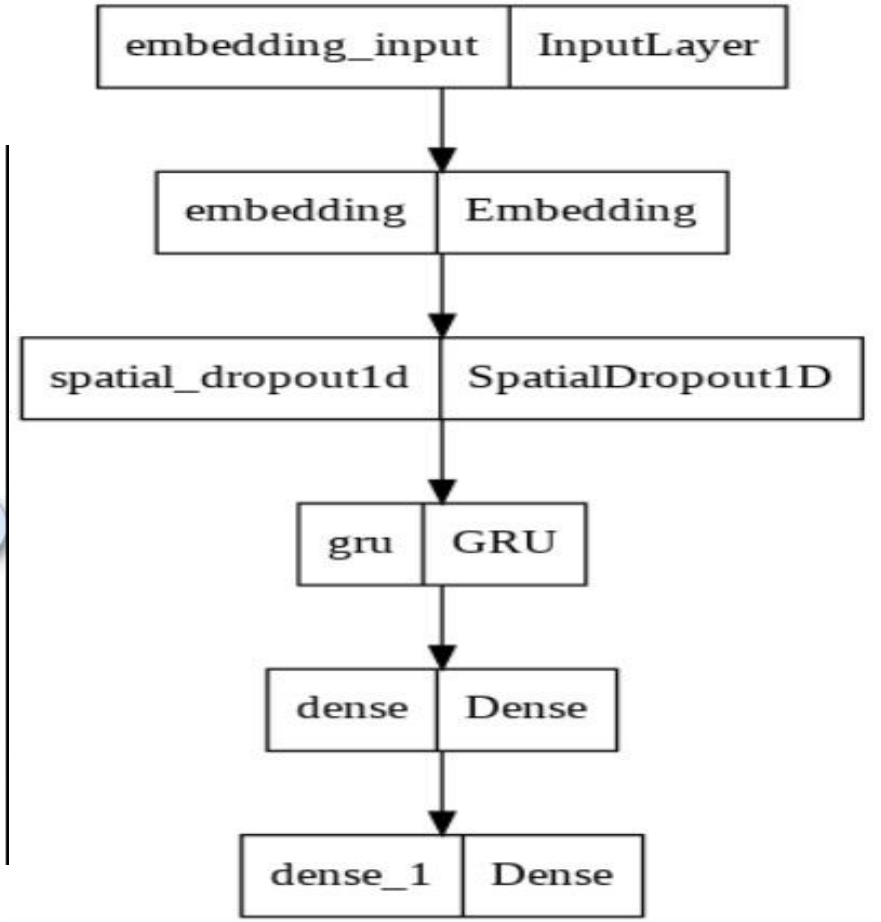
Sequential Modeling

$$\tilde{C} = tanh(W_c[C^{t-1}, X^t] + b_c)$$

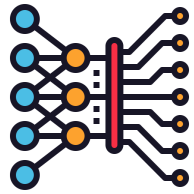
$$\Gamma_u = \sigma(W_u[C^{t-1}, X^t] + b_u)$$

$$C^t = \Gamma_u * \tilde{C} + (1 - \Gamma_u) * C^{t-1}$$





Models **Deep Learning:**



GRU (Evaluation):

	precision	recall	f1-score	support
0	0.65	0.75	0.70	2210
1	0.53	0.38	0.44	1873
2	0.70	0.76	0.73	2521
accuracy			0.65	6604
macro avg	0.63	0.63	0.62	6604
weighted avg	0.64	0.65	0.64	6604

Models **Deep Learning:**

LSTM (Architecture):

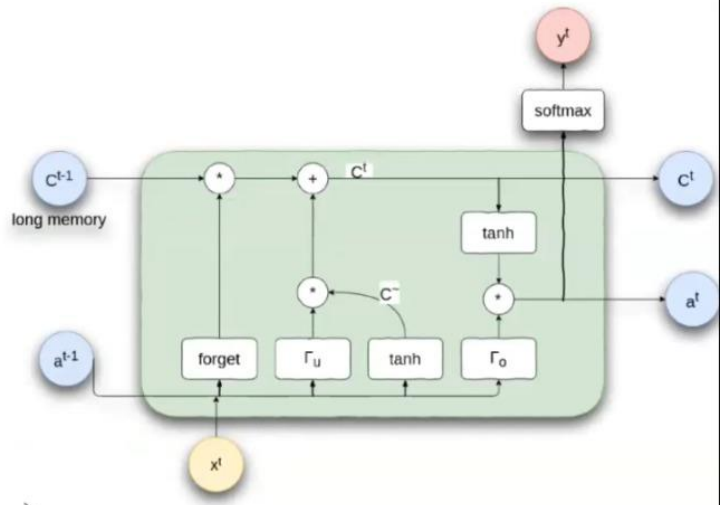
LSTM
Sequential Modeling

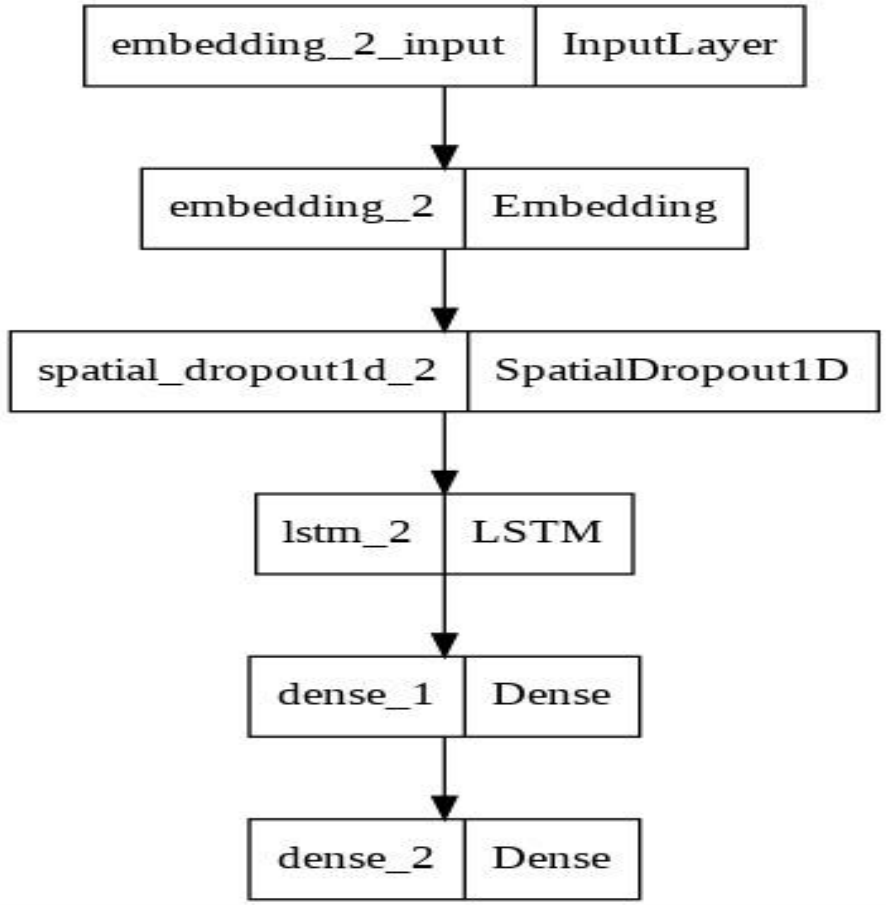
$$\Gamma_f = \sigma(W_f[a^{t-1}, x^t] + b_f)$$

$$\Gamma_u = \sigma(W_u[a^{t-1}, x^t] + b_u)$$

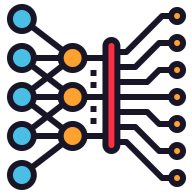
$$\Gamma_o = \sigma(W_o[a^{t-1}, x^t] + b_o)$$

$$\tilde{C} = tanh(W_c[a^{t-1}, x^t] + b_c)$$





Models **Deep Learning:**



LSTM (Evaluation):

	precision	recall	f1-score	support
0	0.70	0.68	0.69	960
1	0.55	0.42	0.48	830
2	0.65	0.79	0.72	1004
accuracy			0.65	2794
macro avg	0.63	0.63	0.63	2794
weighted avg	0.64	0.65	0.64	2794

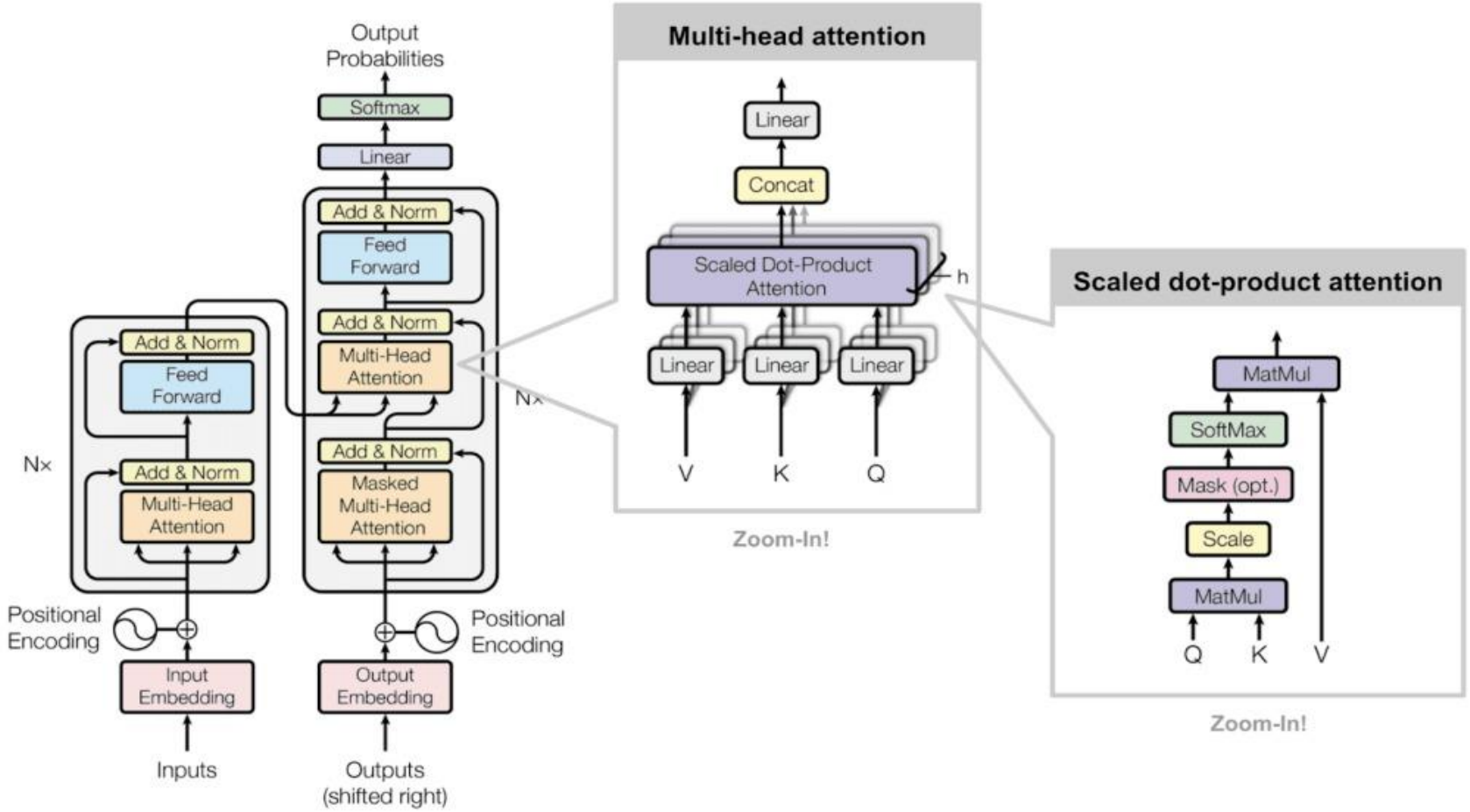
Models

Transformer (pretrained) :

BERT

Models

Transformers (Architecture):

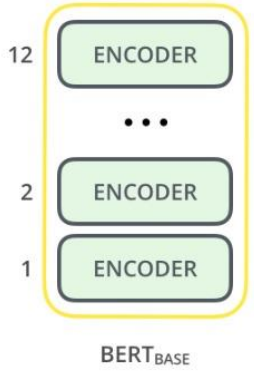


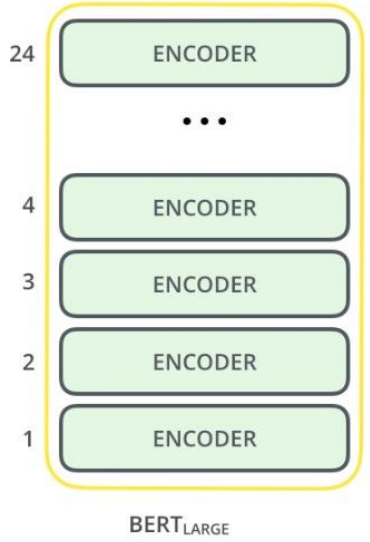
Models

Transformers: BERT (Architecture)

- BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads, 110M parameters.
- BERT-Large, Uncased: 24-layer, 1024-hidden, 16-heads, 340M parameters.
- BERT-Base, Cased: 12-layer, 768-hidden, 12-heads, 110M parameters.
- BERT-Large, Cased: 24-layer, 1024-hidden, 16-heads, 340M parameters.

Bidirectional Encoder Representations from Transformers





Models

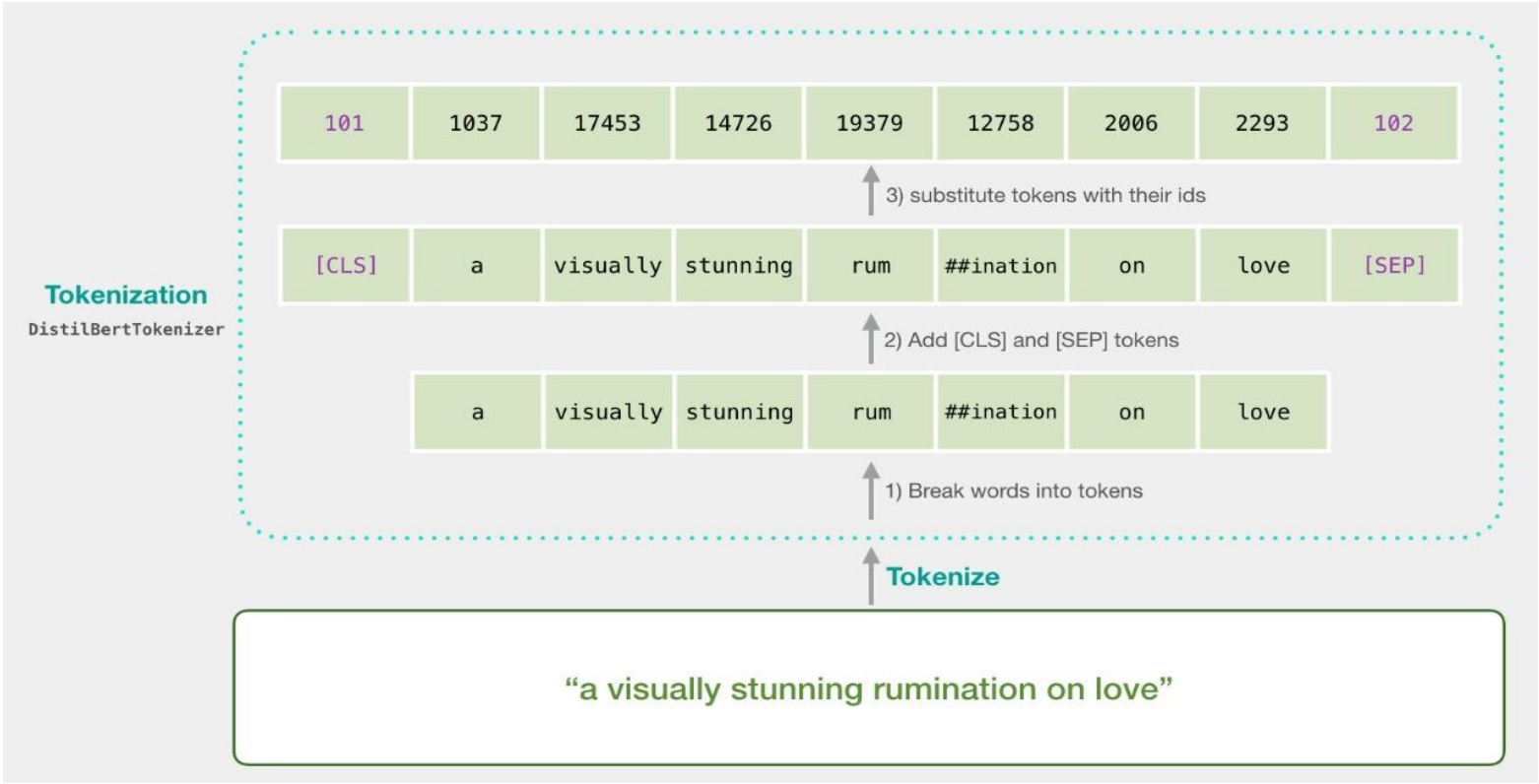
Transformers: BERT (Text Preprocessing)

```
print(len(encoding['input_ids'][0]))
encoding['input_ids'][0]
```

```
80
tensor([  101,  2296,  7292,  1045,  8833,  1999,  1012,  1012,  1012,  1996,
        10439,  5466,  2033,  2041,  1998,  2360,  1000,  2115,  5219,  2038,
         3092,  3531,  8833,  1999,  2153,  1000,  1045, 26618,  1999,  2005,
         1019,  2335,  1999,  5216,  1012,  1045,  2109,  2023, 10439,  2005,
         1016,  2086,  1998,  2023,  1996,  2034,  2051,  2000,  2227,  2023,
         3291,  1012,   102,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0])
```

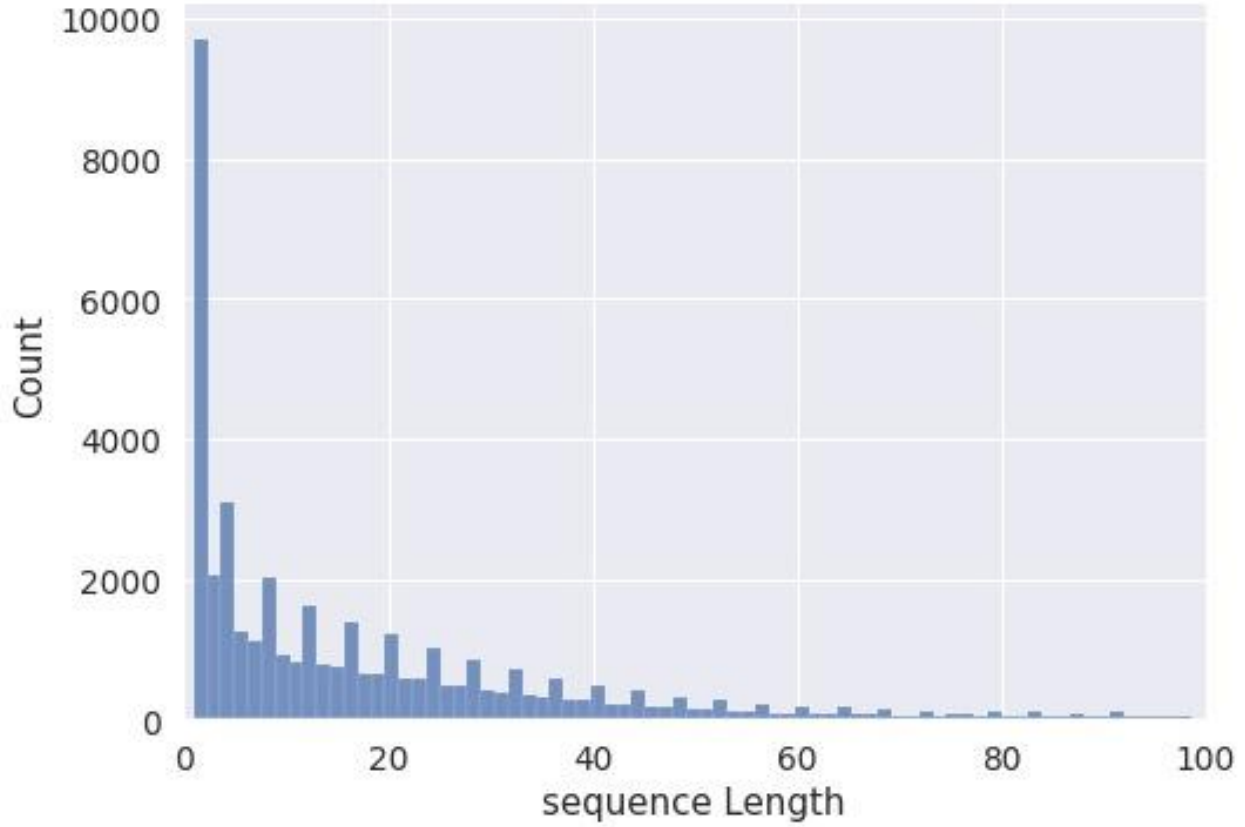
```
print(len(encoding['attention_mask'][0]))
encoding['attention_mask']
```

```
80
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0]])
```



Models

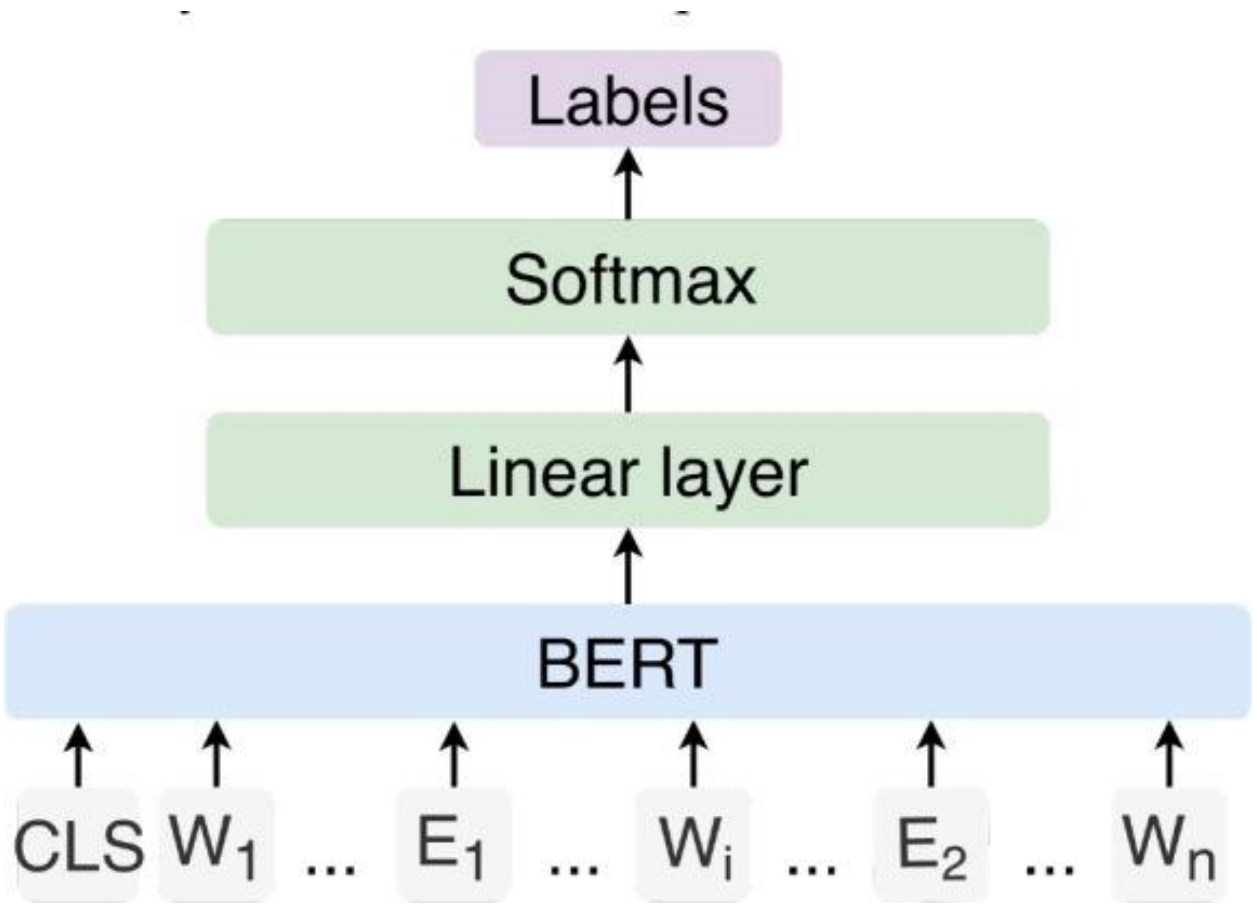
Transformers: BERT (Text Preprocessing)



Models

Transformers: BERT (Fine Tuning)

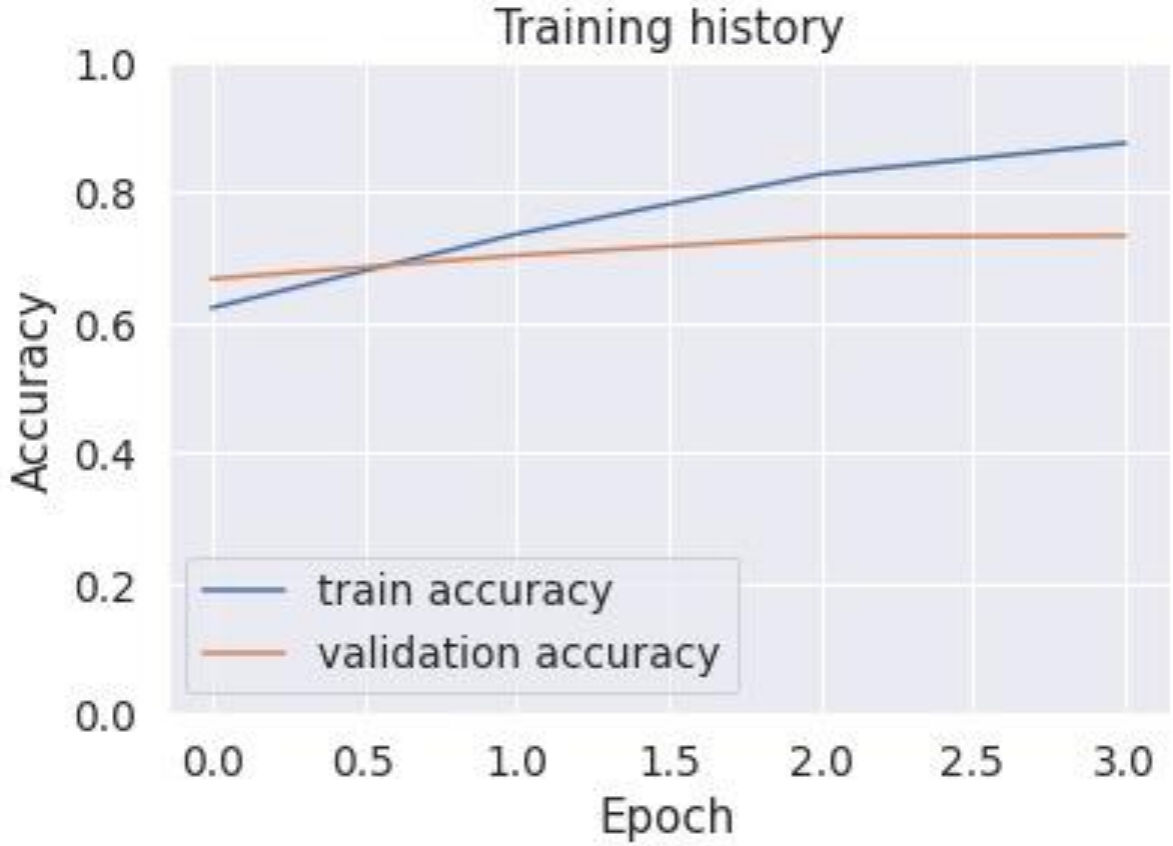
- Batch size: 16, 32
- Learning rate (Adam): 5e-5, 3e-5, 2e-5
- Number of epochs: 2, 3, 4



Models

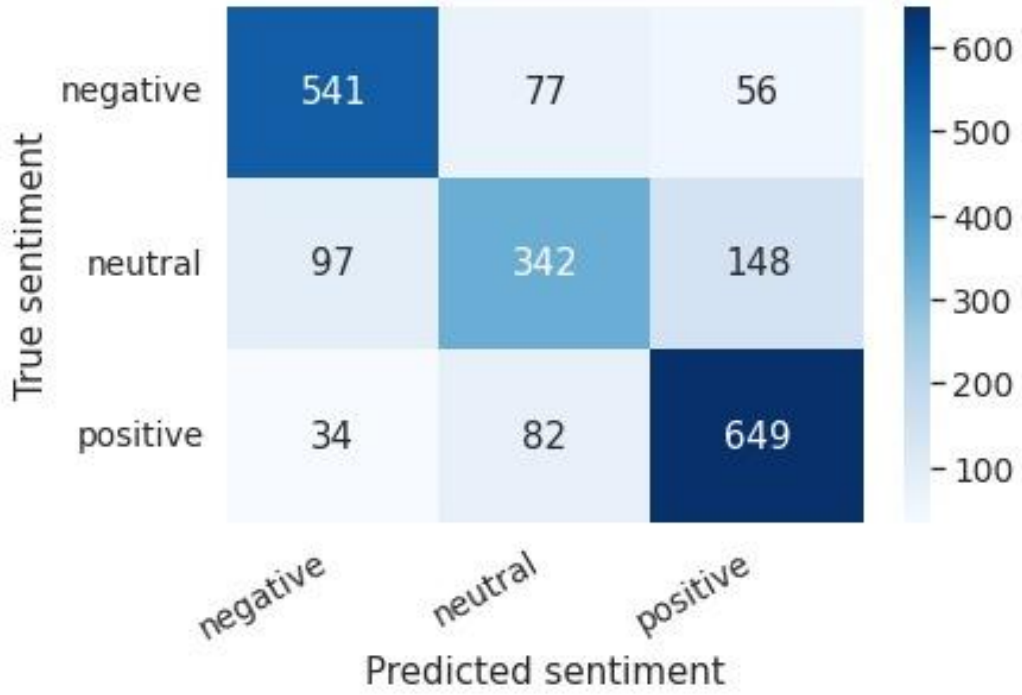
Transformers: BERT (Training)

```
Epoch 1/4
----------
Train loss 0.8296178524170006 accuracy 0.6224976690615917
Val   loss 0.7464140923693776 accuracy 0.6668311944718657

Epoch 2/4
----------
Train loss 0.6353758487915784 accuracy 0.7353973564416169
Val   loss 0.7077576396986842 accuracy 0.702862783810464

Epoch 3/4
----------
Train loss 0.4508616898993128 accuracy 0.8273734437558273
Val   loss 0.7428961358964443 accuracy 0.7309970384995064

Epoch 4/4
----------
Train loss 0.3412153007626011 accuracy 0.8748423188723743
Val   loss 0.8411004249937832 accuracy 0.7329713721618953

CPU times: user 48min 18s, sys: 18min 19s, total: 1h 6min 38s
Wall time: 1h 6min 39s
```



Models

Transformers: BERT (Evaluation)

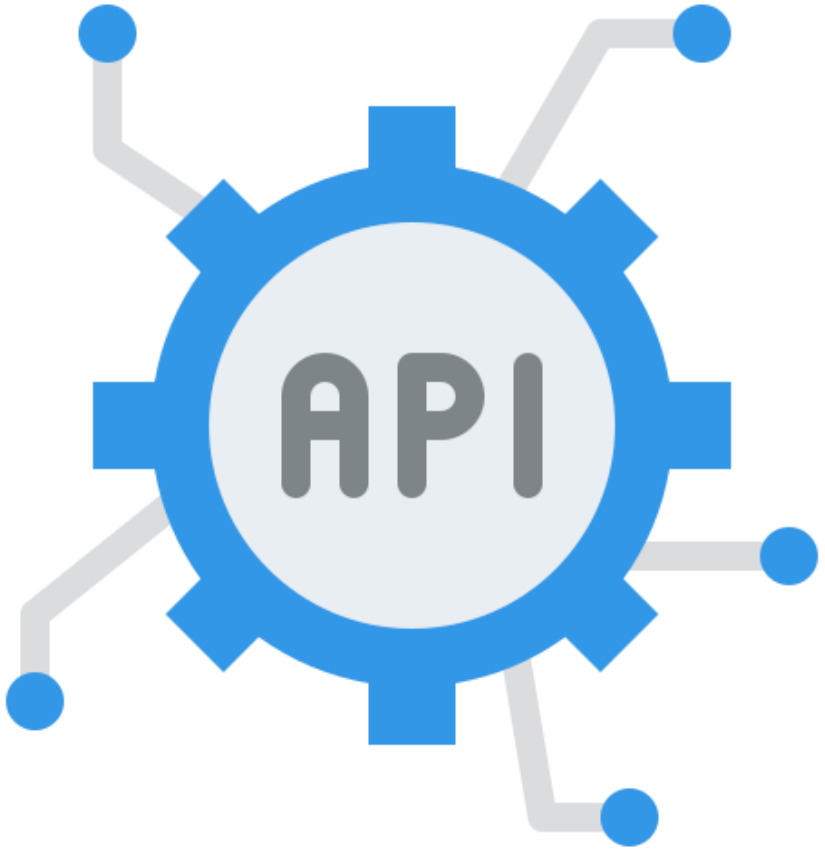


	precision	recall	f1-score	support
negative	0.81	0.80	0.80	674
neutral	0.68	0.58	0.63	587
positive	0.76	0.85	0.80	765
accuracy			0.76	2026
macro avg	0.75	0.74	0.74	2026
weighted avg	0.75	0.76	0.75	2026

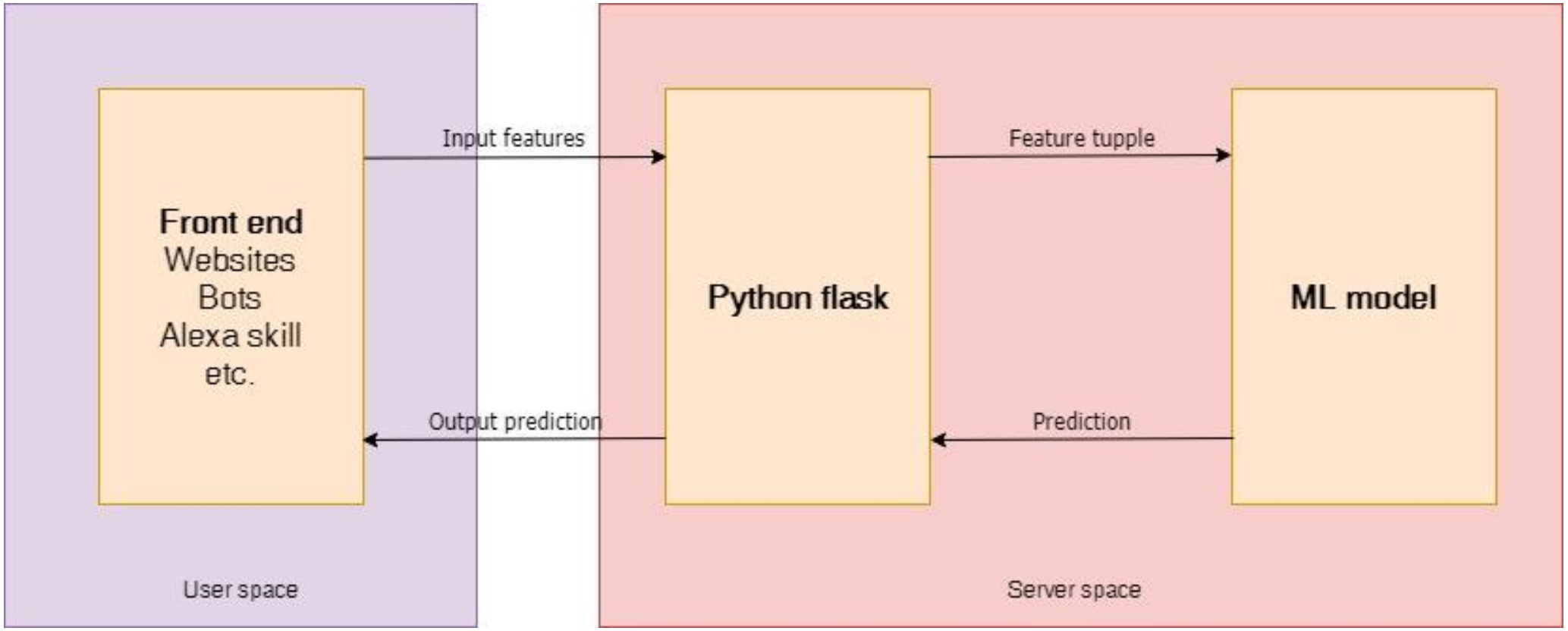
What is the Best Model?

Model	Type	Accuracy	F1 Score			Pros	Cons
			🤝	🙂	🤜		
SVM	Traditional ML	0.68	0.75	0.53	0.71	- Fast - Low Resources	- Low performance with large data
LSTM	DL from scratch	0.65	0.72	0.48	0.69	- Represent sequential data - Saving short and long history	- No parallelization - Slow - Vanishing gradient
BERT 🏆	Transfer Learning	0.76	0.80	0.63	0.80	- Parallelization - Good result	- large data - high resource

API

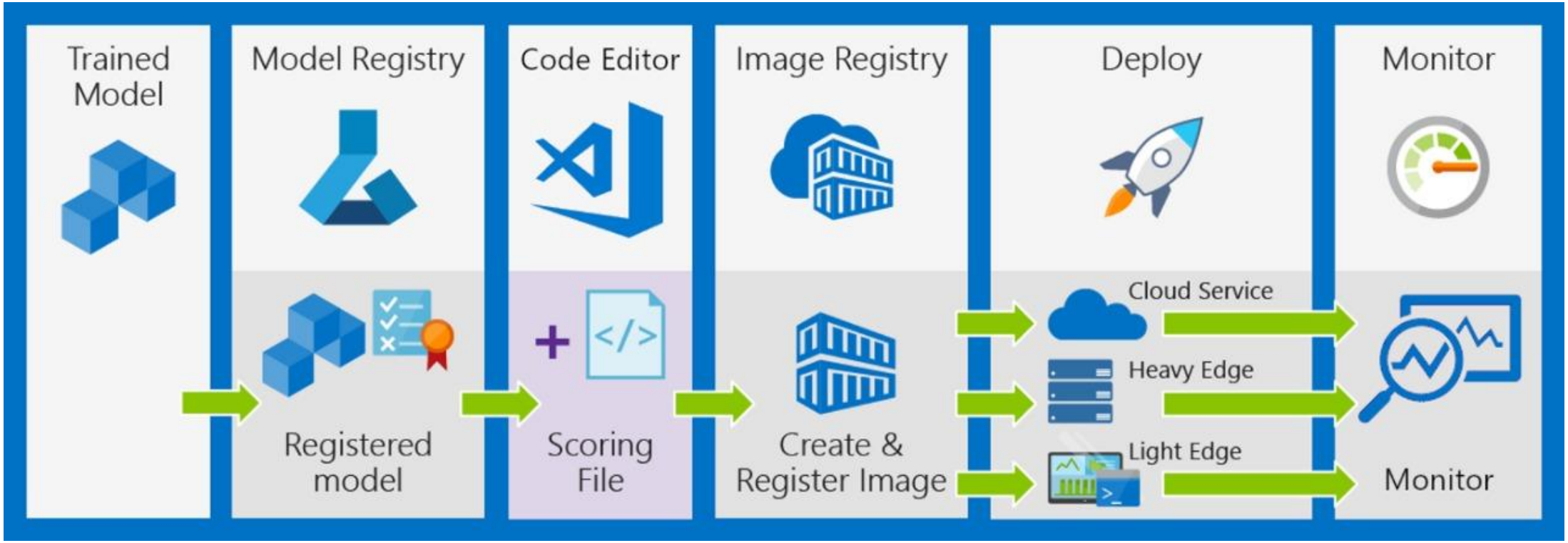


API

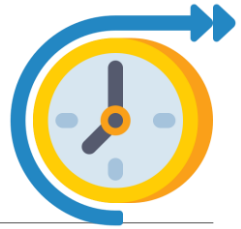

Front end
Websites
Bots
Alexa skill
etc.
Input features
Output prediction
Python flask
Feature tupple
Prediction
ML model
User space
Server space

Deployment

Deployment


Trained Model
Model Registry
Code Editor
Image Registry
Deploy
Monitor
Registered model
Scoring File
Create & Register Image
Cloud Service
Heavy Edge
Light Edge
Monitor

Future Work

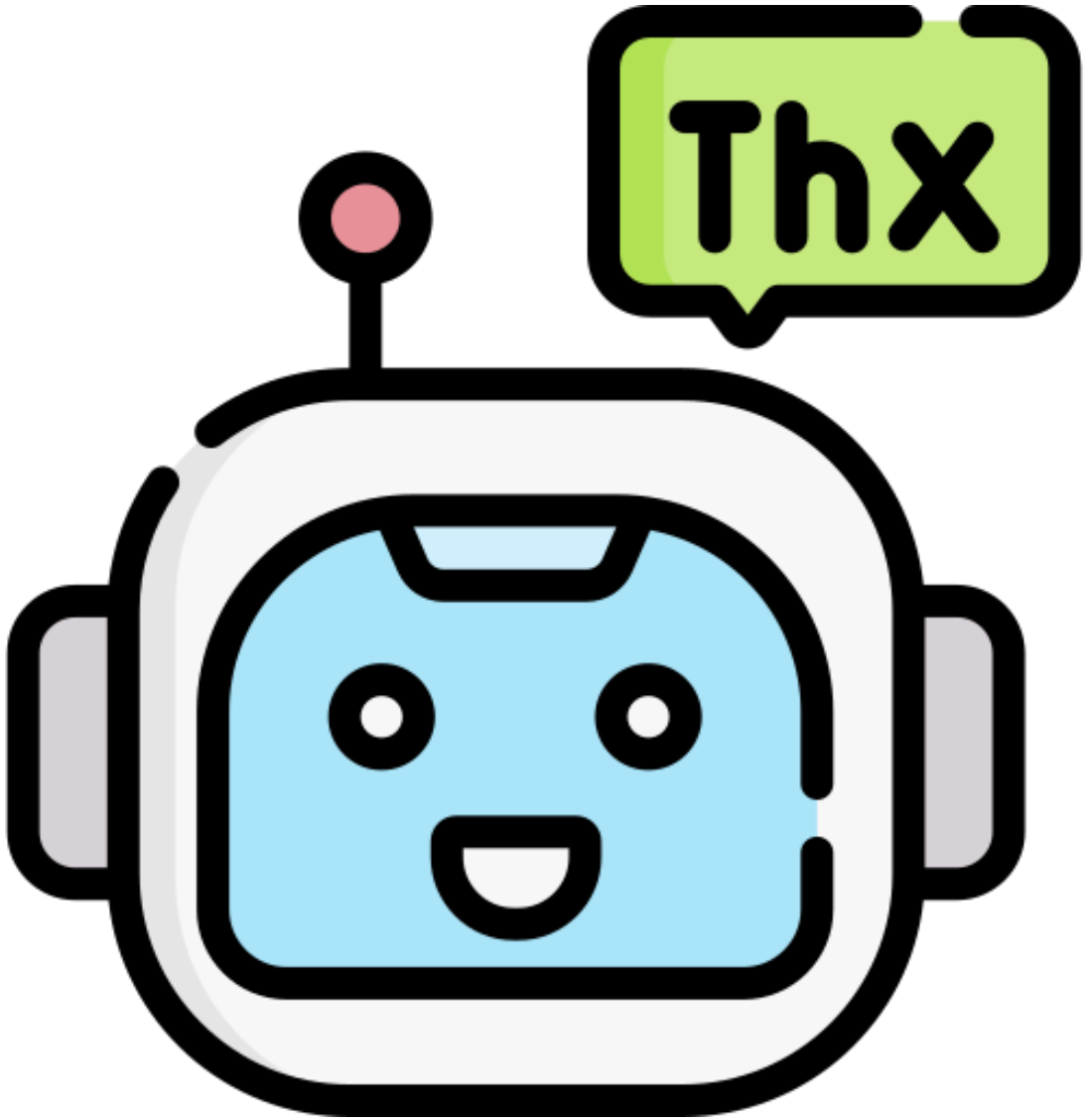


- Scrapping data automatically and store it at database
- Chatbot

https://google-play-reviews-sentiment-analysis.azurewebsites.net/

Papers:

- Pre-training of Deep Bidirectional Transformers for Language Understanding
- Deep Learning for Sentiment Analysis on Google Play Consumer Review
- Study of Sentiment on Google Play Store Applications
- Attention Is All You Need


Thx