DSA_202101_ 17: Empathetic Conversational Model

Use Case: Arabic Language NLU

Supervised By: Prof. Arya.

TA. Pouya Khodaee

Team Members

Last Name, First Name
Abdelazim, Sara
Fahem, Noha
Abdullah, Alaa
Mousa, Naser

Problem Formulation:

There are challenges in Natural Language Understanding regarding Arabic language, and this is due to the lack of appropriate datasets available for training. So, our goal for this project is to develop an empathetic Arabic conversational chatbot.

Data Preparation:

1. Snippet of the data:

	emotion	context	response
0	sentimental	أتذكر أنني ذهبت لمشاهدة الألعاب النارية مع أعز...	هل كان هذا صديقًا كنت تحبه أم مجرد أفضل صديق؟
1	sentimental	كان هذا أفضل صديق. اشتقت لها.	اين ذهبت؟
2	sentimental	لم نعد نتحدث.	هل كان هذا شيء حدث بسبب جدال؟
3	afraid	أشعر وكأنني ضرب على جدار فارغ عندما أرى الظلام	أجل؟ أنا حقا لا أرى كيف
4	afraid	ألا تشعر بذلك .. إنه لأمر عجيب	أصطدم في الواقع بجدران فارغة في كثير من الأحيا...

```
RangeIndex: 36628 entries, 0 to 36627
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   emotion   36628 non-null  object
 1   context   36628 non-null  object
 2   response  36628 non-null  object
dtypes: object(3)
```

	emotion	context	response
count	36628	36628	36628
unique	32	36093	35344
top	surprised	5	ماذا حدث؟
freq	1886	25	62

```
array(['sentimental', 'afraid', 'proud', 'faithful', 'terrified',
       'joyful', 'angry', 'sad', 'jealous', 'grateful', 'prepared',
       'embarrassed', 'excited', 'annoyed', 'lonely', 'ashamed', 'guilty',
       'surprised', 'nostalgic', 'confident', 'furious', 'disappointed',
       'caring', 'trusting', 'disgusted', 'anticipating', 'anxious',
       'hopeful', 'content', 'impressed', 'apprehensive', 'devastated'],
      dtype=object)
```

2. Data Cleaning:

2.1 Emotion Grouping

Using this table as a reference we grouped each set of emotions to belong to a specific category.

Grouped Emotion Labels	Complete Emotion Labels
Joy	Excited
	Proud
	Grateful
	Hopeful
	Confident
	Joyful
	Content
	Prepared
	Anticipating
Love	Caring
	Sentimental
	Trusting
	Faithful
	Nostalgic
Surprise	Surprised
	Impressed
Sadness	Sad
	Lonely
	Guilty
	Disappointed
	Devastated
	Embarrassed
	Ashamed
Anger	Angry
	Annoyed
	Furious
	Disgusted
	Jealous
Fear	Afraid
	Terrified
	Anxious
	Apprehensive

And this resulted into 6 categories:

```
array(['Love', 'fear', 'joy', 'anger', 'Sadness', 'surprise'],
```

2.2 Maintaining Arabic text and removing any other existing languages

```
data['language'] = data['response'].apply(detect_language)
data['language'] = data['context'].apply(detect_language)
# Filter out English rows
data = data[data['language'] != 'en']
# Remove English columns
data = data.drop('language', axis=1)
```

	emotion	context	response	language
0	sentimental	أتذكر أنني ذهبت لمشاهدة الألعاب النارية مع أعز...	هل كان هذا صديقًا كنت تحبه أم مجرد أفضل صديق؟	ar
1	sentimental	كان هذا أفضل صديق. اشتقت لها.	اين ذهبت؟	ar
2	sentimental	لم نعد نتحدث.	هل كان هذا شيء حدث بسبب جدال؟	ar
3	afraid	أشعر وكأنني ضرب على جدار فارغ عندما أرى الظلام	أجل؟ أنا حقا لا أرى كيف	ar
4	afraid	ألا تشعر بذلك .. إنه لأمر عجيب	أصطدم في الواقع بجدران فارغة في كثير من الأحيا...	ar

2.3 Handling Duplicates

```
data.duplicated().sum()
data = data.drop_duplicates()
```

2.4 Standardize text (Tashkeel): Remove diacritics.

2.5 Normalize Arabic text to standard form: uniform Arabic text typography convention.

2.6 Remove non-Arabic characters, numbers, and extra spaces.

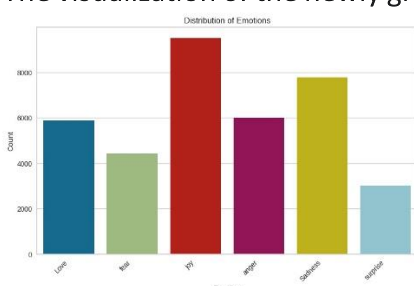
2.7 Remove Arabic Stop words: using a customized Arabic stop words list as it performed better.

```
def preprocess_arabic_text(text):
    # Remove diacritics (Tashkeel) to standardize text
    text = strip_tashkeel(text)
    # Normalize Arabic text to standard form
    text = araby.normalize_ligature(text)
    # Remove non-Arabic characters, numbers, and extra spaces
    text = re.sub(r'[^\u0600-\u06FF\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    text = ' '.join([word for word in text.split() if word not in arabic_stop_words])
    return text
```

Data Exploration:

1. Distribution of Emptions

   The visualization of the newly grouped emotions shows joy is the most occurring category.



2. Word Diversity Visualization

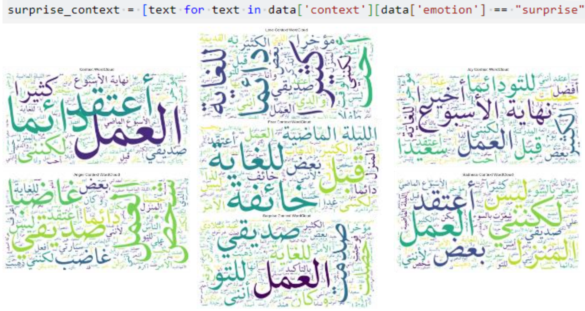
   Using word cloud plot to display the distribution of words in terms of emotions.

   We first filterd the context for each category then we plotted them.

   For each category we notice relevant Arabic words we notice:

   Love: (أحب=I love) Joy: (سعيد =Happy), *Anger:* (غاضب=mad),for *Fear:* (خائف =scared),Surprise (صدمت =shocked), Sadness (أشعر بالسوء=I feel bad)

```
love_context = [text for text in data['context'][data['emotion'] == "Love"]]
joy_context = [text for text in data['context'][data['emotion'] == "joy"]]
sadness_context = [text for text in data['context'][data['emotion'] == "Sadness"]]
fear_context = [text for text in data['context'][data['emotion'] == "fear"]]
anger_context = [text for text in data['context'][data['emotion'] == "anger"]]
surprise_context = [text for text in data['context'][data['emotion'] == "surprise"]]
```



Feature Engineering:

We applied:

1. Term Frequency Inverse Document Frequency of records (TFIDF)

	00	000	10	100	1000	10000	100000	101	103	104	...	يون	يونان	يوناني	يونايتد	يونس	يونيفرسال	يونيو	يويو	ييكيس	ييل
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	0.0	0.0	0.0	0.0	0.0	0.0	0.0
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	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0

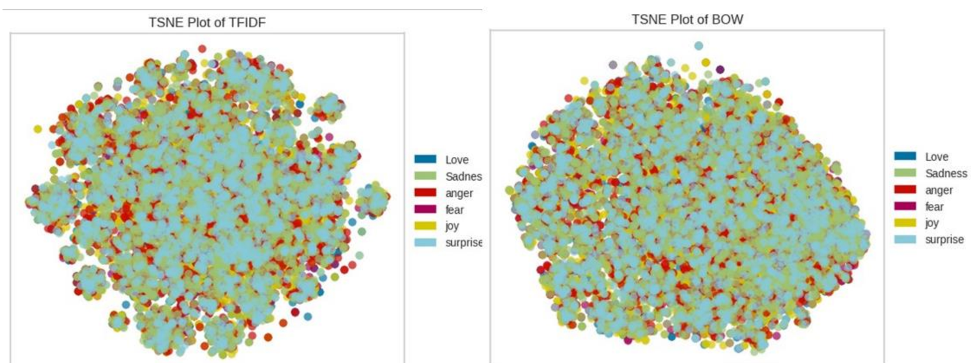
5 rows × 12155 columns

2. Bag of Words (BoW)

	00	000	10	100	1000	10000	100000	101	103	104	...	يون	يونان	يوناني	يونايتد	يونس	يونيفرسال	يونيو	يويو	ييكيس	ييل
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

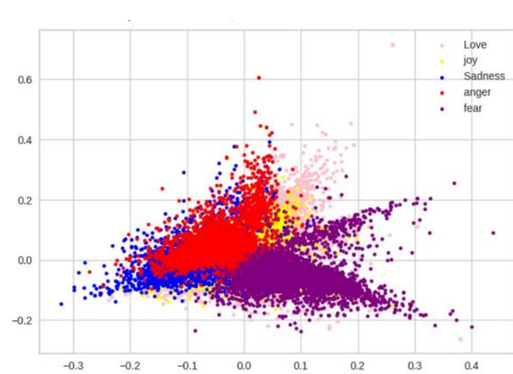
5 rows × 12155 columns

Then we used t-SNE plot to visualize the complexity of the data at hand. We notice an overlap among categories.



Clustering:

K-means was used to cluster the data points, and silhouette score was used to evaluate the model. The score is relevantly low as the data points of each category are overlapped.




```
Silhouette Score: 0.08480114642326161
```

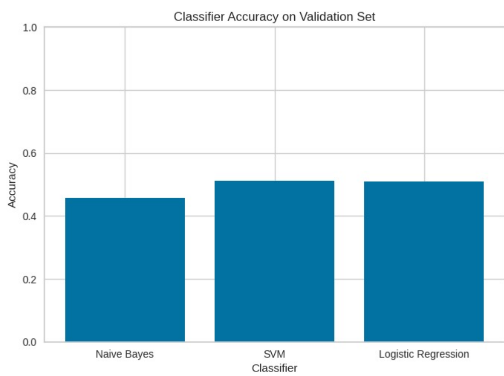
Classification:

The following models were used to classify our data points

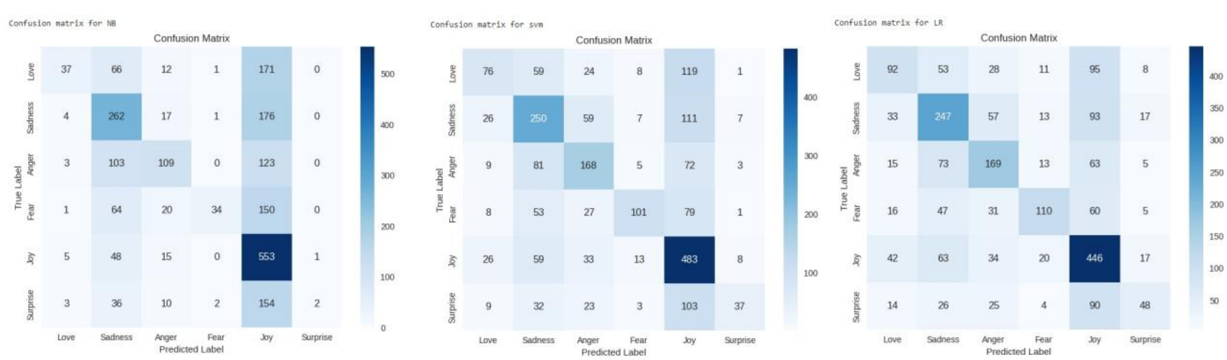
- Naive Bayes
- Support Vector Machine
- Logistic Regression

The results are shown in the following figure.

Among these models, SVM achieved the highest accuracy at 55%, followed closely by logistic regression at 54%



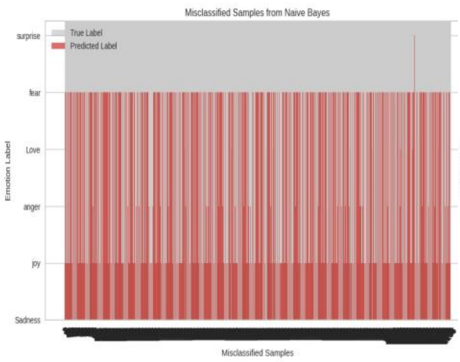
We noticed that the three algorithms were able to reliably predict instances labeled as "joy" and "sadness." However, they faced challenges when it came to accurately predicting instances belonging to other categories, indicating limitations in their ability to distinguish between more diverse emotions.

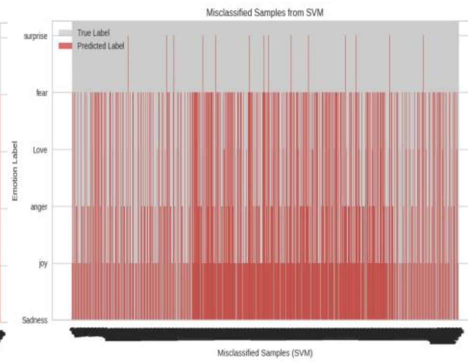


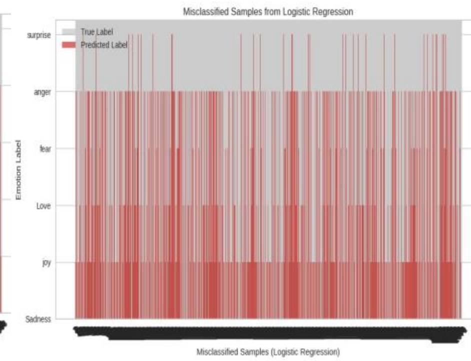
Error Analysis:

After conducting the error analysis for the algorithm, it became evident that the model's performance is subpar. Despite achieving moderate accuracy, the model struggles to predict emotions beyond 'joy' and 'sadness' effectively.

Further investigation and improvements are necessary to enhance the algorithm's accuracy and broaden its ability to correctly predict a wider range of emotions.







Deployment:

Deployment Preparation:

As we noticed the model performance rin the error analysis stage we needed to apply some modification before deploying the model so we used the context and emotion to split our data.

```python
# Use only 5000 samples from the original dataset
data = data.sample(n=3000, random_state=42)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    data['context'], data['emotion'], test_size=0.2, random_state=42
)
```

Then the deployment model was prepared, where a link of the model is generated then passed to dialog flow for the chatbot.

```python
@app.route('/get_response', methods=['POST'])
def get_response():
    if request.method == 'POST':
        data = request.json  # Assuming the incoming data is in JSON format
        context = data.get('queryResult').get('queryText')  # Extract the context from the JSON data

        # Preprocess the context text (if needed)
        preprocessed_context = preprocess_text(context)

        # Transform the preprocessed context into TF-IDF features
        context_tfidf = tfidf_vectorizer.transform([preprocessed_context])

        # Use the trained Logistic Regression model to predict the emotion for the context
        predicted_emotion = clf_lr.predict(context_tfidf)[0]

        # Get a random response for the predicted emotion
        response = random.choice(responses.get(predicted_emotion, ["I don't have a response for that emotion."]))

        # Return the response as a JSON object in Dialogflow format
        return jsonify({
            'fulfillmentText': response,
        })
```

Before goging to dialog-flow directly we wanted to ensure the link is working well, so we used postman to check it’s availability.

Lastly we went to dialog-flow console to deploy our model into our chatbot.