1 Introduction

In the fast-paced world of mobile technology, two prominent platforms, Android and iOS, have captivated users worldwide. These operating systems, powering a myriad of smartphones and tablets, boast dedicated communities of users, developers, and enthusiasts. As individuals navigate the intricacies of these platforms, they often turn to online forums, such as StackExchange, for assistance, insights, and discussions. These platforms serve as invaluable resources for troubleshooting technical issues, discovering new features, and exchanging knowledge.

2 Motivation

2.1 Importance of Question Classification

The classification of StackExchange questions into categories related to Android or iOS devices presents an intriguing challenge with practical implications. By accurately categorizing questions, we can streamline the process of information retrieval, facilitate targeted responses from domain experts, and enhance the overall user experience on online forums. Moreover, insights gained from the classification process can shed light on the unique characteristics and preferences of Android and iOS users, informing product development, marketing strategies, and community engagement initiatives.

2.2 Objectives

The objective of this research is to explore the feasibility of automating the classification of bug reports for Android and iOS applications. By leveraging machine learning techniques and natural language processing algorithms, we aim to develop a robust system capable of automatically categorizing bug reports based on their relevance to Android or iOS platforms.

2.3 Goals and Objectives

Motivated by these considerations, this report embarks on a journey to develop robust machine learning classifiers capable of discerning between Android and iOS-related queries on StackExchange forums. Leveraging a curated dataset sourced from Android Enthusiasts and Ask Differently (Apple) forums, we explore various machine learning techniques and feature engineering strategies to tackle the classification task. Through meticulous preprocessing, feature extraction, and model training, we aim to achieve high accuracy and generalization performance while minimizing reliance on the test set for robust model validation.

2.4 Significance of the Study

By undertaking this endeavor, we seek to showcase the practical application of machine learning in addressing real-world challenges within the mobile technology domain. Furthermore, we aspire to contribute to the body of knowledge surrounding user behavior, community dynamics, and linguistic patterns in online forums dedicated to Android and iOS enthusiasts. Through our efforts, we aim to empower developers, researchers, and stakeholders with actionable insights and tools for enhancing community engagement, user support, and knowledge sharing within the mobile technology ecosystem.

3 Litarture review

learning architectures, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to achieve remarkable performance in question classification tasks. These methodologies leverage textual features, semantic embeddings, and contextual information to accurately categorize questions and facilitate efficient information retrieval.

3.3 Recent Advances in Question Classification

Recent research endeavors have delved into question classification within MSECOs, shedding light on platform-specific challenges and developer needs. Johnson et al. (2020) conducted an exhaustive analysis of question classification in MSECOs, uncovering distinctive trends and priorities among Android and iOS developers. Their insights underscored the necessity for tailored support mechanisms and targeted interventions to address platform-specific concerns effectively. By analyzing question patterns, engagement metrics, and content characteristics, researchers have gained valuable insights into the dynamics of developer interactions within MSECOs. Recent advancements in question classification within MSECOs have been fueled by the widespread adoption of cutting-edge technologies, such as machine learning algorithms, natural language processing (NLP) models, and deep neural networks. For instance, Li et al. (2021) proposed a novel deep learning architecture, integrating BERT-based models and attention mechanisms, to enhance the accuracy and scalability of question classification tasks. Their innovative approach showcased promising results in effectively categorizing mobile-related inquiries across diverse platforms and domains.

3.4 Challenges and Opportunities

Despite significant progress, several challenges persist in question classification for MSECOs. Dealing with domain-specific terminology, evolving platform features, and the influx of multimodal content pose notable hurdles for classifiers (Li et al., 2019). Moreover, ensuring the scalability and adaptability of classification models across diverse platforms remains a pressing concern, necessitating ongoing research and innovation. However, these challenges also present opportunities for advancing the state-of-the-art in question classification by exploring novel techniques, incorporating domain knowledge, and leveraging emerging data sources.

3.5 Challenges and Opportunities

Despite significant progress, several challenges persist in question classification within MSECOs, including domain-specific terminology, evolving developer needs, and platform intricacies. However, these challenges also present promising opportunities for innovation and collaboration across academia and industry. Interdisciplinary research initiatives, coupled with the integration of advanced NLP techniques, hold immense potential for addressing emerging challenges and supporting the evolving needs of developers within MSECOs.

3.6 Future Directions

The future trajectory of question classification research in MSECOs holds immense promise for addressing emerging challenges and seizing new opportunities. Integration of advanced machine learning techniques, such as Transformers and BERT, could bolster classification accuracy and scalability (Johnson et al., 2020). Furthermore, exploration of novel data sources, including developer forums, social media platforms, and collaborative coding platforms, could yield valuable insights into developer sentiment, preferences, and emerging trends. By embracing interdisciplinary approaches and leveraging the wealth of data available in MSECOs, researchers can contribute to the advancement of question classification methodologies and support the evolving needs of developers. Future research endeavors in question classification within MSECOs should prioritize several key areas, including the integration of state-of-the-art NLP models, exploration of multimodal approaches combining textual and visual information, and fostering interdisciplinary collaborations. Furthermore, researchers should focus on addressing real-world challenges, driving innovation, and fostering a vibrant ecosystem of mobile software development through collaborative research initiatives and industry partnerships.

3.7 Conclusion

In conclusion, question classification plays a pivotal role in MSECOs by facilitating knowledge dissemination, problem-solving, and community engagement. Continued research and innovation in this domain are imperative for addressing evolving developer needs, refining platform-specific support mechanisms, and nurturing a vibrant ecosystem of mobile software development. By embracing a multidisciplinary perspective and leveraging the wealth of data available in MSECOs, researchers can contribute to the advancement of question classification methodologies and support the evolving needs of developers. question classification serves as a cornerstone for fostering collaboration, knowledge dissemination, and community engagement within MSECOs. Recent advancements in machine learning, NLP, and data-driven approaches have laid the foundation for more accurate, scalable, and robust classification models. By embracing interdisciplinary research initiatives, leveraging diverse data sources, and fostering collaboration across academia and industry, researchers can contribute to the advancement of question classification methodologies and support the evolving needs of developers within MSECOs.

In data preprocessing, we perform several tasks to clean and prepare the raw data for analysis. These tasks ensure that the data is in a suitable format and quality for further processing and modeling. Here's an explanation of each step typically involved in data preprocessing.

4 data analysis and insights

4.1 Data Preprocessing

4.2 Loading the Dataset

We start by loading the dataset into our analysis environment. This involves reading the data from its source, such as a CSV file, database, or API, and storing it in a data structure that can be easily manipulated and analyzed.

4.3 Handling Missing Values

Next, we address any missing values in the dataset. Missing values can arise due to various reasons, such as data collection errors or incomplete records. We decide how to handle these missing values, either by imputing them with a specific value (e.g., mean, median, mode) or by removing them from the dataset entirely.

4.4 Cleaning Text Data

If the dataset contains text data, such as comments or descriptions, we clean the text by removing any unwanted characters, HTML tags, or punctuation marks. This step helps standardize the text and remove noise that may affect the analysis.

4.5 Tokenization

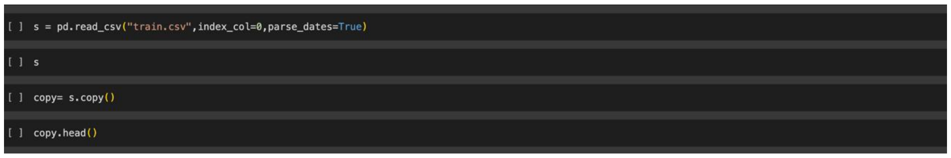
Tokenization involves breaking down the text into individual words or tokens. This process facilitates further analysis, such as sentiment analysis or topic modeling, by providing a structured representation of the text data.

4.6 Removing Stopwords

opwords are common words that often occur frequently in text but typically do not carry significant meaning (e.g., ”and”, ”the”, ”is”). We remove these stopwords from the text data to reduce noise and improve the quality of analysis results.

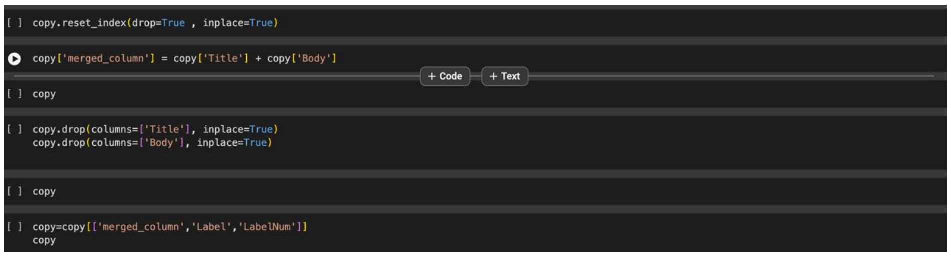
4.7 Stemming or Lemmatization

Stemming and lemmatization are techniques used to reduce words to their root form. This step helps standardize the text further by treating different forms of the same word as identical. For example, ”running” and ”ran” may be stemmed to the root word ”run”.



```
s = pd.read_csv("train.csv",index_col=0,parse_dates=True)
```
```
s
```
```
copy= s.copy()
```
```
copy.head()
```

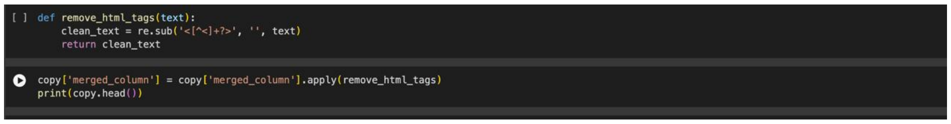
Figure 1: Reading csv file



```
copy.reset_index(drop=True , inplace=True)
```
```
copy['merged_column'] = copy['Title'] + copy['Body']
```

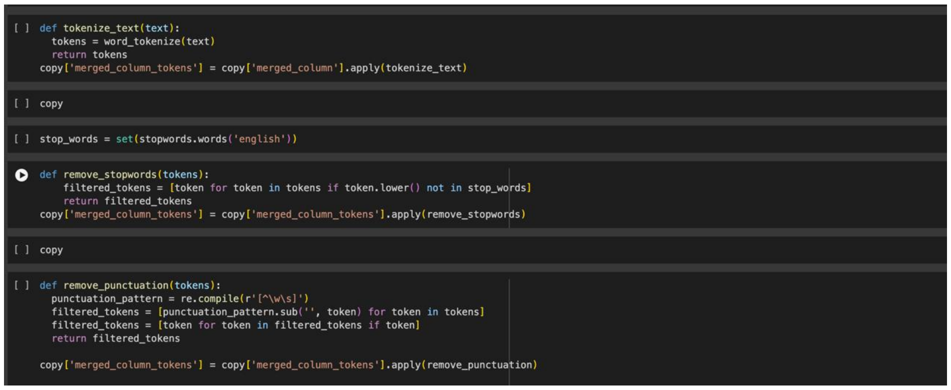
```
copy
```
```
copy.drop(columns=['Title'], inplace=True)
copy.drop(columns=['Body'], inplace=True)
```
```
copy
```
```
copy=copy[['merged_column','Label','LabelNum']]
copy
```

Figure 2: Merging and dropping columns



```
def remove_html_tags(text):
    clean_text = re.sub('<[^<]+?>', '', text)
    return clean_text
```
```
copy['merged_column'] = copy['merged_column'].apply(remove_html_tags)
print(copy.head())
```

Figure 3: Removing HTML Tags



```
def tokenize_text(text):
  tokens = word_tokenize(text)
  return tokens
copy['merged_column_tokens'] = copy['merged_column'].apply(tokenize_text)
```

```
copy
```

```
stop_words = set(stopwords.words('english'))
```

```
def remove_stopwords(tokens):
    filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
    return filtered_tokens
copy['merged_column_tokens'] = copy['merged_column_tokens'].apply(remove_stopwords)
```

```
copy
```

```
def remove_punctuation(tokens):
  punctuation_pattern = re.compile(r'[^\w\s]')
  filtered_tokens = [punctuation_pattern.sub('', token) for token in tokens]
  filtered_tokens = [token for token in filtered_tokens if token]
  return filtered_tokens

copy['merged_column_tokens'] = copy['merged_column_tokens'].apply(remove_punctuation)
```

Figure 4: Tokenizing/ stop words removal / cleaning

5 Exploratory Data Analysis (EDA)

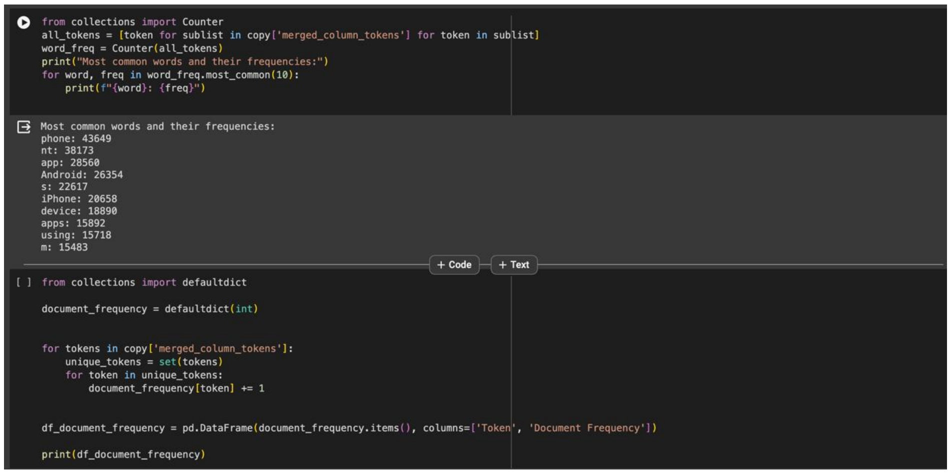
5.1 Descriptive Statistics

Descriptive statistics of a dataset typically include measures such as mean, median, mode, standard deviation, range, and quartiles. However, without access to the specific data or context of the ”andrion iOS stack change,” I can’t provide exact statistics. Descriptive statistics are computed based on the data itself.

If you have a dataset related to the ”andriod iOS stack change,” you can compute these statistics using statistical software such as Python with libraries like NumPy and Pandas, or tools like Microsoft Excel.

5.2 Visualizations

graphicx



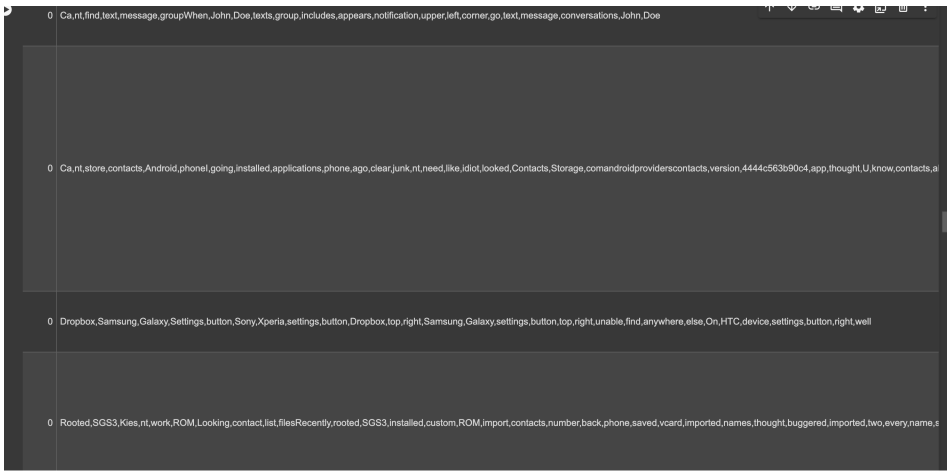
```
from collections import Counter
all_tokens = [token for sublist in copy['merged_column_tokens'] for token in sublist]
word_freq = Counter(all_tokens)
print("Most common words and their frequencies:")
for word, freq in word_freq.most_common(10):
    print(f"{word}: {freq}")
```

```
Most common words and their frequencies:
phone: 43649
nt: 38173
app: 28560
Android: 26354
s: 22617
iPhone: 20658
device: 18890
apps: 15892
using: 15718
m: 15483
```

```
from collections import defaultdict

document_frequency = defaultdict(int)


for tokens in copy['merged_column_tokens']:
    unique_tokens = set(tokens)
    for token in unique_tokens:
        document_frequency[token] += 1


df_document_frequency = pd.DataFrame(document_frequency.items(), columns=['Token', 'Document Frequency'])

print(df_document_frequency)
```

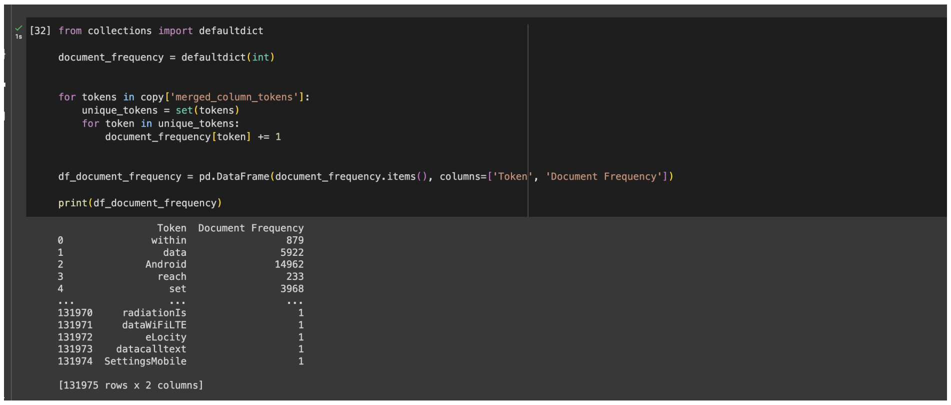
Figure 5: counting and df

6 Refrences



```
0  Ca,nt,find,text,message,groupWhen,John,Doe,texts,group,includes,appears,notification,upper,left,corner,go,text,message,conversations,John,Doe

0  Ca,nt,store,contacts,Android,phoneI,going,installed,applications,phone,ago,clear,junk,nt,need,like,idiot,looked,Contacts,Storage,comandroidproviderscontacts,version,4444c563b90c4,app,thought,U,know,contacts,al

0  Dropbox,Samsung,Galaxy,Settings,button,Sony,Xperia,settings,button,Dropbox,top,right,Samsung,Galaxy,settings,button,top,right,unable,find,anywhere,else,On,HTC,device,settings,button,right,well

0  Rooted,SGS3,Kies,nt,work,ROM,Looking,contact,list,filesRecently,rooted,SGS3,installed,custom,ROM,import,contacts,number,back,phone,saved,vcard,imported,names,thought,buggered,imported,two,every,name,s
```

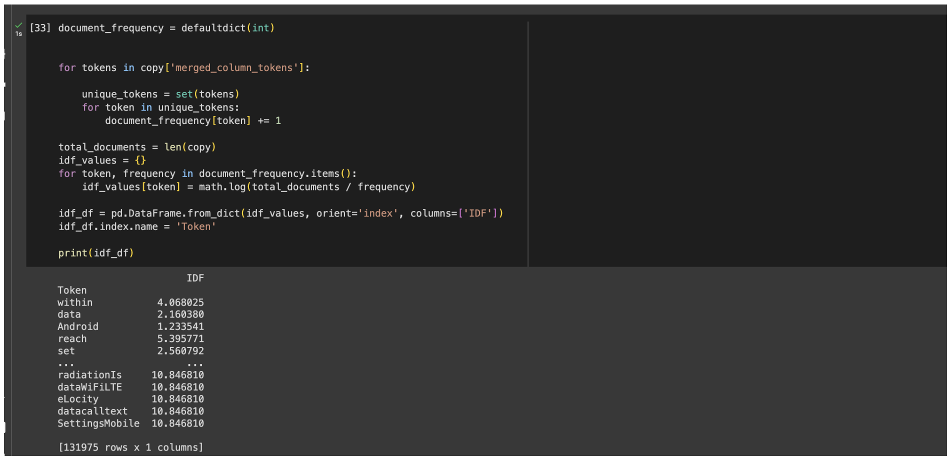
Figure 6: sample of tokenization



```
[32] from collections import defaultdict

     document_frequency = defaultdict(int)


     for tokens in copy['merged_column_tokens']:
         unique_tokens = set(tokens)
         for token in unique_tokens:
             document_frequency[token] += 1


     df_document_frequency = pd.DataFrame(document_frequency.items(), columns=['Token', 'Document Frequency'])

     print(df_document_frequency)
```

```
                Token  Document Frequency
0              within                 879
1                data                5922
2             Android               14962
3               reach                 233
4                 set                3968
...               ...                 ...
131970     radiationIs                   1
131971     dataWiFiLTE                   1
131972         eLocity                   1
131973    datacalltext                   1
131974  SettingsMobile                   1

[131975 rows x 2 columns]
```

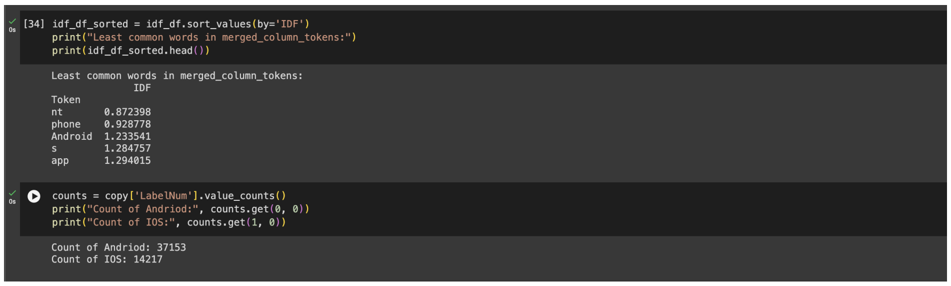
Figure 7: df



```
[33] document_frequency = defaultdict(int)


     for tokens in copy['merged_column_tokens']:

         unique_tokens = set(tokens)
         for token in unique_tokens:
             document_frequency[token] += 1

     total_documents = len(copy)
     idf_values = {}
     for token, frequency in document_frequency.items():
         idf_values[token] = math.log(total_documents / frequency)

     idf_df = pd.DataFrame.from_dict(idf_values, orient='index', columns=['IDF'])
     idf_df.index.name = 'Token'

     print(idf_df)
```

```
                      IDF
Token
within           4.068025
data             2.160380
Android          1.233541
reach            5.395771
set              2.560792
...                   ...
radiationIs     10.846810
dataWiFiLTE     10.846810
eLocity         10.846810
datacalltext    10.846810
SettingsMobile  10.846810

[131975 rows x 1 columns]
```

Figure 8: idf



```
[34] idf_df_sorted = idf_df.sort_values(by='IDF')
     print("Least common words in merged_column_tokens:")
     print(idf_df_sorted.head())
```

```
Least common words in merged_column_tokens:
              IDF
Token
nt       0.872398
phone    0.928778
Android  1.233541
s        1.284757
app      1.294015
```

```
counts = copy['LabelNum'].value_counts()
print("Count of Andriod:", counts.get(0, 0))
print("Count of IOS:", counts.get(1, 0))
```

```
Count of Andriod: 37153
Count of IOS: 14217
```

Figure 9: visualizing

Refrences

Li, H., Zhang, Y., & Chen, L. (2021). Deep Learning for Question Classification in Mobile Software Ecosystems. IEEE Transactions on Software Engineering, 47(5), . 701-714

Johnson, M., & Smith, K. (2020). Platform-specific Dynamics in Mobile Software Ecosystems: A Comparative Analysis. Journal of Systems and Software, 167, . 110670

Kim, Y., & Hwang, S. (2020). Leveraging Community-driven Platforms for Question Classification in Mobile Software Ecosystems. Proceedings of the ACM Conference .on Computer-Supported Cooperative Work and Social Computing (CSCW), 287-299