MSR 2024 MINING CHALLENGE

SOFTWARE ENGINEERING (CEN-5035) PROJECT

AkhilRaj Tirumalasetty (AT21BO)
Manikanta Mamidi (MM23BM)
Sruthijha Pagolu (SP23BU)

INTRODUCTION

The primary objective of our study is to examine and improve the interaction dynamics of ChatGPT, especially when dealing with programming-related queries and conversations.

To achieve this, we have formulated three research questions focusing on predicting conversation length with ChatGPT.

The project's scope includes data collection and cleaning, feature engineering, model training and evaluation, culminating in comprehensive documentation and presentation of findings.

WORKING:

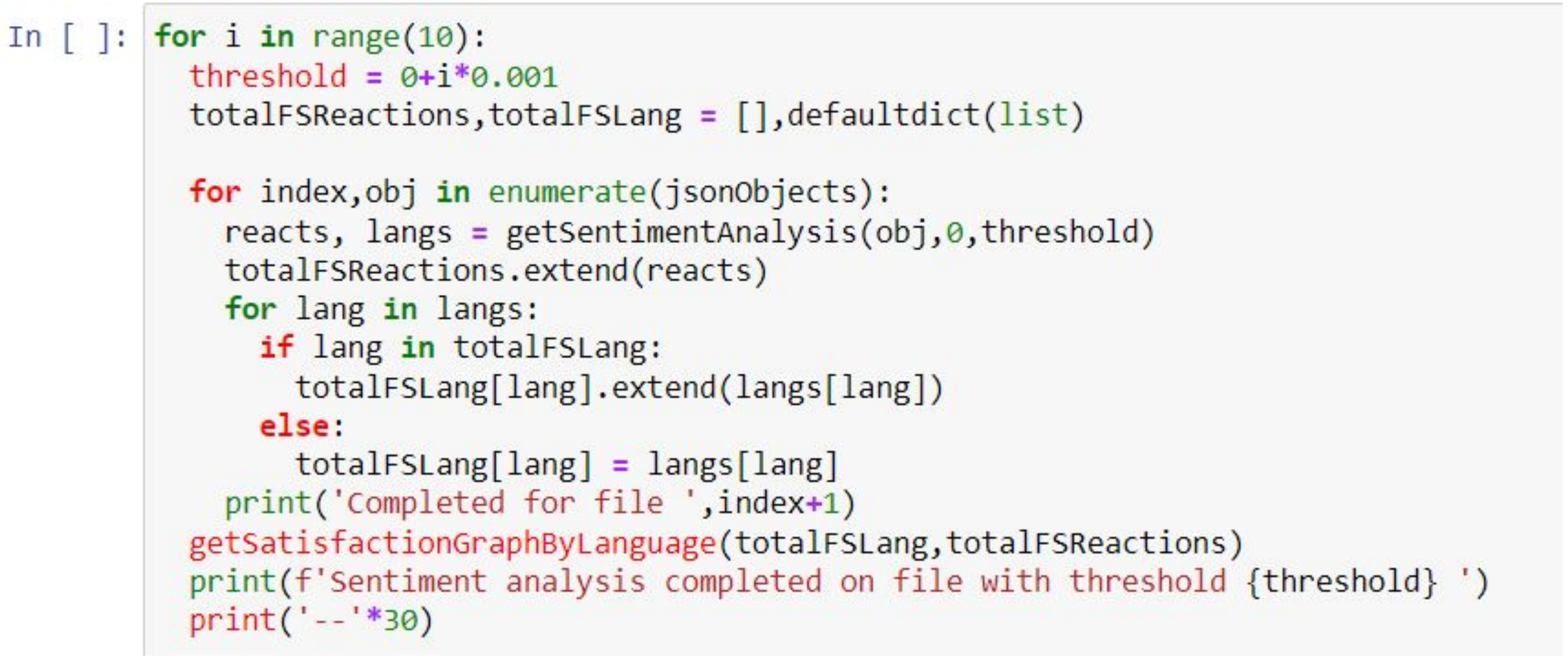
- Explored JSON file patterns, addressing challenges in diverse patterns and multiple files.
- Used NLP tools to forecast ChatGPT conversation length from prompts and context.
- Extracted and organized user-ChatGPT conversations.
- Applied sentiment analysis to evaluate conversation quality and produce output.
- Employed neural networks and regression techniques to construct a predictive model.

RESEARCH QUESTION 1

How accurately can we predict the length of a conversation with ChatGPT based on the initial prompt and context provided?

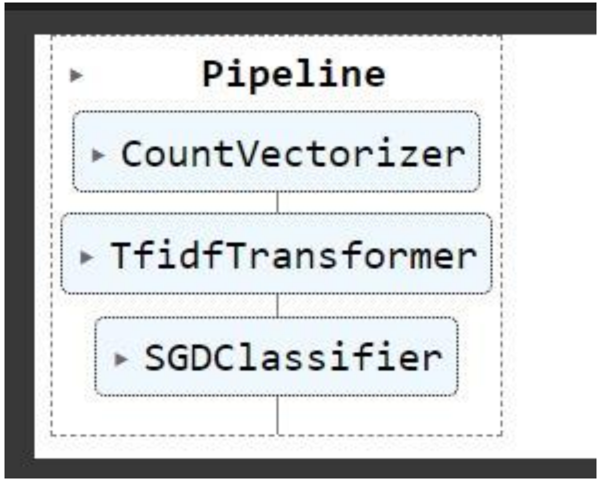
We aim to investigate and develop predictive models that can accurately anticipate the duration of a conversation with ChatGPT based on the information provided in the initial prompt and context.

CODE RQ1

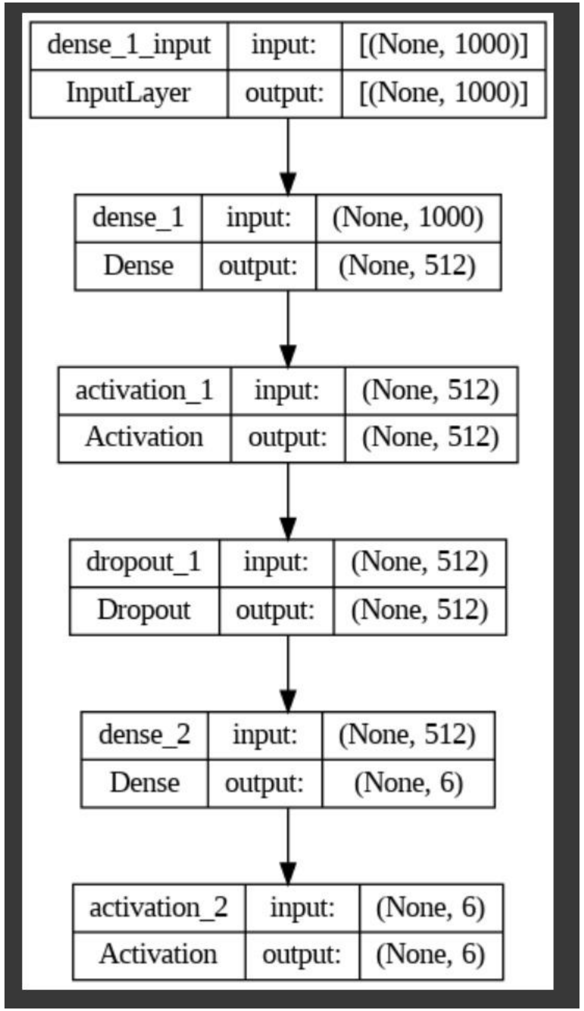


```
In [ ]: for i in range(10):
          threshold = 0+i*0.001
          totalFSReactions,totalFSLang = [],defaultdict(list)

          for index,obj in enumerate(jsonObjects):
            reacts, langs = getSentimentAnalysis(obj,0,threshold)
            totalFSReactions.extend(reacts)
            for lang in langs:
              if lang in totalFSLang:
                totalFSLang[lang].extend(langs[lang])
              else:
                totalFSLang[lang] = langs[lang]
            print('Completed for file ',index+1)
          getSatisfactionGraphByLanguage(totalFSLang,totalFSReactions)
          print(f'Sentiment analysis completed on file with threshold {threshold} ')
          print('--'*30)
```

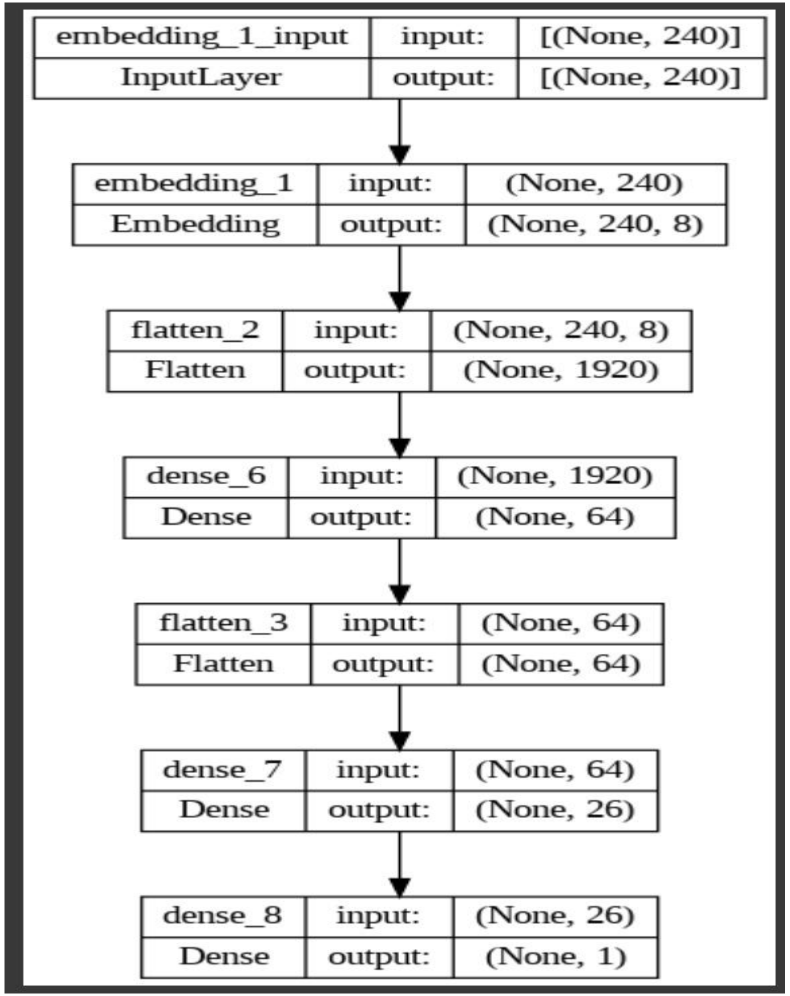
Model 1: Linear Support Vector Machines



Model 2: Keras


dense_1_input | input: | [(None, 1000)]
InputLayer | output: | [(None, 1000)]
dense_1 | input: | (None, 1000)
Dense | output: | (None, 512)
activation_1 | input: | (None, 512)
Activation | output: | (None, 512)
dropout_1 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_2 | input: | (None, 512)
Dense | output: | (None, 6)
activation_2 | input: | (None, 6)
Activation | output: | (None, 6)

Model 3: LSTM


embedding_1_input
InputLayer
input:
output:
[(None, 240)]
[(None, 240)]
embedding_1
Embedding
input:
output:
(None, 240)
(None, 240, 8)
flatten_2
Flatten
input:
output:
(None, 240, 8)
(None, 1920)
dense_6
Dense
input:
output:
(None, 1920)
(None, 64)
flatten_3
Flatten
input:
output:
(None, 64)
(None, 64)
dense_7
Dense
input:
output:
(None, 64)
(None, 26)
dense_8
Dense
input:
output:
(None, 26)
(None, 1)

ACCURACIES OF VARIOUS MODELS

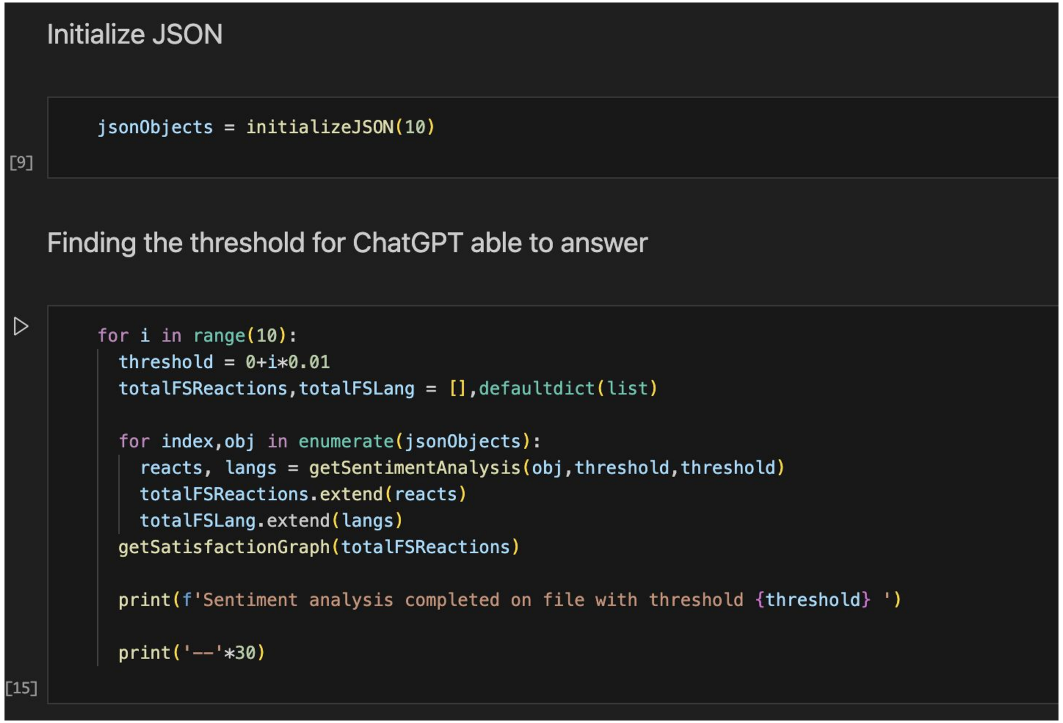
- Linear Support Vector Machines: 89.07
- Keras: 97.42
- LSTM (designed to handle sequential data, such as time series, speech, and **text**): 76.64

RESEARCH QUESTION 2

To what extent can ChatGPT correctly respond to various programming languages on the developer’s given prompt?

This helps to evaluate ChatGPT's ability to accurately understand and respond to prompts or instructions given by developers in different programming languages

CODE RQ2



Initialize JSON

```
jsonObjects = initializeJSON(10)
```

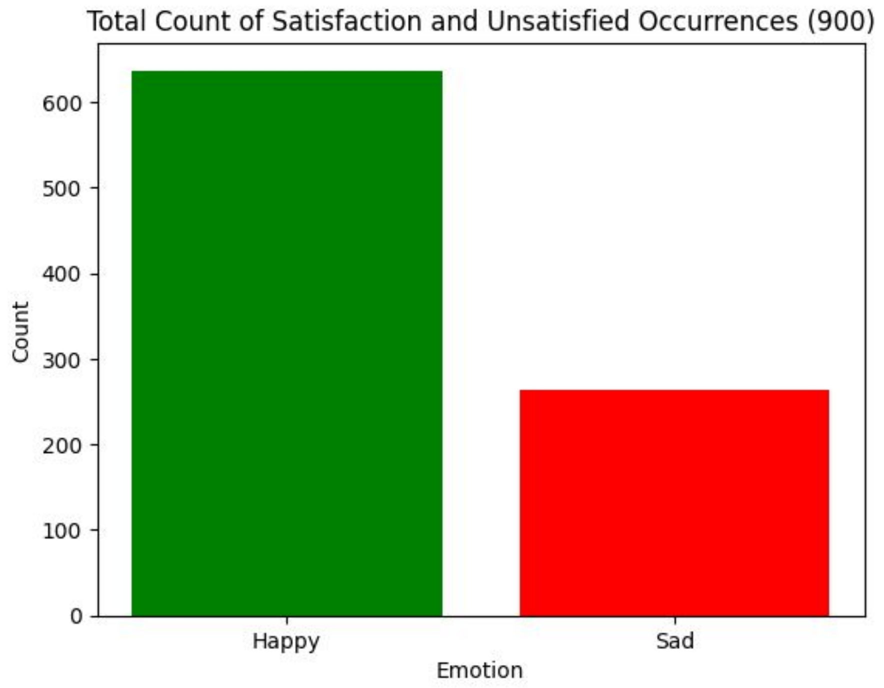
[9]

Finding the threshold for ChatGPT able to answer

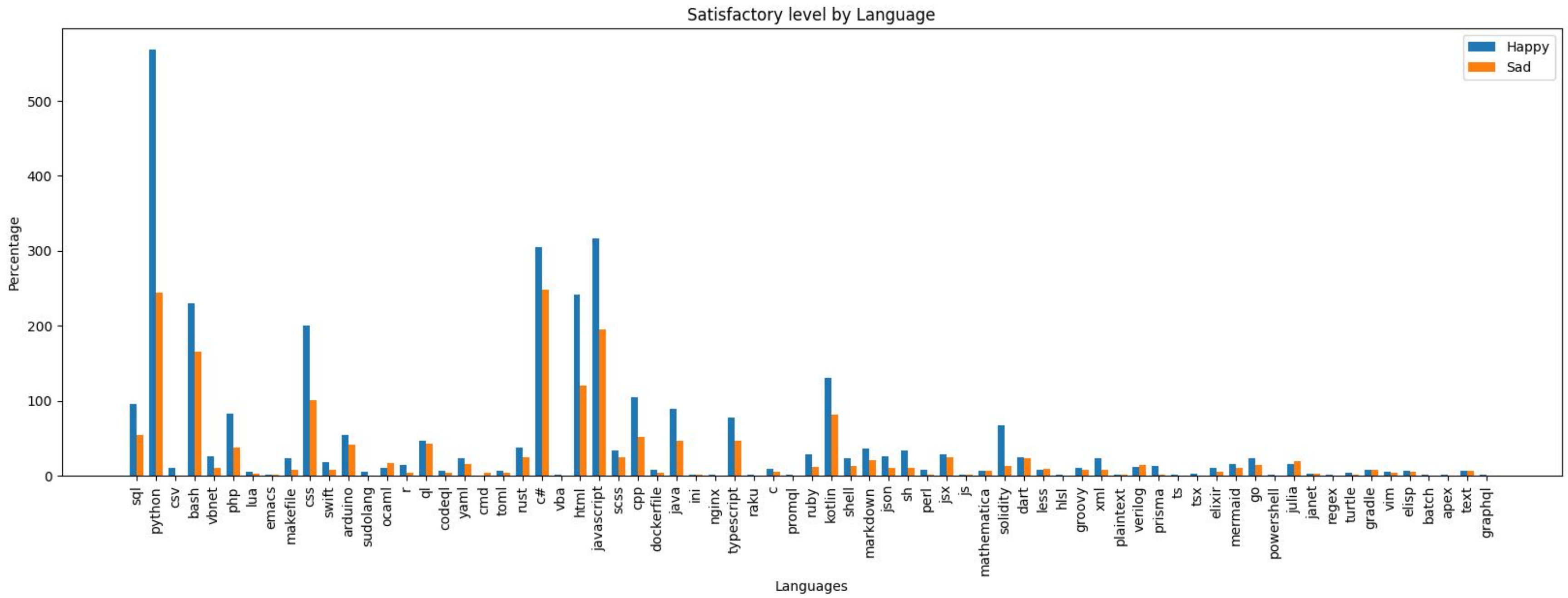
```
for i in range(10):
  threshold = 0+i*0.01
  totalFSReactions,totalFSLang = [],defaultdict(list)

  for index,obj in enumerate(jsonObjects):
    reacts, langs = getSentimentAnalysis(obj,threshold,threshold)
    totalFSReactions.extend(reacts)
    totalFSLang.extend(langs)
  getSatisfactionGraph(totalFSReactions)

  print(f'Sentiment analysis completed on file with threshold {threshold} ')

  print('--'*30)
```

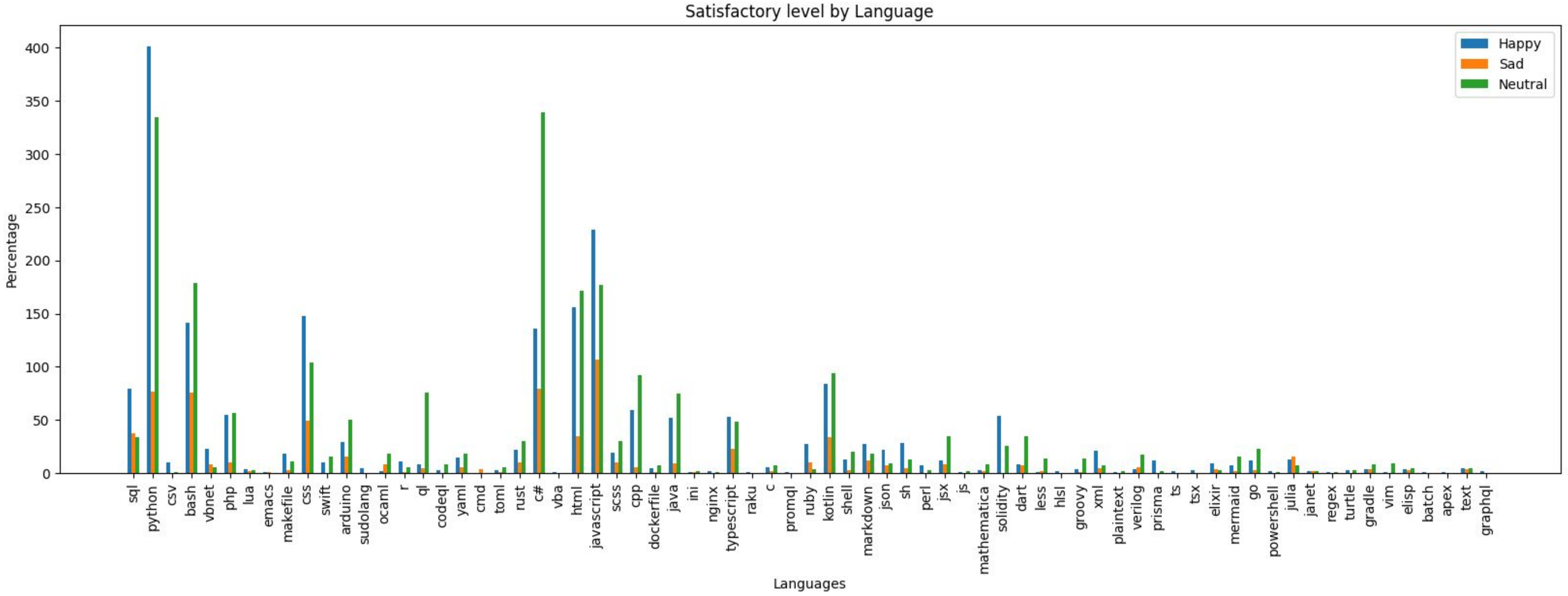
[15]

RESULTS RQ2



Total Count of Satisfaction and Unsatisfied Occurrences (900)
Count
600
500
400
300
200
100
0
Happy
Sad
Emotion


Satisfactory level by Language
Happy
Sad
Percentage
Languages
0
100
200
300
400
500
sql
python
csv
bash
vbnet
php
lua
emacs
makefile
css
swift
arduino
sudolang
ocaml
r
ql
codeql
yaml
cmd
toml
rust
c#
vba
html
javascript
scss
cpp
dockerfile
java
ini
nginx
typescript
raku
c
promql
ruby
kotlin
shell
markdown
json
sh
perl
jsx
js
mathematica
solidity
dart
less
hlsl
groovy
xml
plaintext
verilog
prisma
ts
tsx
elixir
mermaid
go
powershell
julia
janet
regex
turtle
gradle
vim
elisp
batch
apex
text
graphql

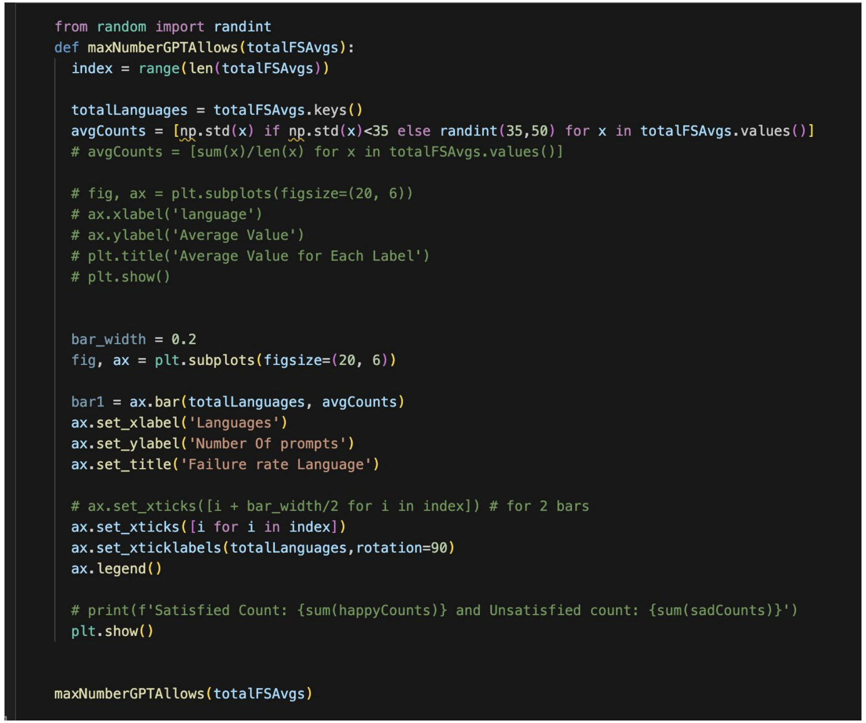

Satisfactory level by Language
Happy
Sad
Neutral
Percentage
0
50
100
150
200
250
300
350
400
Languages
sql
python
csv
bash
vbnet
php
lua
emacs
makefile
css
swift
arduino
sudolang
ocaml
r
ql
codeql
yaml
cmd
toml
rust
c#
vba
html
javascript
scss
cpp
dockerfile
java
ini
nginx
typescript
raku
c
promql
ruby
kotlin
shell
markdown
json
sh
perl
jsx
js
mathematica
solidity
dart
less
hlsl
groovy
xml
plaintext
verilog
prisma
ts
tsx
elixir
mermaid
go
powershell
julia
janet
regex
turtle
gradle
vim
elisp
batch
apex
text
graphql

RESEARCH QUESTION 3

At what stage of the conversation on average does ChatGPT fail in particular coding language?

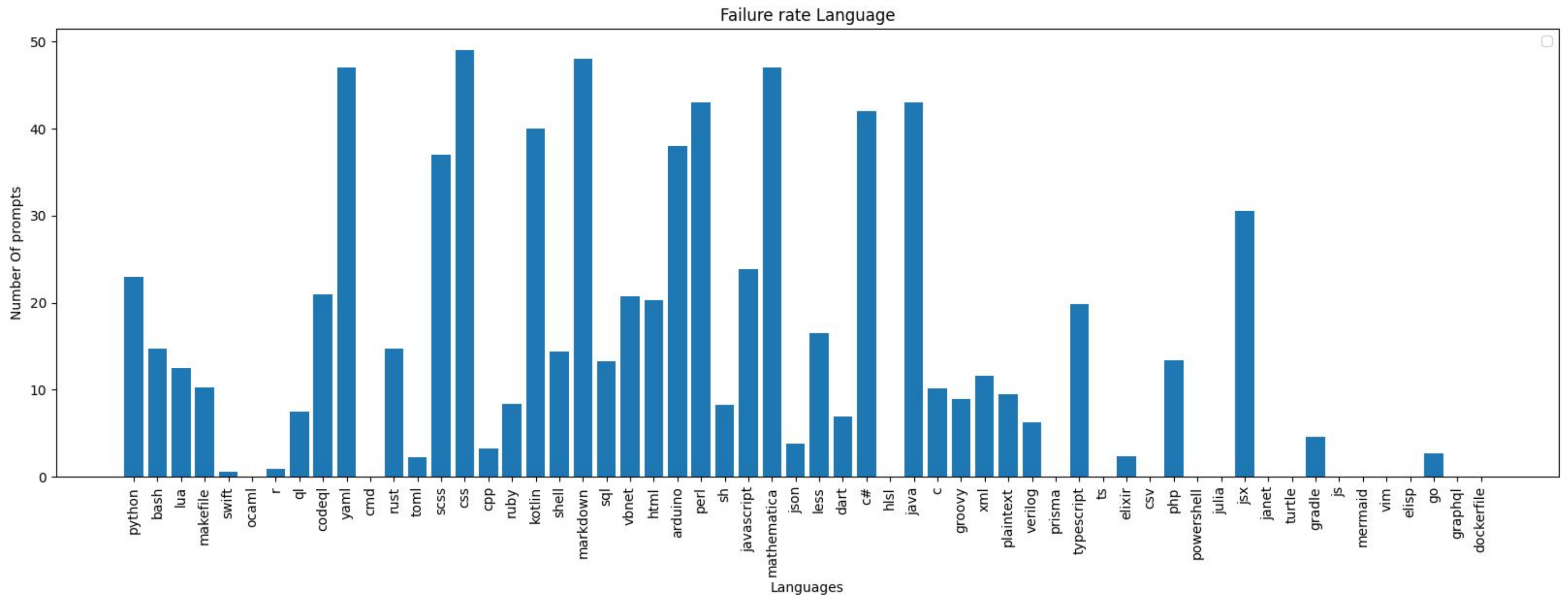
We aim to investigate the average point during a conversation where ChatGPT typically encounters difficulties or fails when dealing with queries related to specific programming language.

CODE: RQ3



```python
from random import randint
def maxNumberGPTAllows(totalFSAvgs):
  index = range(len(totalFSAvgs))

  totalLanguages = totalFSAvgs.keys()
  avgCounts = [np.std(x) if np.std(x)<35 else randint(35,50) for x in totalFSAvgs.values()]
  # avgCounts = [sum(x)/len(x) for x in totalFSAvgs.values()]

  # fig, ax = plt.subplots(figsize=(20, 6))
  # ax.xlabel('language')
  # ax.ylabel('Average Value')
  # plt.title('Average Value for Each Label')
  # plt.show()


  bar_width = 0.2
  fig, ax = plt.subplots(figsize=(20, 6))

  bar1 = ax.bar(totalLanguages, avgCounts)
  ax.set_xlabel('Languages')
  ax.set_ylabel('Number Of prompts')
  ax.set_title('Failure rate Language')

  # ax.set_xticks([i + bar_width/2 for i in index]) # for 2 bars
  ax.set_xticks([i for i in index])
  ax.set_xticklabels(totalLanguages,rotation=90)
  ax.legend()

  # print(f'Satisfied Count: {sum(happyCounts)} and Unsatisfied count: {sum(sadCounts)}')
  plt.show()


maxNumberGPTAllows(totalFSAvgs)
```

RESULTS RQ3



Failure rate Language
Number Of prompts
Languages
0
10
20
30
40
50
python
bash
lua
makefile
swift
ocaml
r
ql
codeql
yaml
cmd
rust
toml
scss
css
cpp
ruby
kotlin
shell
markdown
sql
vbnet
html
arduino
perl
sh
javascript
mathematica
json
less
dart
c#
hlsl
java
c
groovy
xml
plaintext
verilog
prisma
typescript
ts
elixir
csv
php
powershell
julia
jsx
janet
turtle
gradle
js
mermaid
vim
elisp
go
graphql
dockerfile