(Model Training)INM434 Natural Language Processing_Yumi Heo_code

May 8, 2024

INM434 Natural Language Processing MSc Data Science | Yumi Heo | 230003122

Google Colab Folder Link: https://drive.google.com/drive/folders/1Yn99YR6d5iJ79NYjdZwLLUTEPmucnNqH?u

Model Training Code Google Colab Link:https://colab.research.google.com/drive/1nTohqQt6vPr6GIj6mX9jmtQia

Model Test Code Google Colab Link: https://colab.research.google.com/drive/1OgBrPqldusG9K2o68J3SBCF-pTexFih0?usp=sharing

1 Intent Classification for Bank Customer Queries

1.1 Import the dataset 'banking77'

[ ]:
```
# Mount the drive to save and load files and models
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
# Install datasets from huggingface
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-2.19.1-py3-none-any.whl (542 kB)
                                 542.0/542.0

kB 8.1 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from datasets) (3.14.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from datasets) (1.25.2)
Requirement already satisfied: pyarrow>=12.0.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (14.0.2)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-
packages (from datasets) (0.6)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
```

```
                           116.3/116.3

kB 12.5 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-
packages (from datasets) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-
packages (from datasets) (4.66.4)
Collecting xxhash (from datasets)
  Downloading
xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                           194.1/194.1

kB 9.8 MB/s eta 0:00:00
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
                           134.8/134.8

kB 14.0 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<=2024.3.1,>=2023.1.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2023.6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from datasets) (3.9.5)
Collecting huggingface-hub>=0.21.2 (from datasets)
  Downloading huggingface_hub-0.23.0-py3-none-any.whl (401 kB)
                           401.2/401.2

kB 12.4 MB/s eta 0:00:00
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from datasets) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from datasets) (6.0.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.21.2->datasets)
(4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
```

```
/usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.19.0->datasets) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
(2024.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.16.0)
Installing collected packages: xxhash, dill, multiprocess, huggingface-hub,
datasets
  Attempting uninstall: huggingface-hub
    Found existing installation: huggingface-hub 0.20.3
    Uninstalling huggingface-hub-0.20.3:
      Successfully uninstalled huggingface-hub-0.20.3
Successfully installed datasets-2.19.1 dill-0.3.8 huggingface-hub-0.23.0
multiprocess-0.70.16 xxhash-3.4.1
```

[ ]:
```python
# Import libraries
import datasets
from datasets import load_dataset
import numpy as np
import matplotlib.pyplot as plt
import re
import string
import nltk
from nltk.corpus import stopwords
from pprint import pprint
from collections import Counter
from sklearn.metrics import precision_score, recall_score, f1_score
```

[ ]:
```python
# Take the dataset 'banking77'
banking77 = load_dataset('banking77')
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
```

```
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(

Downloading readme:   0%|          | 0.00/14.4k [00:00<?, ?B/s]

Downloading data:   0%|          | 0.00/298k [00:00<?, ?B/s]

Downloading data:   0%|          | 0.00/93.9k [00:00<?, ?B/s]

Generating train split:   0%|          | 0/10003 [00:00<?, ? examples/s]

Generating test split:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Check its structure
banking77
```

[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 10003
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 3080
    })
})
```

[ ]:
```python
# Check the data format of the dataset
banking77.cache_files

# Arrow type
```

[ ]:
```
{'train': [{'filename': '/root/.cache/huggingface/datasets/banking77/default/0.0
.0/f54121560de48f2852f90be299010d1d6dc612ec/banking77-train.arrow'}],
 'test': [{'filename': '/root/.cache/huggingface/datasets/banking77/default/0.0.
0/f54121560de48f2852f90be299010d1d6dc612ec/banking77-test.arrow'}]}
```

1.2 Data Preprocessing: Remove numbers, punctuations, double white spaces and apply lowercase

[ ]:
```python
# Define the function to remove numbers
def remove_num(example):
    return {'text': re.sub(r'\d+', '', example['text'])}
```

[ ]:
```python
# Mapping the lowercase function
banking77_wonum = banking77.map(remove_num)
```

```
Map:   0%|          | 0/10003 [00:00<?, ? examples/s]

Map:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Apply lowercase to all texts in the dataset
# Define the function
def lowercase(example):
  return {'text': example['text'].lower()}
```

[ ]:
```python
# Mapping the lowercase function
banking77_lowercase = banking77_wonum.map(lowercase)
```

```
Map:   0%|          | 0/10003 [00:00<?, ? examples/s]

Map:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Remove punctuations in the dataset
# Check the punctuation
string.punctuation
```

[ ]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

[ ]:
```python
# Define the removing puctuations function
def remove_punctuations(example):
  return {'text': example['text'].translate(str.maketrans('', '', string.
↪punctuation))}
```

[ ]:
```python
# Mapping the function
banking77_wopunc = banking77_lowercase.map(remove_punctuations)
```

```
Map:   0%|          | 0/10003 [00:00<?, ? examples/s]

Map:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Define the removing double white spaces function
def remove_doublespaces(example):
  return {'text': re.sub('\s+', ' ', example['text']).strip()}
```

[ ]:
```python
# Mapping the function
banking77_wods = banking77_wopunc.map(remove_doublespaces)
```

```
Map:   0%|          | 0/10003 [00:00<?, ? examples/s]

Map:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Take stopwords in English version
nltk.download('stopwords')
eng_stop = set(stopwords.words("english"))

# Define the removing double white spaces function
```

```python
def remove_stopwords(example):
    words = example['text'].split()  # Split the text into words
    filtered_words = [word for word in words if word.lower() not in eng_stop] 
 ↪# Remove stopwords
    return {'text': ' '.join(filtered_words)}  # Join the filtered words back
 ↪into a single string
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[ ]:
```python
# Mapping the function
banking77_preprocessed = banking77_wods.map(remove_stopwords)
```

```
Map:   0%|          | 0/10003 [00:00<?, ? examples/s]

Map:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Extract training set and test set
trainset = banking77_preprocessed['train']
testset = banking77_preprocessed['test']
```

[ ]:
```python
# Check columns in the traininig set
trainset.column_names
```

[ ]: ['text', 'label']

[ ]:
```python
# Check the featrues in training set
print(trainset.features)
```

```
{'text': Value(dtype='string', id=None), 'label':
ClassLabel(names=['activate_my_card', 'age_limit', 'apple_pay_or_google_pay',
'atm_support', 'automatic_top_up', 'balance_not_updated_after_bank_transfer',
'balance_not_updated_after_cheque_or_cash_deposit', 'beneficiary_not_allowed',
'cancel_transfer', 'card_about_to_expire', 'card_acceptance', 'card_arrival',
'card_delivery_estimate', 'card_linking', 'card_not_working',
'card_payment_fee_charged', 'card_payment_not_recognised',
'card_payment_wrong_exchange_rate', 'card_swallowed', 'cash_withdrawal_charge',
'cash_withdrawal_not_recognised', 'change_pin', 'compromised_card',
'contactless_not_working', 'country_support', 'declined_card_payment',
'declined_cash_withdrawal', 'declined_transfer',
'direct_debit_payment_not_recognised', 'disposable_card_limits',
'edit_personal_details', 'exchange_charge', 'exchange_rate', 'exchange_via_app',
'extra_charge_on_statement', 'failed_transfer', 'fiat_currency_support',
'get_disposable_virtual_card', 'get_physical_card', 'getting_spare_card',
'getting_virtual_card', 'lost_or_stolen_card', 'lost_or_stolen_phone',
'order_physical_card', 'passcode_forgotten', 'pending_card_payment',
'pending_cash_withdrawal', 'pending_top_up', 'pending_transfer', 'pin_blocked',
'receiving_money', 'Refund_not_showing_up', 'request_refund',
```

```
'reverted_card_payment?', 'supported_cards_and_currencies', 'terminate_account',
'top_up_by_bank_transfer_charge', 'top_up_by_card_charge',
'top_up_by_cash_or_cheque', 'top_up_failed', 'top_up_limits', 'top_up_reverted',
'topping_up_by_card', 'transaction_charged_twice', 'transfer_fee_charged',
'transfer_into_account', 'transfer_not_received_by_recipient',
'transfer_timing', 'unable_to_verify_identity', 'verify_my_identity',
'verify_source_of_funds', 'verify_top_up', 'virtual_card_not_working',
'visa_or_mastercard', 'why_verify_identity', 'wrong_amount_of_cash_received',
'wrong_exchange_rate_for_cash_withdrawal'], id=None)}
```

[ ]:
```python
# Check the first five elements in the training set
pprint(trainset[:5], sort_dicts=False)
pprint(testset[:5], sort_dicts=False)
```

```
{'text': ['still waiting card',
          'card still hasnt arrived weeks',
          'waiting week card still coming',
          'track card process delivery',
          'know get card lost'],
 'label': [11, 11, 11, 11, 11]}
{'text': ['locate card',
          'still received new card ordered week ago',
          'ordered card arrived help please',
          'way know card arrive',
          'card arrived yet'],
 'label': [11, 11, 11, 11, 11]}
```

All preprocessing steps have been applied.

[ ]:
```python
# Describe general information of the training set
pprint(trainset.info)
```

```
DatasetInfo(description='',
            citation='',
            homepage='',
            license='',
            features={'label': ClassLabel(names=['activate_my_card',
                                                 'age_limit',
                                                 'apple_pay_or_google_pay',
                                                 'atm_support',
                                                 'automatic_top_up',
'balance_not_updated_after_bank_transfer',
'balance_not_updated_after_cheque_or_cash_deposit',
                                                 'beneficiary_not_allowed',
                                                 'cancel_transfer',
                                                 'card_about_to_expire',
                                                 'card_acceptance',
                                                 'card_arrival',
                                                 'card_delivery_estimate',
```

```
'card_linking',
'card_not_working',
'card_payment_fee_charged',
'card_payment_not_recognised',
'card_payment_wrong_exchange_rate',
'card_swallowed',
'cash_withdrawal_charge',
'cash_withdrawal_not_recognised',
'change_pin',
'compromised_card',
'contactless_not_working',
'country_support',
'declined_card_payment',
'declined_cash_withdrawal',
'declined_transfer',
'direct_debit_payment_not_recognised',
'disposable_card_limits',
'edit_personal_details',
'exchange_charge',
'exchange_rate',
'exchange_via_app',
'extra_charge_on_statement',
'failed_transfer',
'fiat_currency_support',
'get_disposable_virtual_card',
'get_physical_card',
'getting_spare_card',
'getting_virtual_card',
'lost_or_stolen_card',
'lost_or_stolen_phone',
'order_physical_card',
'passcode_forgotten',
'pending_card_payment',
'pending_cash_withdrawal',
'pending_top_up',
'pending_transfer',
'pin_blocked',
'receiving_money',
'Refund_not_showing_up',
'request_refund',
'reverted_card_payment?',
'supported_cards_and_currencies',
'terminate_account',
'top_up_by_bank_transfer_charge',
'top_up_by_card_charge',
'top_up_by_cash_or_cheque',
'top_up_failed',
'top_up_limits',
```

```
                                                  'top_up_reverted',
                                                  'topping_up_by_card',
                                                  'transaction_charged_twice',
                                                  'transfer_fee_charged',
                                                  'transfer_into_account',
'transfer_not_received_by_recipient',
                                                  'transfer_timing',
                                                  'unable_to_verify_identity',
                                                  'verify_my_identity',
                                                  'verify_source_of_funds',
                                                  'verify_top_up',
                                                  'virtual_card_not_working',
                                                  'visa_or_mastercard',
                                                  'why_verify_identity',
'wrong_amount_of_cash_received',
'wrong_exchange_rate_for_cash_withdrawal'],
                                          id=None),
                      'text': Value(dtype='string', id=None)},
            post_processed=None,
            supervised_keys=None,
            task_templates=None,
            builder_name='parquet',
            dataset_name='banking77',
            config_name='default',
            version=0.0.0,
            splits={'test': SplitInfo(name='test',
                                      num_bytes=204395,
                                      num_examples=3080,
                                      shard_lengths=None,
                                      dataset_name='banking77'),
                    'train': SplitInfo(name='train',
                                       num_bytes=716279,
                                       num_examples=10003,
                                       shard_lengths=None,
                                       dataset_name='banking77')},
            download_checksums={'hf://datasets/banking77@f54121560de48f2852f90be
299010d1d6dc612ec/data/test-00000-of-00001.parquet': {'checksum': None,
                                                       'num_bytes': 93870},
                                'hf://datasets/banking77@f54121560de48f2852f90be
299010d1d6dc612ec/data/train-00000-of-00001.parquet': {'checksum': None,
                                                        'num_bytes': 298170}},
            download_size=392040,
            post_processing_size=None,
            dataset_size=920674,
            size_in_bytes=1312714)
```

[ ]:
```python
# Change the format as dataframe
banking77_preprocessed.set_format(type='pandas')
```

[ ]:
```python
# Define a dataframe for the training set
train_df = banking77_preprocessed['train'][:]
```

[ ]:
```python
# Check the dataframe of the training set
train_df
```

[ ]:
```
                                    text  label
0                     still waiting card     11
1       card still hasnt arrived weeks     11
2       waiting week card still coming     11
3          track card process delivery     11
4                     know get card lost     11
…                                    …      …
9998          provide support countries     24
9999               countries supporting     24
10000         countries getting support     24
10001                cards available eu     24
10002             countries represented     24

[10003 rows x 2 columns]
```

[ ]:
```python
# Check if there is null data
train_df.isnull().sum()
```

[ ]:
```
text     0
label    0
dtype: int64
```

[ ]:
```python
train_df
```

[ ]:

```
                                    text  label
0                     still waiting card     11
1       card still hasnt arrived weeks     11
2       waiting week card still coming     11
3          track card process delivery     11
4                     know get card lost     11
…                                    …      …
9998          provide support countries     24
9999               countries supporting     24
10000         countries getting support     24
10001                cards available eu     24
10002             countries represented     24

[10003 rows x 2 columns]
```

[ ]:
```
# Check if there is the same text after text preprocessing
# Group by text and label, then count occurrences
train_df['count'] = train_df.groupby(['text', 'label'])['text'].
    ↪transform('count')

# Filter rows where count > 1 (duplicates)
duplicates_df = train_df[train_df['count'] > 1]

# Sort it by text to group duplicates together
duplicates_df = duplicates_df.sort_values(by='text')
```

[ ]:
```
duplicates_df[['text', 'label']]
```

[ ]:
```
                          text  label
5028            able get cash atm     26
5011            able get cash atm     26
4178    able get visa mastercard     73
4135    able get visa mastercard     73
5796               able see refund     51
…                            …    …
5460   would like cancel purchase     52
31     would like track card sent     11
98     would like track card sent     11
4949       wouldnt atm give money     26
5024       wouldnt atm give money     26

[1298 rows x 2 columns]
```

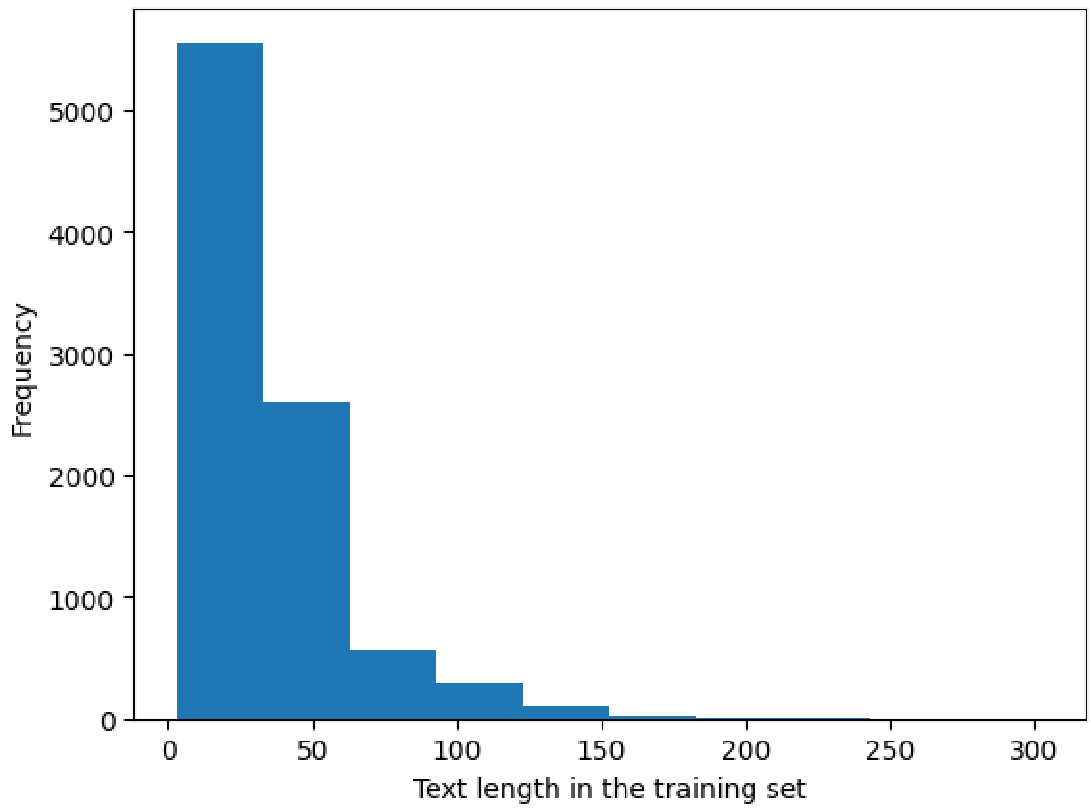
[ ]:
```
# Remove duplicates based on text, keeping the first one
train_df = train_df.drop_duplicates(subset='text', keep='first')
```

[ ]:
```
train_df
```

[ ]:
```
                                 text  label  count
0                  still waiting card     11      2
1       card still hasnt arrived weeks     11      1
2       waiting week card still coming     11      1
3          track card process delivery     11      1
4                  know get card lost     11      1
…                                 …     …      …
9997                moved us get card     24      1
9998        provide support countries     24      1
9999             countries supporting     24      1
10000        countries getting support     24      1
10002          countries represented     24      1

[9183 rows x 3 columns]
```

```
[ ]: # Check the text length in the training set
lengths = np.array([len(text) for text in train_df['text']])
print(f'The average is {np.mean(lengths)}. The median is {np.median(lengths)}.␣
  ↪The max length is {np.max(lengths)}.')

# Plot the histogram
plt.hist(lengths)
plt.xlabel('Text length in the training set')
plt.ylabel('Frequency')
plt.show()
```

```
The average is 36.17641293694871. The median is 29.0. The max length is 303.
```



Since the median is 29, a length of 29 will be used. The rest will be cut or filled in with padding.

```
[ ]: # Define a dataframe for the test set
test_df = banking77_preprocessed['test'][:]
```

```
[ ]: # Check the dataframe of the test set
test_df
```

```
[ ]:                                           text  label
     0                                   locate card     11
     1      still received new card ordered week ago     11
     2              ordered card arrived help please     11
     3                          way know card arrive     11
     4                              card arrived yet     11
     …                                             …      …
     3075                      im uk still get card     24
     3076                    many countries support     24
     3077                        countries business     24
     3078                         countries operate     24
     3079                   card mailed used europe     24

     [3080 rows x 2 columns]
```

```
[ ]: # Check if there is null data
     test_df.isnull().sum()
```

```
[ ]: text     0
     label    0
     dtype: int64
```

```
[ ]: # Check the data type of elements in the dataframes
     print(train_df['text'].dtype)
     print(test_df['text'].dtype)
     print(train_df['label'].dtype)
     print(test_df['label'].dtype)
```

```
object
object
int64
int64
```

```
[ ]: # Check the number of labels in the training set
     train_df['label'].nunique()
```

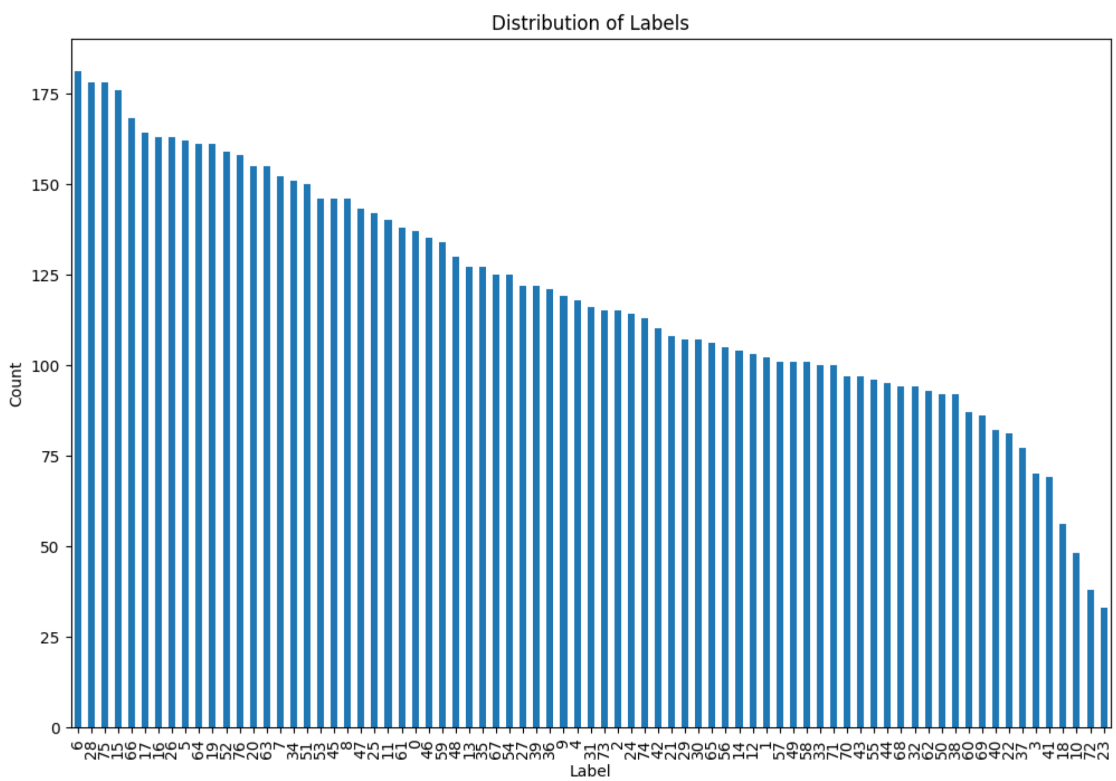
```
[ ]: 77
```

```
[ ]: # Check the balance of labels in the training set
     train_df['label'].value_counts()
```

```
[ ]: label
     6      181
     28     178
     75     178
     15     176
     66     168
```

```
      …
41     69
18     56
10     48
72     38
23     33
Name: count, Length: 77, dtype: int64
```

[ ]:
```python
# Plot histogram for labels in the training set
plt.figure(figsize=(12, 8))
train_df['label'].value_counts().plot(kind='bar')
plt.xlabel('Label')
plt.ylabel('Count')
plt.title('Distribution of Labels')
plt.xticks(rotation=90)
plt.show()
```



The class in the training set is imbalanced.

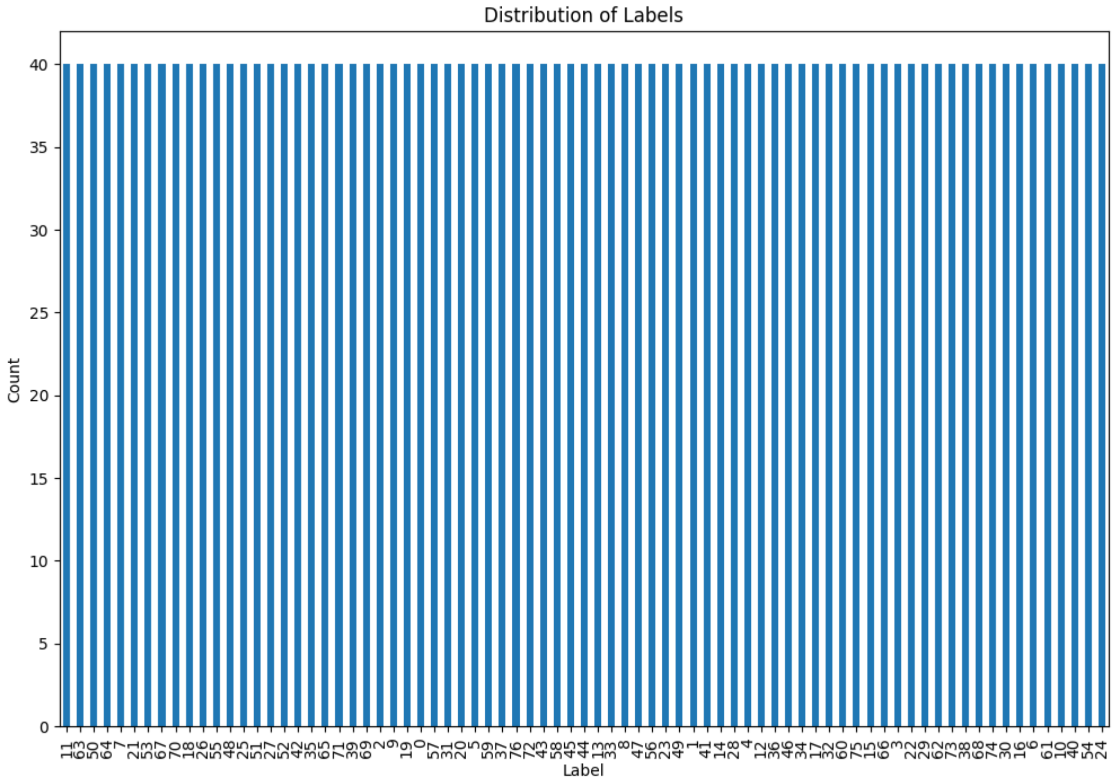
[ ]:
```python
# Check the number of labels in the test set
test_df['label'].nunique()
```

[ ]: 77

[ ]:
```python
# Check the balance of labels in the test set
test_df['label'].value_counts()
```

[ ]:
```
label
11    40
63    40
50    40
64    40
7     40
      ..
61    40
10    40
40    40
54    40
24    40
Name: count, Length: 77, dtype: int64
```

[ ]:
```python
# Plot histogram for labels in the test set
plt.figure(figsize=(12, 8))
test_df['label'].value_counts().plot(kind='bar')
plt.xlabel('Label')
plt.ylabel('Count')
plt.title('Distribution of Labels')
plt.xticks(rotation=90)
plt.show()
```



The class in the test set is balanced.

1.3 Tokenize the text in the dataset

1.3.1 For LSTM Model

[ ]:
```
# Seperate the training set to a training set and a validation set
# Import the library
from sklearn.model_selection import train_test_split

# Split the data to X and y
X = train_df['text']
y = train_df['label']

# Split the training set (80% for the training set, 20% for the validation set)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,␣
↪stratify=train_df['label']) # Balance the class
```

[ ]:
```
# Reset the index
X_train.reset_index(drop=True, inplace=True)
X_val.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)
y_val.reset_index(drop=True, inplace=True)
```

[ ]:
```
# Define X_test and y_test from the test set
X_test = test_df['text']
y_test = test_df['label']
```

[ ]:
```
# Count the words to find the vocabulary size in the training set
# Define the function to cumulate the number of words
def count_word(X_train):
    count = Counter()
    for text in X_train.values:
        for word in text.split():
            count[word] += 1
    return count

# Count the number of words
word_counts = count_word(X_train)

print(f'The vocabulary size in the training set is {len(word_counts)}')
```

```
The vocabulary size in the training set is 2088
```

[ ]:
```
# Check the inside of the vocabulary
word_counts
```

[ ]:
```
Counter({'transaction': 256,
         'reverted': 49,
         'card': 1942,
         'charged': 401,
         'payment': 551,
         'made': 285,
         'cash': 518,
         'withdrawal': 212,
         'atm': 353,
         'listed': 12,
         'dont': 259,
         'remember': 28,
         'making': 40,
         'unauthorized': 11,
         'direct': 76,
         'debit': 106,
         'account': 1059,
         'please': 394,
         'explain': 61,
         'fee': 333,
         'transfer': 807,
         'verify': 95,
         'new': 256,
         'tried': 199,
```

'sending': 13,
'standard': 9,
'five': 5,
'times': 102,
'hasnt': 134,
'gone': 66,
'problem': 69,
'get': 606,
'refund': 213,
'purchase': 112,
'cost': 57,
'time': 122,
'frame': 9,
'getting': 122,
'identity': 152,
'verification': 85,
'isnt': 130,
'working': 145,
'could': 96,
'google': 41,
'pay': 133,
'top': 436,
'together': 2,
'expires': 22,
'soon': 39,
'fast': 13,
'replacement': 10,
'sent': 107,
'costs': 7,
'accidentally': 9,
'chose': 3,
'exchange': 411,
'gbp': 47,
'need': 544,
'pick': 3,
'aud': 16,
'change': 151,
'documents': 12,
'validate': 1,
'fiat': 26,
'currencies': 147,
'supported': 25,
'holding': 19,
'decline': 17,
'incoming': 1,
'next': 13,
'maybe': 8,

'let': 60,
'people': 16,
'know': 230,
'buying': 9,
'things': 21,
'another': 79,
'country': 48,
'extra': 205,
'used': 97,
'item': 72,
'also': 17,
'looks': 26,
'like': 214,
'wasnt': 49,
'aware': 11,
'think': 127,
'updated': 24,
'depositing': 7,
'cheque': 84,
'balance': 89,
'multiple': 44,
'one': 206,
'received': 181,
'incorrect': 43,
'amount': 161,
'today': 76,
'wanted': 42,
'virtual': 161,
'located': 8,
'yesterday': 52,
'topped': 24,
'didnt': 257,
'complete': 50,
'still': 273,
'pending': 304,
'processed': 14,
'cards': 214,
'shipped': 5,
'recently': 39,
'would': 274,
'cancel': 102,
'longer': 26,
'want': 213,
'us': 67,
'arrive': 24,
'assist': 19,
'completing': 4,

'im': 225,
'trying': 72,
'keep': 59,
'error': 43,
'flat': 12,
'initial': 1,
'mortgage': 7,
'find': 104,
'whats': 143,
'happening': 25,
'able': 99,
'use': 310,
'american': 35,
'express': 38,
'add': 84,
'money': 895,
'transfering': 3,
'havent': 83,
'funds': 142,
'yet': 202,
'choose': 14,
'visa': 70,
'mastercard': 77,
'order': 91,
'cant': 175,
'topup': 283,
'declined': 210,
'app': 354,
'shows': 76,
'fraudulent': 9,
'source': 30,
'long': 275,
'uk': 50,
'usually': 14,
'take': 260,
'showing': 159,
'everything': 36,
'actually': 29,
'went': 71,
'okay': 18,
'work': 169,
'shop': 7,
'physical': 59,
'delivered': 36,
'seller': 56,
'told': 11,
'paid': 41,

'reviewing': 4,
'subtract': 1,
'added': 29,
'back': 138,
'care': 3,
'promptly': 1,
'length': 3,
'credit': 75,
'transactions': 70,
'ive': 134,
'category': 1,
'days': 91,
'wont': 68,
'move': 12,
'make': 256,
'go': 198,
'checked': 47,
'receipts': 3,
'correct': 63,
'indicated': 1,
'previous': 5,
'email': 3,
'purchasing': 6,
'noticed': 55,
'rate': 257,
'reason': 73,
'needs': 32,
'verifying': 27,
'personal': 19,
'details': 69,
'got': 192,
'mugged': 8,
'took': 25,
'help': 282,
'steps': 39,
'check': 136,
'code': 43,
'topping': 59,
'pin': 243,
'number': 61,
'machine': 44,
'necessary': 6,
'gave': 29,
'pounds': 28,
'instead': 19,
'requested': 50,
'appear': 16,

'instant': 5,
'statement': 153,
'reflect': 9,
'issued': 9,
'receive': 95,
'access': 50,
'bank': 144,
'old': 30,
'ups': 16,
'done': 45,
'apple': 50,
'watch': 18,
'twice': 50,
'returned': 23,
'something': 169,
'store': 23,
'see': 220,
'appeared': 5,
'exchanging': 26,
'feature': 5,
'short': 2,
'many': 78,
'going': 155,
'wrong': 252,
'anymore': 16,
'possible': 124,
'someone': 131,
'else': 24,
'activate': 67,
'bought': 85,
'delete': 32,
'happy': 8,
'service': 29,
'youre': 6,
'providing': 2,
'smart': 3,
'phone': 73,
'lost': 79,
'stolen': 68,
'vacation': 12,
'immediately': 34,
'hold': 30,
'kids': 11,
'daughter': 28,
'passcode': 47,
'completely': 4,
'slipped': 1,

'mind': 6,
'hi': 56,
'first': 33,
'customer': 18,
'shown': 29,
'last': 54,
'half': 4,
'hour': 14,
'fix': 30,
'assistance': 14,
'really': 52,
'frustrating': 2,
'trouble': 13,
'id': 65,
'deposited': 65,
'mistake': 30,
'process': 62,
'changing': 6,
'much': 110,
'travel': 7,
'abroad': 42,
'ordering': 6,
'disposable': 149,
'methods': 5,
'accepted': 30,
'transferred': 66,
'morning': 59,
'fees': 100,
'using': 175,
'european': 24,
'hidden': 5,
'cancelled': 54,
'typically': 7,
'friend': 55,
'earlier': 57,
'even': 31,
'though': 36,
'hours': 20,
'ago': 139,
'documentation': 2,
'children': 10,
'available': 55,
'found': 40,
'put': 58,
'come': 79,
'yall': 1,
'support': 41,

'small': 5,
'needed': 21,
'applied': 44,
'recipient': 16,
'submit': 4,
'couple': 74,
'continues': 1,
'unsuccessful': 6,
'may': 29,
'status': 24,
'tell': 189,
'failed': 41,
'withdraw': 89,
'tracked': 2,
'hello': 49,
'seems': 58,
'never': 50,
'seen': 23,
'refunds': 4,
'allowed': 40,
'certain': 30,
'items': 12,
'recognize': 51,
'left': 13,
'parents': 1,
'house': 6,
'send': 59,
'spare': 3,
'meantime': 1,
'delivering': 1,
'update': 30,
'sepa': 17,
'exchanges': 30,
'moved': 17,
'acceptable': 6,
'deposit': 77,
'denied': 12,
'option': 26,
'selected': 3,
'beneficiary': 51,
'saw': 35,
'merchant': 26,
'stay': 5,
'keeps': 35,
'declining': 16,
'two': 42,
'different': 73,

'atms': 39,
'already': 73,
'alright': 7,
'notting': 9,
'hill': 9,
'must': 23,
'pound': 36,
'currency': 156,
'ways': 2,
'paying': 25,
'choice': 2,
'reader': 1,
'accept': 57,
'foreign': 55,
'weekdays': 2,
'weekends': 2,
'completed': 29,
'waiting': 59,
'way': 80,
'somewhere': 6,
'form': 7,
'payments': 73,
'prove': 16,
'wallet': 25,
'x': 6,
'approved': 9,
'separate': 3,
'worried': 12,
'post': 13,
'address': 21,
'information': 56,
'contactless': 21,
'function': 9,
'arrived': 44,
'denying': 2,
'transfers': 110,
'horrible': 2,
'request': 21,
'salary': 28,
'might': 22,
'strange': 24,
'restaurant': 19,
'wouldnt': 18,
'taking': 37,
'asked': 34,
'pretty': 12,
'slow': 2,

'increased': 5,
'addition': 2,
'happened': 50,
'downsides': 1,
'reasons': 12,
'fail': 11,
'refills': 1,
'traveling': 19,
'somebody': 9,
'seeing': 21,
'adding': 17,
'missing': 30,
'copy': 4,
'£': 24,
'countries': 39,
'often': 7,
'worked': 28,
'places': 10,
'inform': 5,
'restrictions': 17,
'right': 88,
'away': 13,
'locations': 10,
'higher': 9,
'europe': 23,
'cannot': 26,
'dispute': 20,
'verified': 24,
'per': 13,
'person': 16,
'offer': 23,
'discount': 10,
'since': 50,
'frequently': 10,
'package': 3,
'auto': 45,
'limit': 54,
'anything': 37,
'mines': 2,
'expire': 24,
'takes': 9,
'retract': 1,
'china': 42,
'urgent': 19,
'sure': 75,
'plenty': 1,
'couldnt': 21,

```
'week': 55,
'view': 2,
'history': 6,
'came': 38,
'accounts': 11,
'track': 22,
'wait': 44,
'rejected': 31,
'readily': 1,
'give': 64,
'dollar': 10,
'ready': 3,
'enjoy': 1,
'tutor': 1,
'session': 1,
'withdrawl': 8,
'skip': 1,
'international': 40,
'france': 10,
'forewarned': 1,
'stop': 25,
'accessing': 2,
'eu': 22,
'online': 53,
'says': 52,
'interested': 11,
'rates': 39,
'based': 1,
'live': 20,
'assessed': 2,
'forgot': 14,
'trip': 5,
'overseas': 12,
'unblock': 22,
'blocked': 22,
'giving': 7,
'message': 39,
'device': 5,
'log': 5,
'onto': 3,
'platform': 1,
'needing': 3,
'doesnt': 61,
'seem': 42,
'wondering': 16,
'several': 37,
'charges': 69,
```

'charge': 234,
'mine': 37,
'expiring': 8,
'avoid': 3,
'future': 4,
'checking': 19,
'authorized': 3,
'reversed': 13,
'replaced': 3,
'revert': 15,
'second': 34,
'canceled': 12,
'automatically': 26,
'convert': 7,
'company': 17,
'awful': 1,
'procedure': 10,
'happen': 12,
'quick': 5,
'expedition': 1,
'crucial': 1,
'approximately': 4,
'specify': 2,
'day': 48,
'show': 84,
'orders': 1,
'less': 25,
'activity': 5,
'withdrawals': 24,
'always': 13,
'thought': 57,
'free': 49,
'stopped': 7,
'product': 11,
'ordered': 29,
'unbeknownst': 1,
'additional': 42,
'prior': 2,
'notification': 2,
'sorts': 1,
'required': 12,
'explained': 1,
'figure': 19,
'maximum': 11,
'via': 11,
'seconds': 3,
'hit': 5,

'follow': 4,
'directions': 3,
'calculated': 4,
'reach': 10,
'start': 11,
'list': 9,
'saying': 16,
'anticipated': 1,
'password': 20,
'reset': 16,
'unknowingly': 1,
'redeposied': 1,
'resolve': 10,
'estimated': 3,
'include': 1,
'latest': 4,
'entered': 19,
'unblocked': 7,
'theres': 33,
'somehow': 1,
'average': 2,
'shouldnt': 25,
'name': 32,
'married': 6,
'look': 35,
'owe': 1,
'beneficiery': 2,
'compromised': 11,
'swift': 15,
'open': 35,
'offspring': 1,
'ideal': 1,
'understand': 27,
'portugal': 1,
'operate': 3,
'including': 1,
'past': 20,
'month': 13,
'place': 27,
'good': 12,
'suddenly': 17,
'rewarded': 3,
'quickly': 16,
'type': 13,
'cleared': 8,
'withdrawn': 14,
'holiday': 15,

'guess': 5,
'overcharged': 11,
'withdrew': 32,
'require': 6,
'unfamiliar': 6,
'calling': 1,
'actual': 11,
'period': 3,
'swallow': 1,
'marriage': 2,
'three': 3,
'months': 11,
'cat': 2,
'fluffy': 3,
'unfortunately': 3,
'fan': 1,
'pet': 1,
'said': 15,
'nothing': 15,
'cause': 12,
'failure': 2,
'talk': 7,
'debited': 1,
'contacted': 14,
'directed': 1,
'issue': 41,
'within': 12,
'definitely': 22,
'fully': 2,
'phase': 1,
'normal': 10,
'turn': 4,
'around': 5,
'expect': 28,
'link': 31,
'friends': 20,
'double': 32,
'home': 13,
'cool': 2,
'saturday': 12,
'accident': 1,
'digits': 2,
'fixing': 2,
'purchases': 19,
'authorize': 11,
'quite': 8,
'idea': 10,

'freeze': 21,
'utilizing': 1,
'without': 29,
'authorization': 1,
'previously': 6,
'attempt': 4,
'remove': 14,
'clearly': 6,
'issuing': 1,
'frames': 1,
'recognise': 11,
'particular': 1,
'coming': 19,
'specifically': 2,
'ask': 7,
'although': 6,
'pass': 2,
'acount': 1,
'currently': 7,
'autotop': 8,
'close': 20,
'determined': 2,
'changed': 35,
'remotely': 2,
'visiting': 1,
'purchased': 41,
'low': 6,
'shopping': 16,
'tickets': 2,
'age': 24,
'child': 7,
'services': 17,
'limitations': 3,
'finishing': 1,
'directly': 9,
'matter': 7,
'bit': 12,
'concern': 1,
'notice': 11,
'weeks': 37,
'trace': 8,
'truly': 3,
'unable': 40,
'crypto': 20,
'keen': 1,
'buy': 24,
'applicationi': 1,

'thats': 13,
'expired': 8,
'kind': 13,
'looking': 23,
'price': 5,
'offered': 5,
'disappeared': 18,
'urgency': 1,
'odd': 8,
'remote': 4,
'town': 8,
'reasonable': 1,
'hundred': 2,
'realize': 5,
'single': 5,
'visit': 5,
'removed': 6,
'mistakenly': 2,
'taken': 19,
'occasions': 1,
'elaborate': 2,
'upon': 3,
'replace': 8,
'step': 3,
'businesses': 7,
'limits': 16,
'transferring': 35,
'wheres': 9,
'assess': 1,
'try': 16,
'unlimited': 2,
'reaches': 3,
'forms': 4,
'types': 10,
'disappointed': 4,
'bad': 10,
'hope': 5,
'confirm': 6,
'official': 8,
'interbank': 11,
'unknown': 4,
'submitted': 5,
'query': 1,
'thing': 18,
'receiving': 20,
'separably': 1,
'brand': 2,

'activation': 12,
'failing': 11,
'fraud': 8,
'protect': 1,
'sale': 1,
'transferis': 1,
'expiration': 7,
'near': 3,
'pull': 2,
'unusual': 5,
'believe': 26,
'didny': 1,
'retrieve': 2,
'looked': 13,
'broken': 15,
'given': 12,
'data': 2,
'exposed': 1,
'stole': 15,
'banks': 2,
'happens': 14,
'stipulations': 1,
'entail': 1,
'asap': 18,
'question': 5,
'concerning': 1,
'guys': 18,
'accurate': 5,
'uknown': 1,
'origin': 1,
'sense': 3,
'ok': 11,
'wish': 5,
'select': 2,
'preference': 2,
'recover': 2,
'swallowed': 6,
'stuck': 16,
'changes': 5,
'supplier': 1,
'cardsper': 1,
'problems': 12,
'beneficiarys': 1,
'credited': 7,
'difference': 3,
'union': 2,
'full': 14,

```
'banking': 3,
'mean': 21,
'little': 6,
'bill': 7,
'tired': 3,
'whatwhen': 1,
'set': 35,
'displaying': 3,
'report': 9,
'correctly': 8,
'standing': 5,
'telling': 4,
'austria': 5,
'theretotop': 1,
'exchanged': 9,
'anywhere': 22,
'l': 2,
'return': 20,
'advise': 7,
'performed': 3,
'print': 1,
'date': 18,
'unhappy': 3,
'cut': 2,
'waited': 1,
'retailers': 3,
'activated': 9,
'messed': 5,
'recent': 27,
'goes': 7,
'login': 3,
'point': 11,
'deliveries': 1,
'weird': 4,
'recognizing': 4,
'united': 6,
'states': 9,
'appreciate': 3,
'siphoned': 1,
'helping': 1,
'japan': 1,
'decorations': 1,
'local': 3,
'stores': 4,
'differed': 1,
'greatly': 1,
'showed': 5,
```

```
'hacked': 3,
'policy': 10,
'end': 10,
'gonna': 1,
'enabled': 1,
'living': 4,
'ship': 2,
'sort': 5,
'whenever': 1,
'allow': 20,
'usd': 15,
'hate': 2,
'exactly': 3,
'withdrawls': 1,
'progressseems': 1,
'wrongi': 1,
'memberships': 1,
'appears': 25,
'placed': 8,
'refunded': 15,
'ended': 2,
'info': 22,
'hiaccordingly': 1,
'accountwhat': 1,
'reverse': 11,
'apply': 8,
'reactivate': 17,
'turns': 3,
'jacket': 9,
'pocket': 3,
'withdraws': 3,
'tranfering': 1,
'dispose': 1,
'reactive': 2,
'continue': 3,
'statements': 11,
'upi': 1,
'submitting': 1,
'refilled': 2,
'attempted': 21,
'enter': 4,
'russian': 9,
'ruble': 7,
'wen': 1,
'forget': 2,
'eaten': 1,
'patentp': 1,
```

'proof': 6,
'mail': 15,
'deliver': 7,
'rent': 16,
'side': 8,
'travelling': 6,
'euros': 3,
'drunk': 2,
'mistyped': 1,
'restored': 1,
'seemingly': 2,
'randomly': 2,
'system': 28,
'complicated': 1,
'til': 1,
'severely': 1,
'known': 3,
'requirements': 5,
'opening': 2,
'call': 2,
'locate': 6,
'marital': 1,
'learning': 2,
'supplementary': 1,
'enough': 15,
'unsure': 5,
'topups': 27,
'gotten': 16,
'canceling': 4,
'products': 3,
'realized': 4,
'correcti': 3,
'tomorrow': 10,
'delivery': 25,
'solve': 6,
'discounts': 5,
'urgently': 16,
'duplicated': 7,
'goofed': 1,
'ate': 6,
'identify': 7,
'important': 10,
'paranoid': 1,
'factor': 1,
'configured': 2,
'hey': 12,
'theyre': 4,

'forever': 9,
'difficult': 1,
'electronic': 2,
'suffice': 1,
'confusing': 2,
'confused': 6,
'ass': 1,
'size': 2,
'supposed': 18,
'big': 3,
'regretted': 1,
'cover': 3,
'cast': 1,
'works': 13,
'bring': 1,
'identification': 12,
'swap': 1,
'cloned': 1,
'increase': 3,
'rather': 5,
'decreasing': 1,
'customers': 3,
'incentives': 1,
'youngest': 1,
'simply': 2,
'greece': 1,
'course': 1,
'investigate': 2,
'pulled': 3,
'brought': 1,
'stuff': 1,
'suspend': 1,
'expected': 10,
'recipients': 2,
'machines': 13,
'kept': 9,
'normally': 13,
'confirmed': 6,
'countrys': 3,
'pulling': 3,
'instructions': 4,
'activating': 19,
'manually': 3,
'hotel': 8,
'ran': 2,
'causing': 6,
'thanks': 6,

```
           'institutions': 1,
           'far': 12,
           'obtain': 5,
           'possibility': 3,
           'faster': 7,
           'warned': 4,
           'explanation': 1,
           'hard': 4,
           'application': 2,
           'checque': 2,
           'authorise': 3,
           'arrival': 3,
           'specific': 10,
           'payee': 1,
           'courier': 1,
           'cone': 1,
           'experiencing': 2,
           'experienced': 1,
           'terminated': 2,
           'sorted': 2,
           'inquiring': 2,
           'unlocked': 2,
           'instantly': 2,
           'edit': 11,
           'intervals': 7,
           'accepting': 4,
           'intended': 1,
           'tracking': 8,
           'eur': 15,
           'anyway': 3,
           'setup': 4,
           'automatic': 6,
           'discounted': 1,
           'finish': 5,
           'unsuccessfully': 2,
           'deducted': 8,
           …})
```

[ ]:
```
# Check the most frequent words
word_counts.most_common(5)
```

[ ]:
```
[('card', 1942),
 ('account', 1059),
 ('money', 895),
 ('transfer', 807),
 ('get', 606)]
```

[ ]:
```python
# Store the size of the vocabulary
voca_size = len(word_counts)
print(voca_size)
```

```
2088
```

[ ]:
```python
# Change the X data to arrays
X_train_array = X_train.to_numpy()
X_val_array = X_val.to_numpy()
X_test_array = X_test.to_numpy()
```

[ ]:
```python
# Change the y data to arrays
y_train_array = y_train.to_numpy()
y_val_array = y_val.to_numpy()
y_test_array = y_test.to_numpy()
```

[ ]:
```python
# Check the dimension of the variables
print(X_train_array.shape)
print(X_val_array.shape)
print(X_test_array.shape)
print(y_train_array.shape)
print(y_val_array.shape)
print(y_test_array.shape)
```

```
(7346,)
(1837,)
(3080,)
(7346,)
(1837,)
(3080,)
```

[ ]:
```python
# Check the type of the variables
print(X_train_array.__class__)
print(X_val_array.__class__)
print(X_test_array.__class__)
print(y_train_array.__class__)
print(y_val_array.__class__)
print(y_test_array.__class__)
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

[ ]:
```python
# Import tokenizer
from tensorflow.keras.preprocessing.text import Tokenizer

# Define the tokenizer with vocabulary size of the training data
tokenizer = Tokenizer(num_words=voca_size)
# Fit the tokenizer with X train
tokenizer.fit_on_texts(X_train_array)
```

[ ]:
```python
# Define the word index
word_index = tokenizer.word_index
word_index
```

[ ]:
```
{'card': 1,
 'account': 2,
 'money': 3,
 'transfer': 4,
 'get': 5,
 'payment': 6,
 'need': 7,
 'cash': 8,
 'top': 9,
 'exchange': 10,
 'charged': 11,
 'please': 12,
 'app': 13,
 'atm': 14,
 'fee': 15,
 'use': 16,
 'pending': 17,
 'made': 18,
 'topup': 19,
 'help': 20,
 'long': 21,
 'would': 22,
 'still': 23,
 'take': 24,
 'dont': 25,
 'didnt': 26,
 'rate': 27,
 'transaction': 28,
 'new': 29,
 'make': 30,
 'wrong': 31,
 'pin': 32,
 'charge': 33,
 'know': 34,
 'im': 35,
```

```
'see': 36,
'like': 37,
'cards': 38,
'refund': 39,
'want': 40,
'withdrawal': 41,
'declined': 42,
'one': 43,
'extra': 44,
'yet': 45,
'tried': 46,
'go': 47,
'got': 48,
'tell': 49,
'received': 50,
'cant': 51,
'using': 52,
'work': 53,
'something': 54,
'amount': 55,
'virtual': 56,
'showing': 57,
'currency': 58,
'going': 59,
'statement': 60,
'identity': 61,
'change': 62,
'disposable': 63,
'currencies': 64,
'working': 65,
'bank': 66,
'whats': 67,
'funds': 68,
'ago': 69,
'back': 70,
'check': 71,
'hasnt': 72,
'ive': 73,
'pay': 74,
'someone': 75,
'isnt': 76,
'think': 77,
'possible': 78,
'time': 79,
'getting': 80,
'purchase': 81,
'much': 82,
```

```
'transfers': 83,
'sent': 84,
'debit': 85,
'find': 86,
'times': 87,
'cancel': 88,
'fees': 89,
'able': 90,
'used': 91,
'could': 92,
'verify': 93,
'receive': 94,
'order': 95,
'days': 96,
'balance': 97,
'withdraw': 98,
'right': 99,
'verification': 100,
'bought': 101,
'cheque': 102,
'add': 103,
'show': 104,
'havent': 105,
'way': 106,
'another': 107,
'lost': 108,
'come': 109,
'many': 110,
'mastercard': 111,
'deposit': 112,
'direct': 113,
'today': 114,
'shows': 115,
'credit': 116,
'sure': 117,
'couple': 118,
'reason': 119,
'phone': 120,
'different': 121,
'already': 122,
'payments': 123,
'item': 124,
'trying': 125,
'went': 126,
'visa': 127,
'transactions': 128,
'problem': 129,
```

```
'details': 130,
'charges': 131,
'wont': 132,
'stolen': 133,
'us': 134,
'activate': 135,
'gone': 136,
'transferred': 137,
'id': 138,
'deposited': 139,
'give': 140,
'correct': 141,
'process': 142,
'explain': 143,
'number': 144,
'doesnt': 145,
'let': 146,
'keep': 147,
'physical': 148,
'topping': 149,
'morning': 150,
'send': 151,
'waiting': 152,
'put': 153,
'seems': 154,
'cost': 155,
'earlier': 156,
'accept': 157,
'thought': 158,
'seller': 159,
'hi': 160,
'information': 161,
'noticed': 162,
'friend': 163,
'available': 164,
'foreign': 165,
'week': 166,
'last': 167,
'cancelled': 168,
'limit': 169,
'online': 170,
'yesterday': 171,
'really': 172,
'says': 173,
'recognize': 174,
'beneficiary': 175,
'complete': 176,
```

```
'uk': 177,
'requested': 178,
'access': 179,
'apple': 180,
'twice': 181,
'never': 182,
'happened': 183,
'since': 184,
'reverted': 185,
'wasnt': 186,
'hello': 187,
'free': 188,
'country': 189,
'day': 190,
'gbp': 191,
'checked': 192,
'passcode': 193,
'done': 194,
'auto': 195,
'multiple': 196,
'machine': 197,
'applied': 198,
'arrived': 199,
'wait': 200,
'incorrect': 201,
'error': 202,
'code': 203,
'wanted': 204,
'abroad': 205,
'two': 206,
'china': 207,
'seem': 208,
'additional': 209,
'google': 210,
'paid': 211,
'support': 212,
'failed': 213,
'issue': 214,
'purchased': 215,
'making': 216,
'found': 217,
'allowed': 218,
'international': 219,
'unable': 220,
'soon': 221,
'recently': 222,
'steps': 223,
```

'atms': 224,
'countries': 225,
'rates': 226,
'message': 227,
'express': 228,
'came': 229,
'taking': 230,
'anything': 231,
'several': 232,
'mine': 233,
'weeks': 234,
'everything': 235,
'delivered': 236,
'though': 237,
'pound': 238,
'american': 239,
'saw': 240,
'keeps': 241,
'look': 242,
'open': 243,
'changed': 244,
'transferring': 245,
'set': 246,
'immediately': 247,
'asked': 248,
'second': 249,
'first': 250,
'theres': 251,
'needs': 252,
'delete': 253,
'name': 254,
'withdrew': 255,
'double': 256,
'even': 257,
'rejected': 258,
'link': 259,
'source': 260,
'old': 261,
'hold': 262,
'fix': 263,
'mistake': 264,
'accepted': 265,
'certain': 266,
'update': 267,
'exchanges': 268,
'missing': 269,
'actually': 270,

```
'added': 271,
'gave': 272,
'service': 273,
'shown': 274,
'may': 275,
'completed': 276,
'ordered': 277,
'without': 278,
'remember': 279,
'pounds': 280,
'daughter': 281,
'salary': 282,
'worked': 283,
'expect': 284,
'system': 285,
'verifying': 286,
'understand': 287,
'place': 288,
'recent': 289,
'topups': 290,
'fiat': 291,
'looks': 292,
'longer': 293,
'exchanging': 294,
'option': 295,
'merchant': 296,
'cannot': 297,
'automatically': 298,
'believe': 299,
'supported': 300,
'happening': 301,
'took': 302,
'paying': 303,
'wallet': 304,
'stop': 305,
'less': 306,
'shouldnt': 307,
'appears': 308,
'delivery': 309,
'updated': 310,
'topped': 311,
'arrive': 312,
'else': 313,
'european': 314,
'status': 315,
'strange': 316,
'£': 317,
```

```
'verified': 318,
'expire': 319,
'withdrawals': 320,
'age': 321,
'buy': 322,
'returned': 323,
'store': 324,
'seen': 325,
'must': 326,
'europe': 327,
'offer': 328,
'looking': 329,
'duplicate': 330,
'expires': 331,
'might': 332,
'track': 333,
'eu': 334,
'unblock': 335,
'blocked': 336,
'definitely': 337,
'anywhere': 338,
'info': 339,
'withdrawing': 340,
'things': 341,
'needed': 342,
'address': 343,
'contactless': 344,
'request': 345,
'seeing': 346,
'couldnt': 347,
'freeze': 348,
'mean': 349,
'attempted': 350,
'hours': 351,
'dispute': 352,
'live': 353,
'password': 354,
'past': 355,
'friends': 356,
'close': 357,
'crypto': 358,
'receiving': 359,
'return': 360,
'allow': 361,
'holding': 362,
'assist': 363,
'personal': 364,
```

'instead': 365,
'restaurant': 366,
'traveling': 367,
'urgent': 368,
'checking': 369,
'figure': 370,
'entered': 371,
'purchases': 372,
'coming': 373,
'taken': 374,
'activating': 375,
'okay': 376,
'watch': 377,
'customer': 378,
'wouldnt': 379,
'disappeared': 380,
'thing': 381,
'asap': 382,
'guys': 383,
'date': 384,
'supposed': 385,
'€': 386,
'provide': 387,
'decline': 388,
'also': 389,
'sepa': 390,
'moved': 391,
'adding': 392,
'restrictions': 393,
'company': 394,
'suddenly': 395,
'services': 396,
'reactivate': 397,
'aud': 398,
'people': 399,
'appear': 400,
'ups': 401,
'anymore': 402,
'recipient': 403,
'declining': 404,
'prove': 405,
'person': 406,
'wondering': 407,
'saying': 408,
'reset': 409,
'quickly': 410,
'shopping': 411,

'limits': 412,
'try': 413,
'stuck': 414,
'rent': 415,
'gotten': 416,
'urgently': 417,
'revert': 418,
'swift': 419,
'holiday': 420,
'said': 421,
'nothing': 422,
'broken': 423,
'stole': 424,
'usd': 425,
'refunded': 426,
'mail': 427,
'enough': 428,
'eur': 429,
'processed': 430,
'choose': 431,
'usually': 432,
'hour': 433,
'assistance': 434,
'forgot': 435,
'withdrawn': 436,
'contacted': 437,
'remove': 438,
'happens': 439,
'full': 440,
'sending': 441,
'fast': 442,
'next': 443,
'trouble': 444,
'left': 445,
'post': 446,
'away': 447,
'per': 448,
'reversed': 449,
'always': 450,
'month': 451,
'type': 452,
'home': 453,
'thats': 454,
'kind': 455,
'looked': 456,
'works': 457,
'machines': 458,

'normally': 459,
'fine': 460,
'clear': 461,
'receiver': 462,
'listed': 463,
'documents': 464,
'flat': 465,
'move': 466,
'vacation': 467,
'items': 468,
'denied': 469,
'worried': 470,
'pretty': 471,
'reasons': 472,
'overseas': 473,
'canceled': 474,
'happen': 475,
'required': 476,
'good': 477,
'cause': 478,
'within': 479,
'saturday': 480,
'bit': 481,
'activation': 482,
'given': 483,
'problems': 484,
'hey': 485,
'identification': 486,
'far': 487,
'deal': 488,
'however': 489,
'current': 490,
'well': 491,
'large': 492,
'unauthorized': 493,
'aware': 494,
'told': 495,
'kids': 496,
'fail': 497,
'accounts': 498,
'interested': 499,
'product': 500,
'maximum': 501,
'via': 502,
'start': 503,
'compromised': 504,
'overcharged': 505,

```
'actual': 506,
'months': 507,
'authorize': 508,
'recognise': 509,
'notice': 510,
'interbank': 511,
'failing': 512,
'ok': 513,
'point': 514,
'reverse': 515,
'statements': 516,
'edit': 517,
'checks': 518,
'charging': 519,
'replacement': 520,
'children': 521,
'places': 522,
'locations': 523,
'discount': 524,
'frequently': 525,
'dollar': 526,
'france': 527,
'procedure': 528,
'reach': 529,
'resolve': 530,
'normal': 531,
'idea': 532,
'types': 533,
'bad': 534,
'policy': 535,
'end': 536,
'tomorrow': 537,
'important': 538,
'expected': 539,
'specific': 540,
'create': 541,
'options': 542,
'standard': 543,
'frame': 544,
'accidentally': 545,
'buying': 546,
'fraudulent': 547,
'reflect': 548,
'issued': 549,
'notting': 550,
'hill': 551,
'approved': 552,
```

```
'function': 553,
'somebody': 554,
'higher': 555,
'takes': 556,
'list': 557,
'directly': 558,
'wheres': 559,
'report': 560,
'exchanged': 561,
'activated': 562,
'states': 563,
'jacket': 564,
'russian': 565,
'forever': 566,
'kept': 567,
'issues': 568,
'terminate': 569,
'gas': 570,
'progress': 571,
'landlord': 572,
'high': 573,
'due': 574,
'contact': 575,
'maybe': 576,
'located': 577,
'mugged': 578,
'happy': 579,
'withdrawl': 580,
'expiring': 581,
'cleared': 582,
'quite': 583,
'autotop': 584,
'trace': 585,
'expired': 586,
'odd': 587,
'town': 588,
'replace': 589,
'official': 590,
'fraud': 591,
'correctly': 592,
'placed': 593,
'apply': 594,
'side': 595,
'hotel': 596,
'tracking': 597,
'deducted': 598,
'ones': 599,
```

```
'every': 600,
'arent': 601,
'costs': 602,
'depositing': 603,
'mortgage': 604,
'shop': 605,
'travel': 606,
'typically': 607,
'alright': 608,
'form': 609,
'often': 610,
'giving': 611,
'convert': 612,
'stopped': 613,
'unblocked': 614,
'talk': 615,
'ask': 616,
'currently': 617,
'child': 618,
'matter': 619,
'businesses': 620,
'expiration': 621,
'credited': 622,
'bill': 623,
'advise': 624,
'goes': 625,
'ruble': 626,
'deliver': 627,
'duplicated': 628,
'identify': 629,
'faster': 630,
'intervals': 631,
'play': 632,
'impression': 633,
'recall': 634,
'spain': 635,
'years': 636,
'concerned': 637,
'outside': 638,
'area': 639,
'say': 640,
'locked': 641,
'reflected': 642,
'posted': 643,
'comes': 644,
'purchasing': 645,
'necessary': 646,
```

```
'youre': 647,
'mind': 648,
'changing': 649,
'ordering': 650,
'unsuccessful': 651,
'house': 652,
'acceptable': 653,
'somewhere': 654,
'x': 655,
'history': 656,
'married': 657,
'require': 658,
'unfamiliar': 659,
'previously': 660,
'clearly': 661,
'although': 662,
'low': 663,
'removed': 664,
'confirm': 665,
'swallowed': 666,
'little': 667,
'united': 668,
'proof': 669,
'travelling': 670,
'locate': 671,
'solve': 672,
'ate': 673,
'confused': 674,
'confirmed': 675,
'causing': 676,
'thanks': 677,
'automatic': 678,
'entering': 679,
'block': 680,
'putting': 681,
'frequent': 682,
'car': 683,
'permission': 684,
'proving': 685,
'minutes': 686,
'lot': 687,
'feel': 688,
'letting': 689,
'fixed': 690,
'security': 691,
'results': 692,
'typo': 693,
```

```
'regarding': 694,
'incorrectly': 695,
'location': 696,
'resolved': 697,
'five': 698,
'shipped': 699,
'previous': 700,
'instant': 701,
'appeared': 702,
'feature': 703,
'methods': 704,
'hidden': 705,
'small': 706,
'stay': 707,
'increased': 708,
'inform': 709,
'trip': 710,
'device': 711,
'log': 712,
'quick': 713,
'activity': 714,
'hit': 715,
'guess': 716,
'around': 717,
'price': 718,
'offered': 719,
'realize': 720,
'single': 721,
'visit': 722,
'hope': 723,
'submitted': 724,
'unusual': 725,
'question': 726,
'accurate': 727,
'wish': 728,
'changes': 729,
'standing': 730,
'austria': 731,
'messed': 732,
'showed': 733,
'sort': 734,
'requirements': 735,
'unsure': 736,
'discounts': 737,
'rather': 738,
'obtain': 739,
'finish': 740,
```

```
'extremely': 741,
'satisfied': 742,
'passed': 743,
'modify': 744,
'appearing': 745,
'finding': 746,
'grocery': 747,
'perform': 748,
'gets': 749,
'attempts': 750,
'photos': 751,
'family': 752,
'forgotten': 753,
'associated': 754,
'lesser': 755,
'random': 756,
'thru': 757,
'tries': 758,
'terrible': 759,
'groceries': 760,
'rejecting': 761,
'frozen': 762,
'prefer': 763,
'heard': 764,
'linked': 765,
'refused': 766,
'reported': 767,
'completing': 768,
'reviewing': 769,
'completely': 770,
'half': 771,
'submit': 772,
'refunds': 773,
'copy': 774,
'future': 775,
'approximately': 776,
'follow': 777,
'calculated': 778,
'latest': 779,
'turn': 780,
'attempt': 781,
'remote': 782,
'forms': 783,
'disappointed': 784,
'unknown': 785,
'telling': 786,
'weird': 787,
```

'recognizing': 788,
'stores': 789,
'living': 790,
'enter': 791,
'canceling': 792,
'realized': 793,
'theyre': 794,
'instructions': 795,
'warned': 796,
'hard': 797,
'accepting': 798,
'setup': 799,
'dollars': 800,
'possibly': 801,
'deactivate': 802,
'usa': 803,
'spending': 804,
'refuse': 805,
'alternatives': 806,
'images': 807,
'readable': 808,
'economic': 809,
'switzerland': 810,
'properly': 811,
'ahead': 812,
'fund': 813,
'initiated': 814,
'closest': 815,
'following': 816,
'inside': 817,
'policies': 818,
'along': 819,
'mailed': 820,
'best': 821,
'son': 822,
'unlock': 823,
'case': 824,
'business': 825,
'resident': 826,
'rules': 827,
'attempting': 828,
'hoping': 829,
'better': 830,
'billing': 831,
'apparently': 832,
'finished': 833,
'receipt': 834,

```
'older': 835,
'recieve': 836,
'caused': 837,
'rubles': 838,
'stays': 839,
'expecting': 840,
'chose': 841,
'pick': 842,
'transfering': 843,
'care': 844,
'length': 845,
'receipts': 846,
'email': 847,
'smart': 848,
'spare': 849,
'selected': 850,
'separate': 851,
'package': 852,
'ready': 853,
'onto': 854,
'needing': 855,
'avoid': 856,
'authorized': 857,
'replaced': 858,
'seconds': 859,
'directions': 860,
'estimated': 861,
'operate': 862,
'rewarded': 863,
'period': 864,
'three': 865,
'fluffy': 866,
'unfortunately': 867,
'limitations': 868,
'truly': 869,
'upon': 870,
'step': 871,
'reaches': 872,
'near': 873,
'sense': 874,
'difference': 875,
'banking': 876,
'tired': 877,
'displaying': 878,
'performed': 879,
'unhappy': 880,
'retailers': 881,
```

```
'login': 882,
'appreciate': 883,
'local': 884,
'hacked': 885,
'exactly': 886,
'turns': 887,
'pocket': 888,
'withdraws': 889,
'continue': 890,
'euros': 891,
'known': 892,
'products': 893,
'correcti': 894,
'big': 895,
'cover': 896,
'increase': 897,
'customers': 898,
'pulled': 899,
'countrys': 900,
'pulling': 901,
'manually': 902,
'possibility': 903,
'authorise': 904,
'arrival': 905,
'anyway': 906,
'facing': 907,
'running': 908,
'rest': 909,
'limited': 910,
'finally': 911,
'anyone': 912,
'thinking': 913,
'suspicious': 914,
'cancellation': 915,
'provided': 916,
'gives': 917,
'forward': 918,
'sad': 919,
'decided': 920,
'eligible': 921,
'broke': 922,
'obtained': 923,
'upset': 924,
'false': 925,
'suppose': 926,
'almost': 927,
'visited': 928,
```

```
'explaining': 929,
'uses': 930,
'accessed': 931,
'numbers': 932,
'completion': 933,
'frustrated': 934,
'someones': 935,
'metro': 936,
'active': 937,
'bag': 938,
'regards': 939,
'minimum': 940,
'victim': 941,
'empty': 942,
'reside': 943,
'structure': 944,
'others': 945,
'thank': 946,
'fair': 947,
'method': 948,
'employer': 949,
'review': 950,
'according': 951,
'benow': 952,
'draw': 953,
'wondered': 954,
'separately': 955,
'settings': 956,
'spend': 957,
'realised': 958,
'run': 959,
'speak': 960,
'accepts': 961,
'guide': 962,
'nearly': 963,
'successfully': 964,
'real': 965,
'cancelling': 966,
'successful': 967,
'despite': 968,
'manage': 969,
'yo': 970,
'youve': 971,
'updating': 972,
'stating': 973,
'later': 974,
'max': 975,
```

```
 'carry': 976,
 'processing': 977,
 'night': 978,
 'accidently': 979,
 'typed': 980,
 'simple': 981,
 'havnt': 982,
 'vanished': 983,
 'raise': 984,
 'city': 985,
 'certainly': 986,
 'nowhere': 987,
 'functioning': 988,
 'started': 989,
 'doublecheck': 990,
 'glare': 991,
 'cheques': 992,
 'reached': 993,
 'member': 994,
 'earth': 995,
 'toppedup': 996,
 'residents': 997,
 'asking': 998,
 'specified': 999,
 'lookup': 1000,
 …}
```

[ ]:
```
# Define the maximum length as 29 (the median length of the training set)
max_length_train_text = 29
max_length_train_text
```

[ ]: 29

[ ]:
```
# Encode the text data
X_train_sequences = tokenizer.texts_to_sequences(X_train_array)
X_val_sequences = tokenizer.texts_to_sequences(X_val_array)
X_test_sequences = tokenizer.texts_to_sequences(X_test_array)
```

[ ]:
```
# Add padding to the encoded data
# Import pad_seqesnces
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Padding and truncation will be added to post-texts
X_train_padded = pad_sequences(X_train_sequences, maxlen=max_length_train_text,␣
 ↪padding="post", truncating="post")
X_val_padded = pad_sequences(X_val_sequences, maxlen=max_length_train_text,␣
 ↪padding="post", truncating="post")
```

```
X_test_padded = pad_sequences(X_test_sequences, maxlen=max_length_train_text,␣
↪padding="post", truncating="post")
```

[ ]:
```
# Check the dimension of the variables
print(X_train_padded.shape)
print(X_val_padded.shape)
print(X_test_padded.shape)
```

```
(7346, 29)
(1837, 29)
(3080, 29)
```

[ ]:
```
# Check the first 3 elements of all X train variables
print(X_train_array[3])
print(X_train_sequences[3])
print(X_train_padded[3])
```

```
unauthorized direct debit account
[493, 113, 85, 2]
[493 113  85   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]
```

[ ]:
```
# Define the folder path to save the test files
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/test set/'

# Define the file path
Xtest_path = folder_path + 'X_test_padded.npy'
ytest_path = folder_path + 'y_test_array.npy'

# Save the test set
np.save(Xtest_path, X_test_padded)
np.save(ytest_path, y_test_array)
```

1.4 LSTM (with a word embedding layer)

1.4.1 LSTM (baseline)

[ ]:
```
# Import tensorflow and fix the random seed
import tensorflow as tf
tf.random.set_seed(42)
```

[ ]:
```
# Import keras from tensorflow and layers to build LSTM models
from tensorflow import keras
from tensorflow.keras import layers
```

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
embedding_output_dim = 100 # Random number
```

```
hidden_unit = 30 # Random number
nlabel = 77 # number of classes

# Build the baseline model
model = keras.models.Sequential()
model.add(layers.Embedding(voca_size, embedding_output_dim))
model.add(layers.LSTM(hidden_unit))
model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy'])

# Summary the model
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 100)         208800

 lstm (LSTM)                 (None, 30)                15720

 dense (Dense)               (None, 77)                2387

=================================================================
Total params: 226907 (886.36 KB)
Trainable params: 226907 (886.36 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'LSTM_embedding_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Define early stopping
```

```
es =  tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3) # Random␣
↪number of patience
```

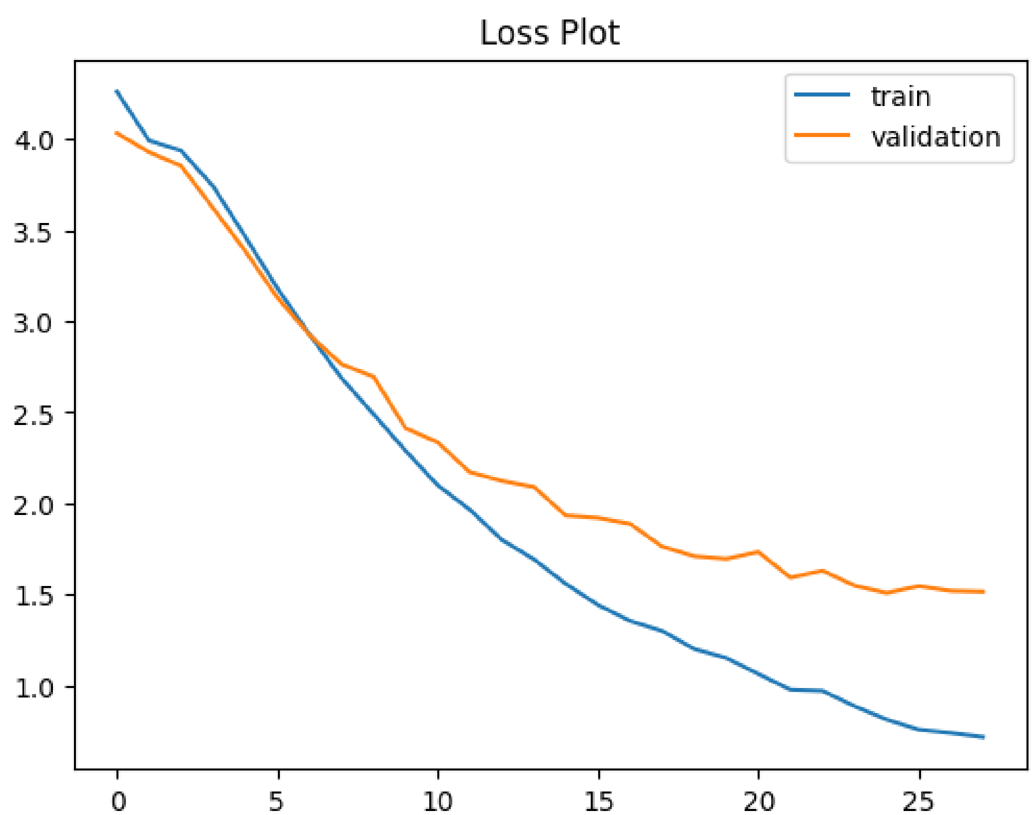
[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = model.fit(
    X_train_padded, y_train,
    epochs = 100, # Random number
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32) # Random number

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Total training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 11s 30ms/step - loss: 4.2625 -
accuracy: 0.0199 - val_loss: 4.0341 - val_accuracy: 0.0338
Epoch 2/100
230/230 [==============================] - 5s 24ms/step - loss: 3.9935 -
accuracy: 0.0317 - val_loss: 3.9296 - val_accuracy: 0.0343
Epoch 3/100
230/230 [==============================] - 7s 31ms/step - loss: 3.9381 -
accuracy: 0.0381 - val_loss: 3.8548 - val_accuracy: 0.0495
Epoch 4/100
230/230 [==============================] - 5s 23ms/step - loss: 3.7424 -
accuracy: 0.0555 - val_loss: 3.6219 - val_accuracy: 0.0719
Epoch 5/100
230/230 [==============================] - 6s 26ms/step - loss: 3.4658 -
accuracy: 0.0826 - val_loss: 3.3864 - val_accuracy: 0.1018
Epoch 6/100
230/230 [==============================] - 7s 29ms/step - loss: 3.1828 -
accuracy: 0.1248 - val_loss: 3.1310 - val_accuracy: 0.1671
Epoch 7/100
230/230 [==============================] - 6s 24ms/step - loss: 2.9293 -
accuracy: 0.1561 - val_loss: 2.9261 - val_accuracy: 0.1622
Epoch 8/100
230/230 [==============================] - 8s 36ms/step - loss: 2.6887 -
```

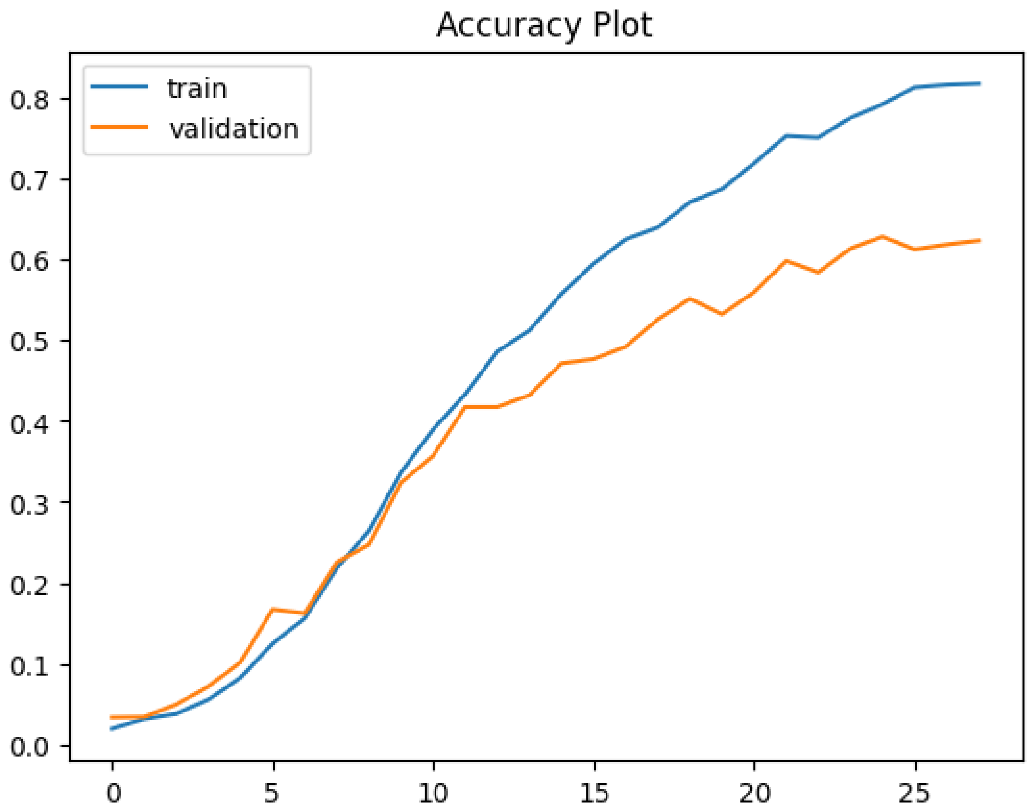
```
accuracy: 0.2181 - val_loss: 2.7648 - val_accuracy: 0.2254
Epoch 9/100
230/230 [==============================] - 5s 24ms/step - loss: 2.4905 -
accuracy: 0.2641 - val_loss: 2.6963 - val_accuracy: 0.2471
Epoch 10/100
230/230 [==============================] - 8s 34ms/step - loss: 2.2890 -
accuracy: 0.3365 - val_loss: 2.4127 - val_accuracy: 0.3239
Epoch 11/100
230/230 [==============================] - 6s 24ms/step - loss: 2.0995 -
accuracy: 0.3897 - val_loss: 2.3355 - val_accuracy: 0.3571
Epoch 12/100
230/230 [==============================] - 5s 22ms/step - loss: 1.9657 -
accuracy: 0.4332 - val_loss: 2.1716 - val_accuracy: 0.4175
Epoch 13/100
230/230 [==============================] - 7s 32ms/step - loss: 1.7997 -
accuracy: 0.4861 - val_loss: 2.1235 - val_accuracy: 0.4175
Epoch 14/100
230/230 [==============================] - 5s 23ms/step - loss: 1.6926 -
accuracy: 0.5123 - val_loss: 2.0899 - val_accuracy: 0.4322
Epoch 15/100
230/230 [==============================] - 6s 27ms/step - loss: 1.5568 -
accuracy: 0.5573 - val_loss: 1.9327 - val_accuracy: 0.4714
Epoch 16/100
230/230 [==============================] - 6s 25ms/step - loss: 1.4417 -
accuracy: 0.5949 - val_loss: 1.9198 - val_accuracy: 0.4769
Epoch 17/100
230/230 [==============================] - 5s 24ms/step - loss: 1.3537 -
accuracy: 0.6250 - val_loss: 1.8871 - val_accuracy: 0.4921
Epoch 18/100
230/230 [==============================] - 10s 44ms/step - loss: 1.2985 -
accuracy: 0.6397 - val_loss: 1.7624 - val_accuracy: 0.5259
Epoch 19/100
230/230 [==============================] - 5s 23ms/step - loss: 1.1997 -
accuracy: 0.6710 - val_loss: 1.7091 - val_accuracy: 0.5514
Epoch 20/100
230/230 [==============================] - 6s 27ms/step - loss: 1.1498 -
accuracy: 0.6870 - val_loss: 1.6951 - val_accuracy: 0.5324
Epoch 21/100
230/230 [==============================] - 6s 24ms/step - loss: 1.0622 -
accuracy: 0.7189 - val_loss: 1.7325 - val_accuracy: 0.5596
Epoch 22/100
230/230 [==============================] - 5s 23ms/step - loss: 0.9759 -
accuracy: 0.7529 - val_loss: 1.5918 - val_accuracy: 0.5983
Epoch 23/100
230/230 [==============================] - 8s 34ms/step - loss: 0.9697 -
accuracy: 0.7509 - val_loss: 1.6296 - val_accuracy: 0.5841
Epoch 24/100
230/230 [==============================] - 5s 23ms/step - loss: 0.8849 -
```

```
accuracy: 0.7753 - val_loss: 1.5478 - val_accuracy: 0.6135
Epoch 25/100
230/230 [==============================] - 6s 24ms/step - loss: 0.8118 -
accuracy: 0.7920 - val_loss: 1.5075 - val_accuracy: 0.6282
Epoch 26/100
230/230 [==============================] - 13s 54ms/step - loss: 0.7563 -
accuracy: 0.8128 - val_loss: 1.5453 - val_accuracy: 0.6124
Epoch 27/100
230/230 [==============================] - 12s 52ms/step - loss: 0.7383 -
accuracy: 0.8161 - val_loss: 1.5192 - val_accuracy: 0.6184
Epoch 28/100
230/230 [==============================] - 9s 39ms/step - loss: 0.7161 -
accuracy: 0.8176 - val_loss: 1.5145 - val_accuracy: 0.6233
Total training time: 193.95937156677246 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 10ms/step - loss: 1.4729 - accuracy:
0.6429
Test Loss: 1.472944736480713
Test Accuracy: 64.29
```

[ ]:
```python
# Import the library to check precision, recall, and F1 score
from sklearn.metrics import precision_score, recall_score, f1_score

# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 2s 8ms/step
Precision: 66.22
Recall: 64.29
F1 Score: 62.87
```

[ ]:
```python
# Error analysis
# Import the library for classification report
from sklearn.metrics import classification_report

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
```

```python
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.79      0.85      0.82        40
           1       0.91      0.78      0.84        40
           2       0.93      0.97      0.95        40
           3       0.68      0.68      0.68        40
           4       0.97      0.72      0.83        40
           5       0.37      0.50      0.43        40
           6       0.57      0.60      0.59        40
           7       0.80      0.70      0.75        40
           8       0.79      0.68      0.73        40
           9       0.93      0.95      0.94        40
          10       0.77      0.25      0.38        40
          11       0.71      0.55      0.62        40
          12       0.65      0.65      0.65        40
          13       0.62      0.85      0.72        40
          14       0.59      0.60      0.59        40
          15       0.54      0.70      0.61        40
          16       0.35      0.28      0.31        40
          17       0.87      0.85      0.86        40
          18       0.77      0.50      0.61        40
          19       0.81      0.75      0.78        40
          20       0.49      0.45      0.47        40
          21       0.78      0.80      0.79        40
          22       0.50      0.40      0.44        40
          23       1.00      0.12      0.22        40
          24       0.70      0.78      0.74        40
          25       0.47      0.70      0.56        40
          26       0.60      0.60      0.60        40
          27       0.65      0.70      0.67        40
          28       0.50      0.68      0.57        40
          29       0.40      0.85      0.54        40
          30       0.97      0.90      0.94        40
          31       0.80      0.80      0.80        40
          32       0.86      0.90      0.88        40
          33       0.61      0.78      0.68        40
          34       0.68      0.62      0.65        40
          35       0.65      0.50      0.56        40
          36       0.76      0.65      0.70        40
          37       0.00      0.00      0.00        40
          38       0.66      0.88      0.75        40
          39       0.83      0.25      0.38        40
```

40	0.61	0.95	0.75	40
41	0.49	0.47	0.48	40
42	0.79	0.85	0.82	40
43	0.42	0.55	0.47	40
44	0.58	0.88	0.70	40
45	0.72	0.70	0.71	40
46	0.82	0.90	0.86	40
47	0.67	0.55	0.60	40
48	0.63	0.47	0.54	40
49	0.85	0.72	0.78	40
50	0.67	0.55	0.60	40
51	0.85	0.85	0.85	40
52	0.76	0.88	0.81	40
53	0.66	0.57	0.61	40
54	0.35	0.72	0.47	40
55	0.97	0.72	0.83	40
56	0.63	0.55	0.59	40
57	0.73	0.60	0.66	40
58	0.62	0.62	0.62	40
59	0.53	0.65	0.58	40
60	0.79	0.82	0.80	40
61	0.57	0.65	0.60	40
62	0.53	0.68	0.59	40
63	0.82	0.78	0.79	40
64	0.53	0.65	0.58	40
65	0.34	0.45	0.39	40
66	0.57	0.53	0.55	40
67	0.58	0.38	0.45	40
68	0.68	0.33	0.44	40
69	0.00	0.00	0.00	40
70	0.72	0.78	0.75	40
71	0.81	0.85	0.83	40
72	1.00	0.03	0.05	40
73	0.77	0.82	0.80	40
74	0.35	0.90	0.50	40
75	0.60	0.72	0.66	40
76	0.69	0.62	0.66	40
accuracy			0.64	3080
macro avg	0.66	0.64	0.63	3080
weighted avg	0.66	0.64	0.63	3080

```
The number of misclassifications: 1100
Proportion of misclassifications: 35.71%
Input Text: way know card arrive
Actual Label: 11
Predicted Label: 12
```

```
Input Text: card arrived yet
Actual Label: 11
Predicted Label: 13

Input Text: get card
Actual Label: 11
Predicted Label: 54

Input Text: received card
Actual Label: 11
Predicted Label: 12

Input Text: normal wait week new card
Actual Label: 11
Predicted Label: 56

Input Text: long card delivery take
Actual Label: 11
Predicted Label: 12

Input Text: still dont card weeks
Actual Label: 11
Predicted Label: 54

Input Text: still waiting card week ok
Actual Label: 11
Predicted Label: 56

Input Text: waiting longer expected bank card could provide information arrive
Actual Label: 11
Predicted Label: 62

Input Text: ive waiting longer expected card
Actual Label: 11
Predicted Label: 26

Input Text: card still hasnt arrived weeks lost
Actual Label: 11
Predicted Label: 13

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 41

Input Text: ordered card weeks ago still isnt
Actual Label: 11
Predicted Label: 12
```

```
Input Text: card arrived yet
Actual Label: 11
Predicted Label: 13

Input Text: think something went wrong card delivery havent received yet
Actual Label: 11
Predicted Label: 62

Input Text: expecting new card wondering havent received yet
Actual Label: 11
Predicted Label: 5

Input Text: know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: ordered card still havent received two weeks
Actual Label: 11
Predicted Label: 12

Input Text: wont card show app
Actual Label: 13
Predicted Label: 56

Input Text: add card account
Actual Label: 13
Predicted Label: 54

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 31

Input Text: view card received app
Actual Label: 13
Predicted Label: 11

Input Text: website go link card
Actual Label: 13
Predicted Label: 62

Input Text: app doesnt show card received
Actual Label: 13
Predicted Label: 56

Input Text: international exchange rates
Actual Label: 32
Predicted Label: 75
```

```
Input Text: please advise exchange rate
Actual Label: 32
Predicted Label: 17

Input Text: good time exchange
Actual Label: 32
Predicted Label: 61

Input Text: much get exchange rate
Actual Label: 32
Predicted Label: 76

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: rate low sure using right exchange rate
Actual Label: 17
Predicted Label: 76
```

1.4.2 LSTM (with dropout)

[ ]:
```python
# Define the output dimension for the embedding layer and hidden units
embedding_output_dim = 100 # Random number
hidden_unit = 30 # Random number
nlabel = 77 # number of classes

# Build the baseline model
dropout_model = keras.models.Sequential()
dropout_model.add(layers.Embedding(voca_size, embedding_output_dim))
dropout_model.add(layers.LSTM(hidden_unit, dropout=0.2)) # Random number
dropout_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
dropout_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy'])

# Summary the model
dropout_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 100)         208800


 lstm_1 (LSTM)               (None, 30)                15720
```

```
 dense_1 (Dense)             (None, 77)                2387

=================================================================
Total params: 226907 (886.36 KB)
Trainable params: 226907 (886.36 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'dropout_LSTM_embedding_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = dropout_model.fit(
    X_train_padded, y_train,
    epochs = 100, # Random number
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Total training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 10s 29ms/step - loss: 4.2711 -
accuracy: 0.0182 - val_loss: 4.0823 - val_accuracy: 0.0338
Epoch 2/100
230/230 [==============================] - 7s 30ms/step - loss: 4.0866 -
```

```
accuracy: 0.0283 - val_loss: 4.1577 - val_accuracy: 0.0278
Epoch 3/100
230/230 [==============================] - 8s 34ms/step - loss: 4.0583 -
accuracy: 0.0252 - val_loss: 4.0054 - val_accuracy: 0.0327
Epoch 4/100
230/230 [==============================] - 11s 47ms/step - loss: 3.9583 -
accuracy: 0.0294 - val_loss: 3.8965 - val_accuracy: 0.0463
Epoch 5/100
230/230 [==============================] - 9s 38ms/step - loss: 3.8102 -
accuracy: 0.0438 - val_loss: 3.7361 - val_accuracy: 0.0550
Epoch 6/100
230/230 [==============================] - 7s 31ms/step - loss: 3.6619 -
accuracy: 0.0636 - val_loss: 3.6422 - val_accuracy: 0.0659
Epoch 7/100
230/230 [==============================] - 8s 33ms/step - loss: 3.5815 -
accuracy: 0.0679 - val_loss: 3.6077 - val_accuracy: 0.0680
Epoch 8/100
230/230 [==============================] - 11s 50ms/step - loss: 3.4941 -
accuracy: 0.0760 - val_loss: 3.5286 - val_accuracy: 0.0724
Epoch 9/100
230/230 [==============================] - 11s 46ms/step - loss: 3.4428 -
accuracy: 0.0825 - val_loss: 3.4720 - val_accuracy: 0.0838
Epoch 10/100
230/230 [==============================] - 11s 50ms/step - loss: 3.3883 -
accuracy: 0.0962 - val_loss: 3.3982 - val_accuracy: 0.0920
Epoch 11/100
230/230 [==============================] - 10s 44ms/step - loss: 3.3377 -
accuracy: 0.1033 - val_loss: 3.4235 - val_accuracy: 0.0942
Epoch 12/100
230/230 [==============================] - 9s 41ms/step - loss: 3.3131 -
accuracy: 0.1075 - val_loss: 3.3779 - val_accuracy: 0.1062
Epoch 13/100
230/230 [==============================] - 10s 45ms/step - loss: 3.2711 -
accuracy: 0.1108 - val_loss: 3.3510 - val_accuracy: 0.1127
Epoch 14/100
230/230 [==============================] - 6s 25ms/step - loss: 3.2236 -
accuracy: 0.1207 - val_loss: 3.3295 - val_accuracy: 0.1165
Epoch 15/100
230/230 [==============================] - 8s 33ms/step - loss: 3.2137 -
accuracy: 0.1284 - val_loss: 3.3404 - val_accuracy: 0.1230
Epoch 16/100
230/230 [==============================] - 6s 25ms/step - loss: 3.1697 -
accuracy: 0.1374 - val_loss: 3.3129 - val_accuracy: 0.1263
Epoch 17/100
230/230 [==============================] - 7s 31ms/step - loss: 3.1523 -
accuracy: 0.1410 - val_loss: 3.2974 - val_accuracy: 0.1345
Epoch 18/100
230/230 [==============================] - 11s 47ms/step - loss: 3.1204 -
```

```
accuracy: 0.1504 - val_loss: 3.1890 - val_accuracy: 0.1475
Epoch 19/100
230/230 [==============================] - 11s 47ms/step - loss: 2.9976 -
accuracy: 0.1678 - val_loss: 3.0844 - val_accuracy: 0.1579
Epoch 20/100
230/230 [==============================] - 11s 46ms/step - loss: 2.8706 -
accuracy: 0.1808 - val_loss: 2.9859 - val_accuracy: 0.1688
Epoch 21/100
230/230 [==============================] - 11s 47ms/step - loss: 2.8179 -
accuracy: 0.1899 - val_loss: 3.0967 - val_accuracy: 0.1590
Epoch 22/100
230/230 [==============================] - 7s 30ms/step - loss: 2.7734 -
accuracy: 0.1978 - val_loss: 2.9003 - val_accuracy: 0.1873
Epoch 23/100
230/230 [==============================] - 8s 33ms/step - loss: 2.6465 -
accuracy: 0.2169 - val_loss: 2.8283 - val_accuracy: 0.2150
Epoch 24/100
230/230 [==============================] - 7s 31ms/step - loss: 2.5462 -
accuracy: 0.2343 - val_loss: 2.7194 - val_accuracy: 0.2226
Epoch 25/100
230/230 [==============================] - 8s 35ms/step - loss: 2.4750 -
accuracy: 0.2473 - val_loss: 2.6554 - val_accuracy: 0.2395
Epoch 26/100
230/230 [==============================] - 5s 23ms/step - loss: 2.3949 -
accuracy: 0.2691 - val_loss: 2.6915 - val_accuracy: 0.2368
Epoch 27/100
230/230 [==============================] - 7s 29ms/step - loss: 2.3553 -
accuracy: 0.2731 - val_loss: 2.5588 - val_accuracy: 0.2695
Epoch 28/100
230/230 [==============================] - 6s 26ms/step - loss: 2.2898 -
accuracy: 0.2919 - val_loss: 2.5037 - val_accuracy: 0.2793
Epoch 29/100
230/230 [==============================] - 5s 24ms/step - loss: 2.1817 -
accuracy: 0.3185 - val_loss: 2.4204 - val_accuracy: 0.2885
Epoch 30/100
230/230 [==============================] - 8s 35ms/step - loss: 2.0855 -
accuracy: 0.3339 - val_loss: 2.4059 - val_accuracy: 0.2907
Epoch 31/100
230/230 [==============================] - 6s 26ms/step - loss: 2.0326 -
accuracy: 0.3464 - val_loss: 2.3102 - val_accuracy: 0.3103
Epoch 32/100
230/230 [==============================] - 7s 29ms/step - loss: 1.9632 -
accuracy: 0.3765 - val_loss: 2.2238 - val_accuracy: 0.3538
Epoch 33/100
230/230 [==============================] - 7s 30ms/step - loss: 1.8574 -
accuracy: 0.4063 - val_loss: 2.2024 - val_accuracy: 0.3664
Epoch 34/100
230/230 [==============================] - 5s 22ms/step - loss: 1.7783 -
```

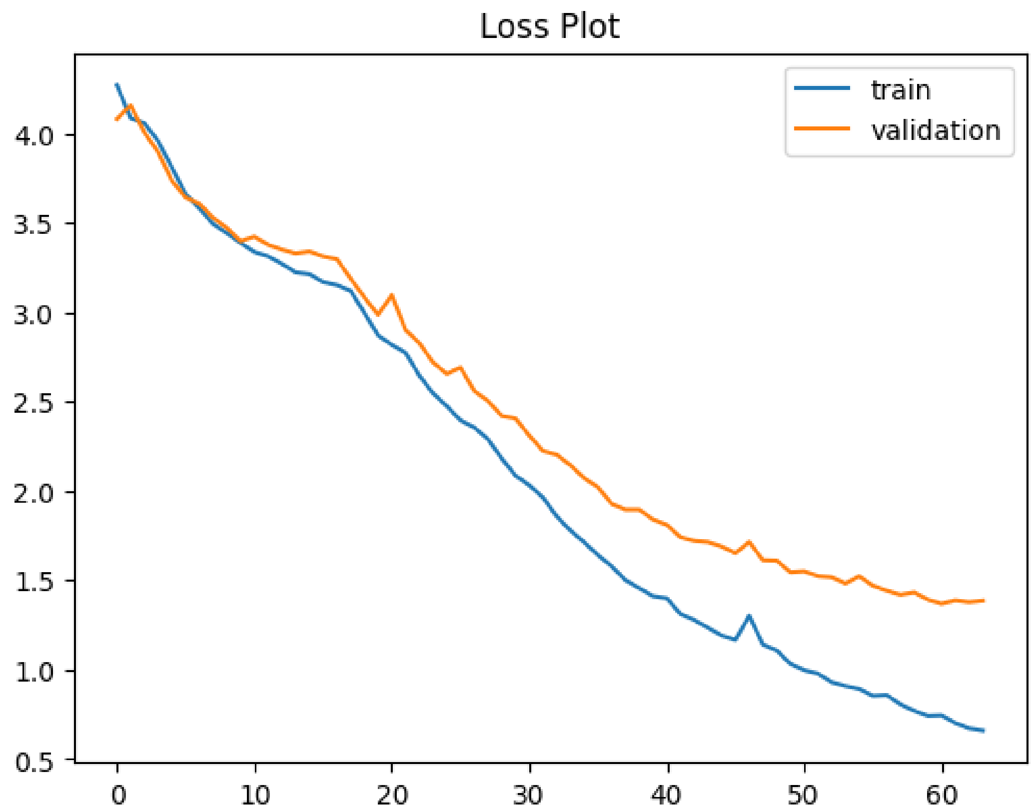
```
accuracy: 0.4283 - val_loss: 2.1439 - val_accuracy: 0.3685
Epoch 35/100
230/230 [==============================] - 8s 35ms/step - loss: 1.7123 -
accuracy: 0.4396 - val_loss: 2.0713 - val_accuracy: 0.3996
Epoch 36/100
230/230 [==============================] - 5s 23ms/step - loss: 1.6415 -
accuracy: 0.4639 - val_loss: 2.0214 - val_accuracy: 0.4148
Epoch 37/100
230/230 [==============================] - 7s 29ms/step - loss: 1.5779 -
accuracy: 0.4936 - val_loss: 1.9272 - val_accuracy: 0.4339
Epoch 38/100
230/230 [==============================] - 6s 28ms/step - loss: 1.5008 -
accuracy: 0.5144 - val_loss: 1.8958 - val_accuracy: 0.4502
Epoch 39/100
230/230 [==============================] - 6s 24ms/step - loss: 1.4564 -
accuracy: 0.5238 - val_loss: 1.8957 - val_accuracy: 0.4638
Epoch 40/100
230/230 [==============================] - 8s 36ms/step - loss: 1.4113 -
accuracy: 0.5411 - val_loss: 1.8400 - val_accuracy: 0.4703
Epoch 41/100
230/230 [==============================] - 8s 34ms/step - loss: 1.3995 -
accuracy: 0.5502 - val_loss: 1.8106 - val_accuracy: 0.4850
Epoch 42/100
230/230 [==============================] - 11s 48ms/step - loss: 1.3116 -
accuracy: 0.5784 - val_loss: 1.7411 - val_accuracy: 0.5019
Epoch 43/100
230/230 [==============================] - 6s 26ms/step - loss: 1.2781 -
accuracy: 0.5828 - val_loss: 1.7210 - val_accuracy: 0.5166
Epoch 44/100
230/230 [==============================] - 9s 40ms/step - loss: 1.2346 -
accuracy: 0.5992 - val_loss: 1.7140 - val_accuracy: 0.5182
Epoch 45/100
230/230 [==============================] - 6s 28ms/step - loss: 1.1909 -
accuracy: 0.6100 - val_loss: 1.6889 - val_accuracy: 0.5433
Epoch 46/100
230/230 [==============================] - 6s 26ms/step - loss: 1.1664 -
accuracy: 0.6297 - val_loss: 1.6523 - val_accuracy: 0.5422
Epoch 47/100
230/230 [==============================] - 8s 35ms/step - loss: 1.3018 -
accuracy: 0.5883 - val_loss: 1.7145 - val_accuracy: 0.5514
Epoch 48/100
230/230 [==============================] - 6s 25ms/step - loss: 1.1404 -
accuracy: 0.6510 - val_loss: 1.6129 - val_accuracy: 0.5504
Epoch 49/100
230/230 [==============================] - 7s 32ms/step - loss: 1.1076 -
accuracy: 0.6570 - val_loss: 1.6105 - val_accuracy: 0.5607
Epoch 50/100
230/230 [==============================] - 6s 25ms/step - loss: 1.0322 -
```

```
accuracy: 0.6862 - val_loss: 1.5449 - val_accuracy: 0.5955
Epoch 51/100
230/230 [==============================] - 6s 28ms/step - loss: 0.9976 -
accuracy: 0.7028 - val_loss: 1.5503 - val_accuracy: 0.6026
Epoch 52/100
230/230 [==============================] - 7s 30ms/step - loss: 0.9781 -
accuracy: 0.7013 - val_loss: 1.5233 - val_accuracy: 0.6102
Epoch 53/100
230/230 [==============================] - 5s 23ms/step - loss: 0.9301 -
accuracy: 0.7280 - val_loss: 1.5173 - val_accuracy: 0.6102
Epoch 54/100
230/230 [==============================] - 8s 33ms/step - loss: 0.9084 -
accuracy: 0.7360 - val_loss: 1.4822 - val_accuracy: 0.6238
Epoch 55/100
230/230 [==============================] - 6s 24ms/step - loss: 0.8920 -
accuracy: 0.7457 - val_loss: 1.5222 - val_accuracy: 0.6113
Epoch 56/100
230/230 [==============================] - 8s 35ms/step - loss: 0.8535 -
accuracy: 0.7542 - val_loss: 1.4702 - val_accuracy: 0.6287
Epoch 57/100
230/230 [==============================] - 7s 29ms/step - loss: 0.8575 -
accuracy: 0.7518 - val_loss: 1.4424 - val_accuracy: 0.6353
Epoch 58/100
230/230 [==============================] - 6s 24ms/step - loss: 0.8061 -
accuracy: 0.7735 - val_loss: 1.4190 - val_accuracy: 0.6413
Epoch 59/100
230/230 [==============================] - 8s 34ms/step - loss: 0.7699 -
accuracy: 0.7806 - val_loss: 1.4324 - val_accuracy: 0.6478
Epoch 60/100
230/230 [==============================] - 5s 24ms/step - loss: 0.7414 -
accuracy: 0.7927 - val_loss: 1.3918 - val_accuracy: 0.6511
Epoch 61/100
230/230 [==============================] - 6s 28ms/step - loss: 0.7441 -
accuracy: 0.7916 - val_loss: 1.3689 - val_accuracy: 0.6576
Epoch 62/100
230/230 [==============================] - 6s 28ms/step - loss: 0.7009 -
accuracy: 0.8045 - val_loss: 1.3883 - val_accuracy: 0.6543
Epoch 63/100
230/230 [==============================] - 6s 25ms/step - loss: 0.6720 -
accuracy: 0.8128 - val_loss: 1.3779 - val_accuracy: 0.6565
Epoch 64/100
230/230 [==============================] - 8s 37ms/step - loss: 0.6601 -
accuracy: 0.8200 - val_loss: 1.3864 - val_accuracy: 0.6614
Total training time: 484.262811422348 seconds
```

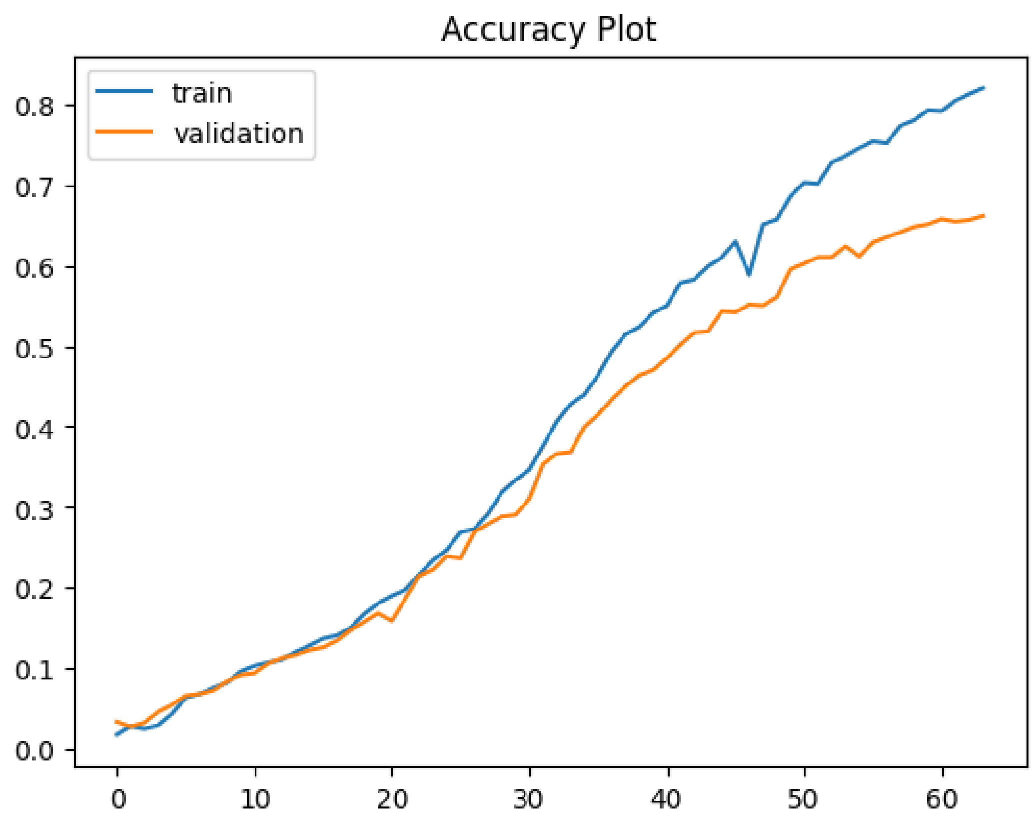
[ ]:

```
# Plot the loss
plt.title('Loss Plot')
```

```
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 8ms/step - loss: 1.3496 - accuracy:
0.6834
Test Loss: 1.349645972251892
Test Accuracy: 68.34
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 2s 9ms/step
Precision: 68.85
Recall: 68.34
F1 Score: 67.45
```

[ ]:
```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.77      0.90      0.83        40
           1       0.89      0.85      0.87        40
           2       0.83      0.97      0.90        40
           3       0.80      0.60      0.69        40
           4       0.79      0.82      0.80        40
           5       0.54      0.78      0.64        40
```

6	0.66	0.72	0.69	40
7	0.66	0.72	0.69	40
8	0.75	0.82	0.79	40
9	0.92	0.85	0.88	40
10	0.55	0.15	0.24	40
11	0.70	0.78	0.74	40
12	0.71	0.60	0.65	40
13	0.94	0.80	0.86	40
14	0.38	0.60	0.46	40
15	0.69	0.88	0.77	40
16	0.58	0.70	0.64	40
17	0.82	0.82	0.82	40
18	0.71	0.68	0.69	40
19	0.87	0.82	0.85	40
20	0.59	0.47	0.53	40
21	0.49	0.85	0.62	40
22	0.56	0.57	0.57	40
23	1.00	0.70	0.82	40
24	0.92	0.85	0.88	40
25	0.62	0.75	0.68	40
26	0.43	0.65	0.51	40
27	0.66	0.62	0.64	40
28	0.77	0.68	0.72	40
29	0.79	0.75	0.77	40
30	0.66	0.95	0.78	40
31	0.67	0.72	0.70	40
32	0.82	0.93	0.87	40
33	0.70	0.82	0.76	40
34	0.77	0.85	0.81	40
35	0.71	0.72	0.72	40
36	0.76	0.65	0.70	40
37	0.64	0.70	0.67	40
38	0.60	0.30	0.40	40
39	0.68	0.70	0.69	40
40	0.69	0.85	0.76	40
41	0.50	0.15	0.23	40
42	0.92	0.88	0.90	40
43	0.65	0.60	0.62	40
44	0.93	0.97	0.95	40
45	0.79	0.78	0.78	40
46	0.81	0.88	0.84	40
47	0.42	0.55	0.47	40
48	0.40	0.45	0.42	40
49	0.88	0.70	0.78	40
50	0.62	0.65	0.63	40
51	0.67	0.78	0.72	40
52	0.62	0.65	0.63	40
53	0.82	0.78	0.79	40

```
          54       0.55      0.65      0.60        40
          55       0.94      0.78      0.85        40
          56       0.66      0.47      0.55        40
          57       0.00      0.00      0.00        40
          58       0.80      0.70      0.75        40
          59       0.32      0.45      0.37        40
          60       0.94      0.75      0.83        40
          61       0.52      0.62      0.57        40
          62       0.53      0.57      0.55        40
          63       0.79      0.85      0.82        40
          64       0.67      0.80      0.73        40
          65       0.69      0.60      0.64        40
          66       0.48      0.25      0.33        40
          67       0.73      0.68      0.70        40
          68       0.35      0.55      0.43        40
          69       0.62      0.25      0.36        40
          70       0.84      0.95      0.89        40
          71       0.84      0.95      0.89        40
          72       0.83      0.38      0.52        40
          73       0.76      0.88      0.81        40
          74       0.57      0.50      0.53        40
          75       0.62      0.60      0.61        40
          76       0.83      0.62      0.71        40

    accuracy                           0.68      3080
   macro avg       0.69      0.68      0.67      3080
weighted avg       0.69      0.68      0.67      3080

The number of misclassifications: 975
Proportion of misclassifications: 31.66%
Input Text: locate card
Actual Label: 11
Predicted Label: 39

Input Text: way know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: get card
Actual Label: 11
Predicted Label: 43

Input Text: card still hasnt arrived weeks lost
Actual Label: 11
Predicted Label: 0

Input Text: get card yet lost
Actual Label: 11
```

```
Predicted Label: 0

Input Text: think something went wrong card delivery havent received yet
Actual Label: 11
Predicted Label: 43

Input Text: expecting new card wondering havent received yet
Actual Label: 11
Predicted Label: 66

Input Text: know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: ordered card still havent received two weeks
Actual Label: 11
Predicted Label: 47

Input Text: readd card app
Actual Label: 13
Predicted Label: 0

Input Text: add card account
Actual Label: 13
Predicted Label: 24

Input Text: view card received app
Actual Label: 13
Predicted Label: 11

Input Text: ive received card need know sync app
Actual Label: 13
Predicted Label: 11

Input Text: app doesnt show card received
Actual Label: 13
Predicted Label: 11

Input Text: way make old card usable app
Actual Label: 13
Predicted Label: 54

Input Text: need go app enter card info
Actual Label: 13
Predicted Label: 11

Input Text: link another card account
Actual Label: 13
```

```
Predicted Label: 0

Input Text: often exchange rates change
Actual Label: 32
Predicted Label: 33

Input Text: good time exchange
Actual Label: 32
Predicted Label: 31

Input Text: currencies exchange rate calculated
Actual Label: 32
Predicted Label: 33

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: rate low sure using right exchange rate
Actual Label: 17
Predicted Label: 76

Input Text: rate applied foreign purchase incorrect
Actual Label: 17
Predicted Label: 2

Input Text: charged
Actual Label: 17
Predicted Label: 15

Input Text: charged wrong currency exchange purchase
Actual Label: 17
Predicted Label: 31

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 2

Input Text: paid something foreign currency noticed exchange rate incorrect
Actual Label: 17
Predicted Label: 76

Input Text: fee dont recognize statement
Actual Label: 34
Predicted Label: 15

Input Text: explain random charge
Actual Label: 34
```

```
Predicted Label: 16

Input Text: transaction credited
Actual Label: 34
Predicted Label: 64
```

1.4.3 Hyperparameter tuning

```
[ ]: # The code for hyperparameter tuning is derived from the Tensorflow website.
     # (https://www.tensorflow.org/tutorials/keras/keras_tuner)

     # Import and install libraries for hyperparameter tuning
     import IPython
     !pip install -q -U keras-tuner
     import kerastuner as kt
```

```
                              129.1/129.1

kB 1.4 MB/s eta 0:00:00


<ipython-input-81-8b29936803b5>:7: DeprecationWarning: `import kerastuner` is
deprecated, please use `import keras_tuner`.
  import kerastuner as kt
```

```
[ ]: # The code for hyperparameter tuning is derived from the Tensorflow website.
     # (https://www.tensorflow.org/tutorials/keras/keras_tuner)

     # Define the model for hyperparameter tuning
     def model_builder(hp):
       model = keras.models.Sequential()
       model.add(layers.Embedding(voca_size, embedding_output_dim)) # Use the same␣
      ↪dimension from the baseline model
       hp_units = hp.Int('units', min_value = 20, max_value = 50, step = 10) # Set␣
      ↪up the hyperparameters
       model.add(layers.LSTM(units = hp_units)) # We will check the optimal hidden␣
      ↪unit for the LSTM layer
       model.add(layers.Dense(nlabel, activation='softmax'))

       hp_learning_rate = hp.Choice('learning_rate', values = [0.01, 0.001, 0.0001])␣
      ↪# Set up the hyperparameters
       model.compile(optimizer = keras.optimizers.Adam(learning_rate =␣
      ↪hp_learning_rate), # We will check the optimal learning rate
                     loss = 'sparse_categorical_crossentropy',
                     metrics = ['accuracy'])
       return model
```

```
[ ]: # The code for hyperparameter tuning is derived from the Tensorflow website.
     # (https://www.tensorflow.org/tutorials/keras/keras_tuner)

     # Specify the tuner
     tuner = kt.Hyperband(model_builder,
                          objective = 'val_accuracy',
                          max_epochs = 100)
```

```
[ ]: # The code for hyperparameter tuning is derived from the Tensorflow website.
     # (https://www.tensorflow.org/tutorials/keras/keras_tuner)

     # Set up a callback for early stopping
     stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
```

```
[ ]: # The code for hyperparameter tuning is derived from the Tensorflow website.
     # (https://www.tensorflow.org/tutorials/keras/keras_tuner)

     # Run the tuner
     tuner.search(X_train_padded, y_train, epochs = 100, validation_data =␣
      ↪(X_val_padded, y_val), callbacks = [stop_early])

     # Get the optimal hyperparameters
     best_hps = tuner.get_best_hyperparameters(num_trials = 1)[0]

     print(f"The optimal number of units: {best_hps.get('units')}. The optimal␣
      ↪learning rate: {best_hps.get('learning_rate')}.")
```

```
The optimal number of units: 40. The optimal learning rate: 0.001.
```

1.4.4 Tuned LSTM

```
[ ]: # Define the output dimension for the embedding layer and hidden units
     embedding_output_dim = 100 # Random number
     hidden_unit = 40 # Number from hyperparameter tuning
     nlabel = 77 # number of classes

     # Build the baseline model
     tuned_model = keras.models.Sequential()
     tuned_model.add(layers.Embedding(voca_size, embedding_output_dim))
     tuned_model.add(layers.LSTM(hidden_unit))
     tuned_model.add(layers.Dense(nlabel, activation='softmax'))

     # Compile the model
     tuned_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
      ↪metrics=['accuracy']) # 0.001 is the default of Adam

     # Summary the model
```

```
tuned_model.summary()
```

```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, None, 100)         208800

 lstm_2 (LSTM)               (None, 40)                22560

 dense_2 (Dense)             (None, 77)                3157

=================================================================
Total params: 234517 (916.08 KB)
Trainable params: 234517 (916.08 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'tuned_LSTM_embedding_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = tuned_model.fit(
    X_train_padded, y_train,
    epochs = 100, # Random number
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()
```

```python
# Measure the training time
training_time = end_time - start_time
print("Total training time:", training_time, "seconds")
```

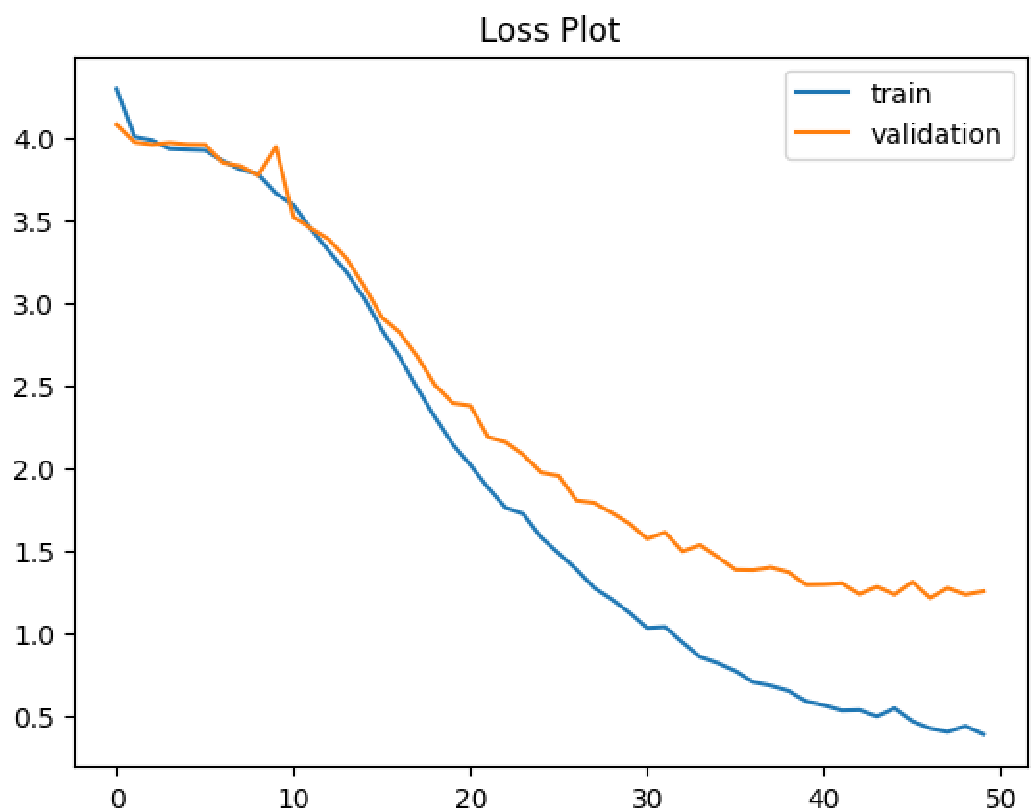
```
Epoch 1/100
230/230 [==============================] - 9s 28ms/step - loss: 4.2946 -
accuracy: 0.0165 - val_loss: 4.0789 - val_accuracy: 0.0272
Epoch 2/100
230/230 [==============================] - 8s 34ms/step - loss: 4.0052 -
accuracy: 0.0297 - val_loss: 3.9721 - val_accuracy: 0.0321
Epoch 3/100
230/230 [==============================] - 5s 23ms/step - loss: 3.9831 -
accuracy: 0.0282 - val_loss: 3.9574 - val_accuracy: 0.0299
Epoch 4/100
230/230 [==============================] - 6s 27ms/step - loss: 3.9313 -
accuracy: 0.0282 - val_loss: 3.9680 - val_accuracy: 0.0338
Epoch 5/100
230/230 [==============================] - 8s 36ms/step - loss: 3.9267 -
accuracy: 0.0274 - val_loss: 3.9583 - val_accuracy: 0.0310
Epoch 6/100
230/230 [==============================] - 6s 25ms/step - loss: 3.9227 -
accuracy: 0.0320 - val_loss: 3.9566 - val_accuracy: 0.0359
Epoch 7/100
230/230 [==============================] - 7s 33ms/step - loss: 3.8577 -
accuracy: 0.0359 - val_loss: 3.8474 - val_accuracy: 0.0463
Epoch 8/100
230/230 [==============================] - 5s 23ms/step - loss: 3.8067 -
accuracy: 0.0388 - val_loss: 3.8269 - val_accuracy: 0.0397
Epoch 9/100
230/230 [==============================] - 7s 31ms/step - loss: 3.7778 -
accuracy: 0.0406 - val_loss: 3.7698 - val_accuracy: 0.0474
Epoch 10/100
230/230 [==============================] - 6s 24ms/step - loss: 3.6627 -
accuracy: 0.0501 - val_loss: 3.9440 - val_accuracy: 0.0419
Epoch 11/100
230/230 [==============================] - 5s 24ms/step - loss: 3.5898 -
accuracy: 0.0542 - val_loss: 3.5175 - val_accuracy: 0.0648
Epoch 12/100
230/230 [==============================] - 8s 36ms/step - loss: 3.4428 -
accuracy: 0.0672 - val_loss: 3.4485 - val_accuracy: 0.0784
Epoch 13/100
230/230 [==============================] - 13s 59ms/step - loss: 3.3126 -
accuracy: 0.0894 - val_loss: 3.3826 - val_accuracy: 0.1056
Epoch 14/100
230/230 [==============================] - 14s 61ms/step - loss: 3.1820 -
accuracy: 0.1115 - val_loss: 3.2661 - val_accuracy: 0.1225
```

```
Epoch 15/100
230/230 [==============================] - 14s 62ms/step - loss: 3.0260 -
accuracy: 0.1417 - val_loss: 3.0991 - val_accuracy: 0.1410
Epoch 16/100
230/230 [==============================] - 14s 62ms/step - loss: 2.8354 -
accuracy: 0.1756 - val_loss: 2.9109 - val_accuracy: 0.1824
Epoch 17/100
230/230 [==============================] - 9s 40ms/step - loss: 2.6705 -
accuracy: 0.2081 - val_loss: 2.8185 - val_accuracy: 0.2009
Epoch 18/100
230/230 [==============================] - 6s 25ms/step - loss: 2.4823 -
accuracy: 0.2554 - val_loss: 2.6742 - val_accuracy: 0.2640
Epoch 19/100
230/230 [==============================] - 7s 28ms/step - loss: 2.3056 -
accuracy: 0.3056 - val_loss: 2.5004 - val_accuracy: 0.2847
Epoch 20/100
230/230 [==============================] - 7s 29ms/step - loss: 2.1421 -
accuracy: 0.3471 - val_loss: 2.3917 - val_accuracy: 0.3130
Epoch 21/100
230/230 [==============================] - 6s 25ms/step - loss: 2.0168 -
accuracy: 0.3855 - val_loss: 2.3748 - val_accuracy: 0.3435
Epoch 22/100
230/230 [==============================] - 8s 35ms/step - loss: 1.8786 -
accuracy: 0.4239 - val_loss: 2.1859 - val_accuracy: 0.4028
Epoch 23/100
230/230 [==============================] - 6s 25ms/step - loss: 1.7576 -
accuracy: 0.4608 - val_loss: 2.1545 - val_accuracy: 0.3990
Epoch 24/100
230/230 [==============================] - 7s 29ms/step - loss: 1.7189 -
accuracy: 0.4763 - val_loss: 2.0777 - val_accuracy: 0.4366
Epoch 25/100
230/230 [==============================] - 6s 27ms/step - loss: 1.5775 -
accuracy: 0.5347 - val_loss: 1.9711 - val_accuracy: 0.4736
Epoch 26/100
230/230 [==============================] - 5s 23ms/step - loss: 1.4813 -
accuracy: 0.5585 - val_loss: 1.9505 - val_accuracy: 0.4769
Epoch 27/100
230/230 [==============================] - 9s 38ms/step - loss: 1.3838 -
accuracy: 0.5996 - val_loss: 1.8017 - val_accuracy: 0.5253
Epoch 28/100
230/230 [==============================] - 6s 25ms/step - loss: 1.2723 -
accuracy: 0.6327 - val_loss: 1.7867 - val_accuracy: 0.5329
Epoch 29/100
230/230 [==============================] - 7s 31ms/step - loss: 1.2030 -
accuracy: 0.6576 - val_loss: 1.7268 - val_accuracy: 0.5297
Epoch 30/100
230/230 [==============================] - 6s 25ms/step - loss: 1.1218 -
accuracy: 0.6779 - val_loss: 1.6611 - val_accuracy: 0.5525
```

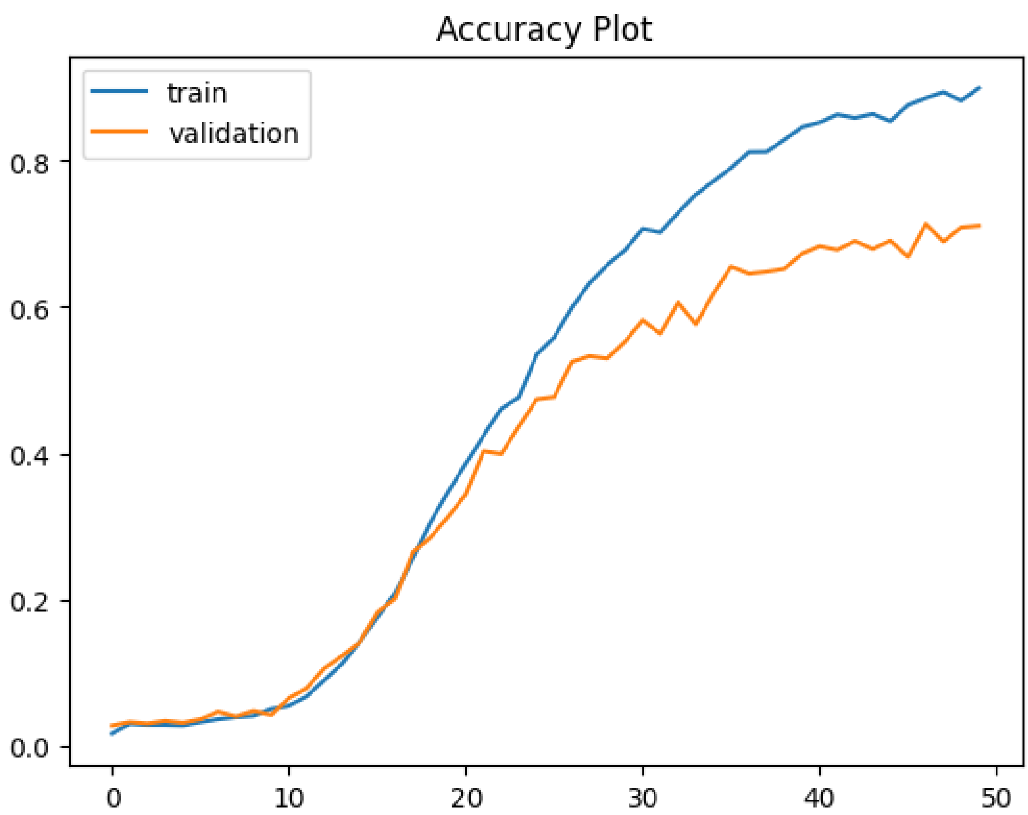
```
Epoch 31/100
230/230 [==============================] - 7s 29ms/step - loss: 1.0285 -
accuracy: 0.7069 - val_loss: 1.5686 - val_accuracy: 0.5819
Epoch 32/100
230/230 [==============================] - 9s 38ms/step - loss: 1.0353 -
accuracy: 0.7022 - val_loss: 1.6081 - val_accuracy: 0.5634
Epoch 33/100
230/230 [==============================] - 5s 24ms/step - loss: 0.9411 -
accuracy: 0.7291 - val_loss: 1.4938 - val_accuracy: 0.6064
Epoch 34/100
230/230 [==============================] - 7s 30ms/step - loss: 0.8535 -
accuracy: 0.7539 - val_loss: 1.5312 - val_accuracy: 0.5765
Epoch 35/100
230/230 [==============================] - 6s 26ms/step - loss: 0.8156 -
accuracy: 0.7723 - val_loss: 1.4590 - val_accuracy: 0.6184
Epoch 36/100
230/230 [==============================] - 6s 24ms/step - loss: 0.7677 -
accuracy: 0.7902 - val_loss: 1.3816 - val_accuracy: 0.6554
Epoch 37/100
230/230 [==============================] - 8s 33ms/step - loss: 0.7014 -
accuracy: 0.8121 - val_loss: 1.3796 - val_accuracy: 0.6456
Epoch 38/100
230/230 [==============================] - 5s 24ms/step - loss: 0.6795 -
accuracy: 0.8124 - val_loss: 1.3965 - val_accuracy: 0.6483
Epoch 39/100
230/230 [==============================] - 7s 31ms/step - loss: 0.6476 -
accuracy: 0.8288 - val_loss: 1.3664 - val_accuracy: 0.6522
Epoch 40/100
230/230 [==============================] - 6s 25ms/step - loss: 0.5844 -
accuracy: 0.8460 - val_loss: 1.2906 - val_accuracy: 0.6728
Epoch 41/100
230/230 [==============================] - 6s 25ms/step - loss: 0.5609 -
accuracy: 0.8526 - val_loss: 1.2927 - val_accuracy: 0.6832
Epoch 42/100
230/230 [==============================] - 8s 33ms/step - loss: 0.5291 -
accuracy: 0.8633 - val_loss: 1.3004 - val_accuracy: 0.6783
Epoch 43/100
230/230 [==============================] - 6s 24ms/step - loss: 0.5322 -
accuracy: 0.8587 - val_loss: 1.2343 - val_accuracy: 0.6903
Epoch 44/100
230/230 [==============================] - 8s 37ms/step - loss: 0.4936 -
accuracy: 0.8641 - val_loss: 1.2784 - val_accuracy: 0.6794
Epoch 45/100
230/230 [==============================] - 5s 24ms/step - loss: 0.5428 -
accuracy: 0.8542 - val_loss: 1.2315 - val_accuracy: 0.6908
Epoch 46/100
230/230 [==============================] - 6s 27ms/step - loss: 0.4640 -
accuracy: 0.8767 - val_loss: 1.3079 - val_accuracy: 0.6685
```

```
Epoch 47/100
230/230 [==============================] - 7s 30ms/step - loss: 0.4211 -
accuracy: 0.8859 - val_loss: 1.2118 - val_accuracy: 0.7137
Epoch 48/100
230/230 [==============================] - 6s 24ms/step - loss: 0.3998 -
accuracy: 0.8940 - val_loss: 1.2706 - val_accuracy: 0.6897
Epoch 49/100
230/230 [==============================] - 8s 34ms/step - loss: 0.4356 -
accuracy: 0.8825 - val_loss: 1.2313 - val_accuracy: 0.7088
Epoch 50/100
230/230 [==============================] - 6s 25ms/step - loss: 0.3856 -
accuracy: 0.8995 - val_loss: 1.2514 - val_accuracy: 0.7109
Total training time: 361.4873790740967 seconds
```

[ ]:
```python
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 7ms/step - loss: 1.1920 - accuracy:
0.7286
Test Loss: 1.1920427083969116
Test Accuracy: 72.86
```

```
[ ]: # Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 1s 6ms/step
Precision: 74.3
Recall: 72.86
F1 Score: 72.14
```

```
[ ]: # Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support
```

0	0.93	0.93	0.93	40
1	0.68	0.75	0.71	40
2	1.00	0.95	0.97	40
3	0.80	0.80	0.80	40
4	0.97	0.85	0.91	40
5	0.58	0.78	0.67	40
6	0.76	0.78	0.77	40
7	0.89	0.78	0.83	40
8	0.65	0.82	0.73	40
9	0.90	0.90	0.90	40
10	0.50	0.40	0.44	40
11	0.65	0.65	0.65	40
12	0.48	0.72	0.58	40
13	0.71	0.85	0.77	40
14	0.63	0.60	0.62	40
15	0.68	0.80	0.74	40
16	0.64	0.62	0.63	40
17	0.74	0.88	0.80	40
18	0.86	0.60	0.71	40
19	0.87	0.82	0.85	40
20	0.59	0.68	0.63	40
21	0.74	0.78	0.76	40
22	0.68	0.65	0.67	40
23	0.81	0.62	0.70	40
24	0.92	0.82	0.87	40
25	0.70	0.65	0.68	40
26	0.58	0.78	0.67	40
27	0.74	0.80	0.77	40
28	0.79	0.78	0.78	40
29	0.70	0.75	0.72	40
30	0.77	0.85	0.81	40
31	0.85	0.85	0.85	40
32	0.94	0.80	0.86	40
33	0.76	0.72	0.74	40
34	0.59	0.72	0.65	40
35	0.69	0.68	0.68	40
36	0.84	0.68	0.75	40
37	0.67	0.55	0.60	40
38	0.75	0.95	0.84	40
39	0.77	0.75	0.76	40
40	0.57	0.97	0.72	40
41	0.67	0.65	0.66	40
42	0.95	0.90	0.92	40
43	0.64	0.72	0.68	40
44	0.97	0.85	0.91	40
45	0.76	0.65	0.70	40
46	0.75	0.75	0.75	40

```
          47       0.65      0.65      0.65        40
          48       0.56      0.55      0.56        40
          49       0.88      0.70      0.78        40
          50       0.70      0.80      0.74        40
          51       0.88      0.75      0.81        40
          52       0.77      0.68      0.72        40
          53       0.66      0.78      0.71        40
          54       0.60      0.78      0.67        40
          55       0.97      0.85      0.91        40
          56       0.88      0.72      0.79        40
          57       0.75      0.82      0.79        40
          58       0.62      0.70      0.66        40
          59       0.75      0.75      0.75        40
          60       0.79      0.75      0.77        40
          61       0.80      0.70      0.75        40
          62       0.68      0.65      0.67        40
          63       0.88      0.90      0.89        40
          64       0.83      0.75      0.79        40
          65       0.69      0.62      0.66        40
          66       0.60      0.60      0.60        40
          67       0.58      0.62      0.60        40
          68       0.00      0.00      0.00        40
          69       1.00      0.03      0.05        40
          70       0.87      0.82      0.85        40
          71       0.89      1.00      0.94        40
          72       0.82      0.23      0.35        40
          73       0.92      0.82      0.87        40
          74       0.37      1.00      0.54        40
          75       0.89      0.80      0.84        40
          76       0.81      0.65      0.72        40

    accuracy                           0.73      3080
   macro avg       0.74      0.73      0.72      3080
weighted avg       0.74      0.73      0.72      3080

The number of misclassifications: 836
Proportion of misclassifications: 27.14%
Input Text: locate card
Actual Label: 11
Predicted Label: 12

Input Text: way know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: card arrived yet
Actual Label: 11
Predicted Label: 12
```

```
Input Text: get card
Actual Label: 11
Predicted Label: 12

Input Text: long card delivery take
Actual Label: 11
Predicted Label: 12

Input Text: still dont card weeks
Actual Label: 11
Predicted Label: 14

Input Text: ive waiting longer expected card
Actual Label: 11
Predicted Label: 14

Input Text: hasnt card delivered
Actual Label: 11
Predicted Label: 12

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 13

Input Text: status card ordered
Actual Label: 11
Predicted Label: 9

Input Text: card arrived yet
Actual Label: 11
Predicted Label: 12

Input Text: know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: im starting think card lost still hasnt arrived help
Actual Label: 11
Predicted Label: 41

Input Text: tracking info available
Actual Label: 11
Predicted Label: 12

Input Text: wont card show app
Actual Label: 13
Predicted Label: 41
```

```
Input Text: add card account
Actual Label: 13
Predicted Label: 38

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 30

Input Text: app doesnt show card received
Actual Label: 13
Predicted Label: 41

Input Text: way make old card usable app
Actual Label: 13
Predicted Label: 54

Input Text: could help reactivate card previously lost found morning jacket
Actual Label: 13
Predicted Label: 42

Input Text: often exchange rates change
Actual Label: 32
Predicted Label: 31

Input Text: good time exchange
Actual Label: 32
Predicted Label: 50

Input Text: exchange rate like app
Actual Label: 32
Predicted Label: 17

Input Text: currencies exchange rate calculated
Actual Label: 32
Predicted Label: 31

Input Text: much get exchange rate
Actual Label: 32
Predicted Label: 17

Input Text: exchange rate would
Actual Label: 32
Predicted Label: 17

Input Text: exchange rate like
Actual Label: 32
Predicted Label: 17
```

```
Input Text: rate get determined
Actual Label: 32
Predicted Label: 76

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: charged
Actual Label: 17
Predicted Label: 34
```

1.4.5 Tuned LSTM (with dropout)

[ ]:
```python
# Define the output dimension for the embedding layer and hidden units
embedding_output_dim = 100 # Random number
hidden_unit = 40 # Number from hyperparameter tuning
nlabel = 77 # number of classes

# Build the baseline model
dropout_tuned_model = keras.models.Sequential()
dropout_tuned_model.add(layers.Embedding(voca_size, embedding_output_dim))
dropout_tuned_model.add(layers.LSTM(hidden_unit, dropout=0.2)) # Random number
dropout_tuned_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
dropout_tuned_model.compile(optimizer='adam',
 ↪loss='sparse_categorical_crossentropy', metrics=['accuracy']) # 0.001 is the
 ↪default of Adam

# Summary the model
dropout_tuned_model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, None, 100)         208800

 lstm_5 (LSTM)               (None, 40)                22560

 dense_5 (Dense)             (None, 77)                3157

=================================================================
Total params: 234517 (916.08 KB)
Trainable params: 234517 (916.08 KB)
```

```
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```python
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'dropout_tuned_LSTM_embedding_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```python
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = dropout_tuned_model.fit(
    X_train_padded, y_train,
    epochs = 100, # Random number
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Total training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 12s 39ms/step - loss: 4.2630 -
accuracy: 0.0189 - val_loss: 4.1256 - val_accuracy: 0.0212
Epoch 2/100
230/230 [==============================] - 7s 31ms/step - loss: 3.9792 -
accuracy: 0.0313 - val_loss: 3.9345 - val_accuracy: 0.0299
Epoch 3/100
230/230 [==============================] - 7s 32ms/step - loss: 3.9507 -
accuracy: 0.0313 - val_loss: 3.9015 - val_accuracy: 0.0305
Epoch 4/100
230/230 [==============================] - 6s 26ms/step - loss: 3.9079 -
```

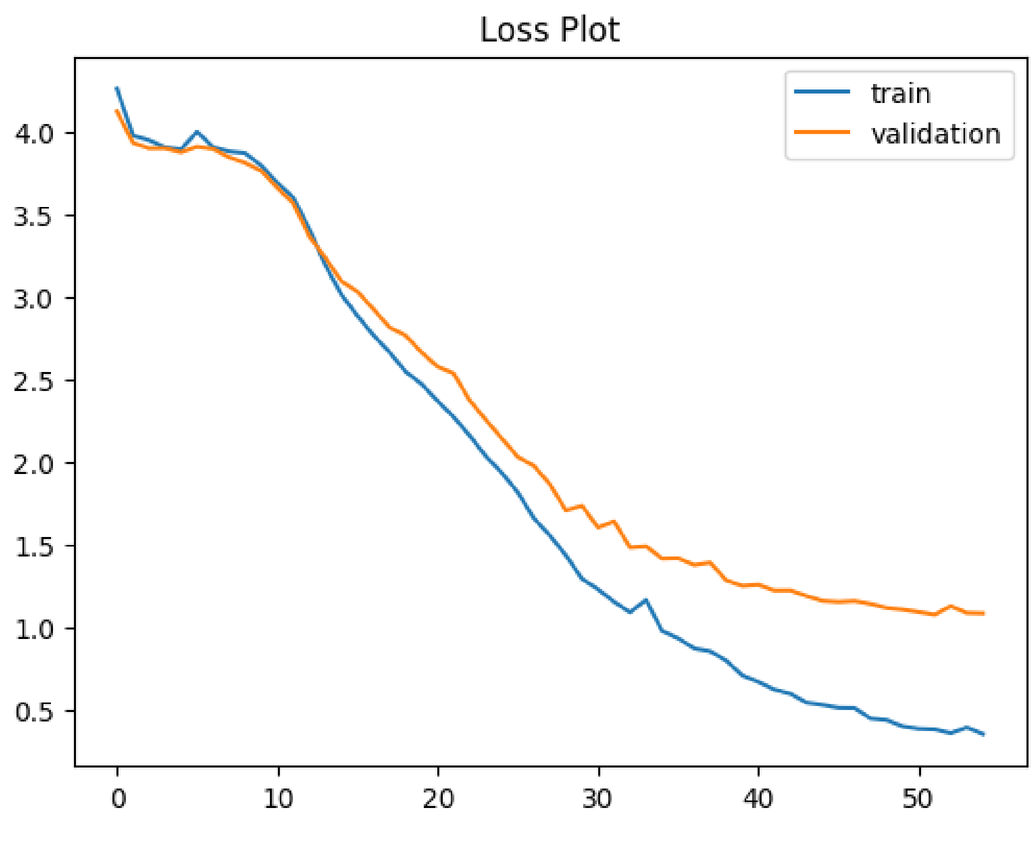
```
accuracy: 0.0289 - val_loss: 3.9015 - val_accuracy: 0.0327
Epoch 5/100
230/230 [==============================] - 8s 34ms/step - loss: 3.8954 -
accuracy: 0.0294 - val_loss: 3.8779 - val_accuracy: 0.0327
Epoch 6/100
230/230 [==============================] - 11s 46ms/step - loss: 4.0020 -
accuracy: 0.0316 - val_loss: 3.9115 - val_accuracy: 0.0343
Epoch 7/100
230/230 [==============================] - 8s 34ms/step - loss: 3.9081 -
accuracy: 0.0285 - val_loss: 3.8978 - val_accuracy: 0.0327
Epoch 8/100
230/230 [==============================] - 5s 24ms/step - loss: 3.8826 -
accuracy: 0.0313 - val_loss: 3.8469 - val_accuracy: 0.0419
Epoch 9/100
230/230 [==============================] - 9s 37ms/step - loss: 3.8721 -
accuracy: 0.0350 - val_loss: 3.8135 - val_accuracy: 0.0414
Epoch 10/100
230/230 [==============================] - 13s 55ms/step - loss: 3.7973 -
accuracy: 0.0422 - val_loss: 3.7652 - val_accuracy: 0.0457
Epoch 11/100
230/230 [==============================] - 10s 44ms/step - loss: 3.6923 -
accuracy: 0.0505 - val_loss: 3.6624 - val_accuracy: 0.0474
Epoch 12/100
230/230 [==============================] - 12s 52ms/step - loss: 3.6041 -
accuracy: 0.0557 - val_loss: 3.5686 - val_accuracy: 0.0729
Epoch 13/100
230/230 [==============================] - 6s 26ms/step - loss: 3.4120 -
accuracy: 0.0785 - val_loss: 3.3685 - val_accuracy: 0.0942
Epoch 14/100
230/230 [==============================] - 8s 35ms/step - loss: 3.1942 -
accuracy: 0.0939 - val_loss: 3.2385 - val_accuracy: 0.0974
Epoch 15/100
230/230 [==============================] - 6s 26ms/step - loss: 3.0150 -
accuracy: 0.1156 - val_loss: 3.0965 - val_accuracy: 0.1241
Epoch 16/100
230/230 [==============================] - 8s 34ms/step - loss: 2.8881 -
accuracy: 0.1334 - val_loss: 3.0342 - val_accuracy: 0.1361
Epoch 17/100
230/230 [==============================] - 6s 26ms/step - loss: 2.7694 -
accuracy: 0.1538 - val_loss: 2.9277 - val_accuracy: 0.1541
Epoch 18/100
230/230 [==============================] - 6s 26ms/step - loss: 2.6670 -
accuracy: 0.1740 - val_loss: 2.8175 - val_accuracy: 0.1807
Epoch 19/100
230/230 [==============================] - 7s 30ms/step - loss: 2.5498 -
accuracy: 0.1992 - val_loss: 2.7681 - val_accuracy: 0.1818
Epoch 20/100
230/230 [==============================] - 6s 27ms/step - loss: 2.4741 -
```

```
accuracy: 0.2164 - val_loss: 2.6662 - val_accuracy: 0.2020
Epoch 21/100
230/230 [==============================] - 8s 36ms/step - loss: 2.3711 -
accuracy: 0.2389 - val_loss: 2.5793 - val_accuracy: 0.2439
Epoch 22/100
230/230 [==============================] - 6s 24ms/step - loss: 2.2750 -
accuracy: 0.2721 - val_loss: 2.5388 - val_accuracy: 0.2330
Epoch 23/100
230/230 [==============================] - 8s 34ms/step - loss: 2.1622 -
accuracy: 0.2974 - val_loss: 2.3749 - val_accuracy: 0.2880
Epoch 24/100
230/230 [==============================] - 6s 26ms/step - loss: 2.0376 -
accuracy: 0.3366 - val_loss: 2.2583 - val_accuracy: 0.3217
Epoch 25/100
230/230 [==============================] - 7s 29ms/step - loss: 1.9376 -
accuracy: 0.3711 - val_loss: 2.1459 - val_accuracy: 0.3636
Epoch 26/100
230/230 [==============================] - 7s 32ms/step - loss: 1.8185 -
accuracy: 0.4249 - val_loss: 2.0330 - val_accuracy: 0.4143
Epoch 27/100
230/230 [==============================] - 6s 27ms/step - loss: 1.6616 -
accuracy: 0.4781 - val_loss: 1.9790 - val_accuracy: 0.4371
Epoch 28/100
230/230 [==============================] - 9s 38ms/step - loss: 1.5570 -
accuracy: 0.5174 - val_loss: 1.8680 - val_accuracy: 0.4823
Epoch 29/100
230/230 [==============================] - 7s 31ms/step - loss: 1.4374 -
accuracy: 0.5553 - val_loss: 1.7090 - val_accuracy: 0.5139
Epoch 30/100
230/230 [==============================] - 8s 36ms/step - loss: 1.2950 -
accuracy: 0.6055 - val_loss: 1.7371 - val_accuracy: 0.5210
Epoch 31/100
230/230 [==============================] - 6s 27ms/step - loss: 1.2320 -
accuracy: 0.6276 - val_loss: 1.6056 - val_accuracy: 0.5700
Epoch 32/100
230/230 [==============================] - 8s 33ms/step - loss: 1.1562 -
accuracy: 0.6654 - val_loss: 1.6430 - val_accuracy: 0.5487
Epoch 33/100
230/230 [==============================] - 6s 27ms/step - loss: 1.0923 -
accuracy: 0.6835 - val_loss: 1.4850 - val_accuracy: 0.6075
Epoch 34/100
230/230 [==============================] - 7s 29ms/step - loss: 1.1673 -
accuracy: 0.6454 - val_loss: 1.4920 - val_accuracy: 0.6026
Epoch 35/100
230/230 [==============================] - 7s 31ms/step - loss: 0.9790 -
accuracy: 0.7133 - val_loss: 1.4176 - val_accuracy: 0.6217
Epoch 36/100
230/230 [==============================] - 6s 25ms/step - loss: 0.9349 -
```

```
accuracy: 0.7284 - val_loss: 1.4201 - val_accuracy: 0.6309
Epoch 37/100
230/230 [==============================] - 8s 37ms/step - loss: 0.8739 -
accuracy: 0.7483 - val_loss: 1.3790 - val_accuracy: 0.6364
Epoch 38/100
230/230 [==============================] - 6s 26ms/step - loss: 0.8562 -
accuracy: 0.7592 - val_loss: 1.3934 - val_accuracy: 0.6434
Epoch 39/100
230/230 [==============================] - 8s 34ms/step - loss: 0.7994 -
accuracy: 0.7736 - val_loss: 1.2857 - val_accuracy: 0.6734
Epoch 40/100
230/230 [==============================] - 6s 26ms/step - loss: 0.7100 -
accuracy: 0.8037 - val_loss: 1.2523 - val_accuracy: 0.6946
Epoch 41/100
230/230 [==============================] - 6s 28ms/step - loss: 0.6713 -
accuracy: 0.8121 - val_loss: 1.2592 - val_accuracy: 0.6903
Epoch 42/100
230/230 [==============================] - 8s 33ms/step - loss: 0.6246 -
accuracy: 0.8307 - val_loss: 1.2244 - val_accuracy: 0.7050
Epoch 43/100
230/230 [==============================] - 6s 28ms/step - loss: 0.5993 -
accuracy: 0.8337 - val_loss: 1.2247 - val_accuracy: 0.7137
Epoch 44/100
230/230 [==============================] - 9s 37ms/step - loss: 0.5454 -
accuracy: 0.8527 - val_loss: 1.1916 - val_accuracy: 0.7137
Epoch 45/100
230/230 [==============================] - 6s 27ms/step - loss: 0.5321 -
accuracy: 0.8545 - val_loss: 1.1619 - val_accuracy: 0.7213
Epoch 46/100
230/230 [==============================] - 9s 37ms/step - loss: 0.5138 -
accuracy: 0.8616 - val_loss: 1.1556 - val_accuracy: 0.7235
Epoch 47/100
230/230 [==============================] - 6s 26ms/step - loss: 0.5126 -
accuracy: 0.8579 - val_loss: 1.1608 - val_accuracy: 0.7289
Epoch 48/100
230/230 [==============================] - 8s 36ms/step - loss: 0.4501 -
accuracy: 0.8816 - val_loss: 1.1435 - val_accuracy: 0.7295
Epoch 49/100
230/230 [==============================] - 6s 28ms/step - loss: 0.4402 -
accuracy: 0.8805 - val_loss: 1.1190 - val_accuracy: 0.7305
Epoch 50/100
230/230 [==============================] - 7s 31ms/step - loss: 0.4019 -
accuracy: 0.8900 - val_loss: 1.1093 - val_accuracy: 0.7382
Epoch 51/100
230/230 [==============================] - 7s 29ms/step - loss: 0.3875 -
accuracy: 0.8908 - val_loss: 1.0944 - val_accuracy: 0.7441
Epoch 52/100
230/230 [==============================] - 6s 27ms/step - loss: 0.3842 -
```

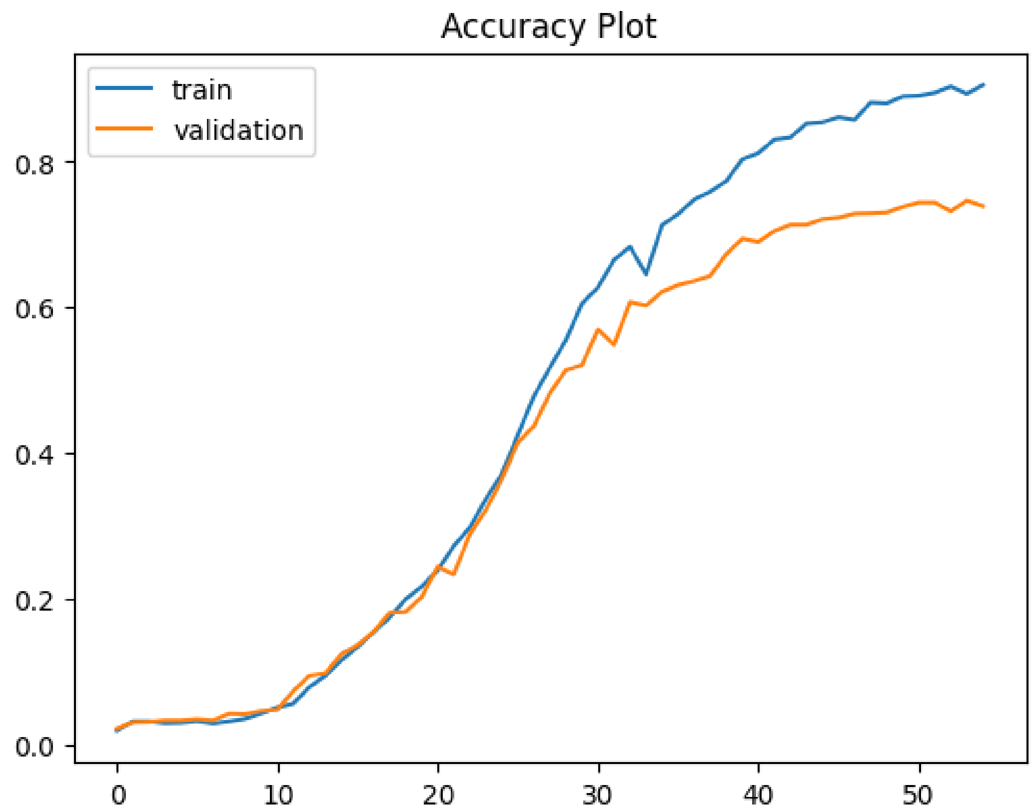
```
accuracy: 0.8950 - val_loss: 1.0782 - val_accuracy: 0.7441
Epoch 53/100
230/230 [==============================] - 8s 35ms/step - loss: 0.3607 -
accuracy: 0.9035 - val_loss: 1.1300 - val_accuracy: 0.7322
Epoch 54/100
230/230 [==============================] - 6s 27ms/step - loss: 0.3953 -
accuracy: 0.8937 - val_loss: 1.0888 - val_accuracy: 0.7469
Epoch 55/100
230/230 [==============================] - 9s 37ms/step - loss: 0.3558 -
accuracy: 0.9055 - val_loss: 1.0863 - val_accuracy: 0.7392
Total training time: 408.2757978439331 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
```

```
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 7ms/step - loss: 1.0432 - accuracy:
0.7659
Test Loss: 1.0432207584381104
Test Accuracy: 76.59
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)
```

```
# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 1s 7ms/step
Precision: 77.65
Recall: 76.59
F1 Score: 76.47
```

[ ]:

```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        40
```

1	1.00	0.95	0.97	40
2	0.97	0.97	0.97	40
3	0.92	0.60	0.73	40
4	0.91	0.80	0.85	40
5	0.55	0.78	0.65	40
6	0.85	0.85	0.85	40
7	0.82	0.78	0.79	40
8	0.87	0.82	0.85	40
9	0.97	0.88	0.92	40
10	0.58	0.45	0.51	40
11	0.85	0.82	0.84	40
12	0.56	0.62	0.59	40
13	0.90	0.93	0.91	40
14	0.77	0.85	0.81	40
15	0.72	0.78	0.75	40
16	0.59	0.55	0.57	40
17	0.85	0.88	0.86	40
18	0.75	0.82	0.79	40
19	0.80	0.82	0.81	40
20	0.67	0.65	0.66	40
21	0.97	0.78	0.86	40
22	0.73	0.55	0.63	40
23	0.97	0.85	0.91	40
24	0.97	0.80	0.88	40
25	0.57	0.78	0.66	40
26	0.64	0.80	0.71	40
27	0.76	0.78	0.77	40
28	0.88	0.75	0.81	40
29	0.74	0.78	0.76	40
30	0.95	0.93	0.94	40
31	0.77	0.75	0.76	40
32	0.88	0.88	0.88	40
33	0.74	0.72	0.73	40
34	0.79	0.68	0.73	40
35	0.71	0.72	0.72	40
36	0.81	0.75	0.78	40
37	0.81	0.62	0.70	40
38	0.82	1.00	0.90	40
39	0.64	0.85	0.73	40
40	0.81	0.95	0.87	40
41	0.69	0.68	0.68	40
42	0.90	0.88	0.89	40
43	0.55	0.70	0.62	40
44	0.98	1.00	0.99	40
45	0.76	0.78	0.77	40
46	0.73	0.82	0.78	40
47	0.63	0.68	0.65	40
48	0.79	0.57	0.67	40

```
          49       0.83      0.75      0.79        40
          50       0.89      0.85      0.87        40
          51       0.70      0.80      0.74        40
          52       0.74      0.72      0.73        40
          53       0.64      0.85      0.73        40
          54       0.62      0.78      0.69        40
          55       0.89      0.82      0.86        40
          56       0.78      0.70      0.74        40
          57       0.95      0.93      0.94        40
          58       0.85      0.70      0.77        40
          59       0.78      0.88      0.82        40
          60       0.76      0.88      0.81        40
          61       0.82      0.68      0.74        40
          62       0.74      0.72      0.73        40
          63       0.62      0.82      0.71        40
          64       0.76      0.78      0.77        40
          65       0.62      0.57      0.60        40
          66       0.69      0.55      0.61        40
          67       0.66      0.68      0.67        40
          68       0.71      0.38      0.49        40
          69       0.50      0.20      0.29        40
          70       0.92      0.90      0.91        40
          71       0.93      0.97      0.95        40
          72       0.81      0.65      0.72        40
          73       0.78      0.88      0.82        40
          74       0.43      0.85      0.57        40
          75       0.82      0.70      0.76        40
          76       0.68      0.65      0.67        40

    accuracy                           0.77      3080
   macro avg       0.78      0.77      0.76      3080
weighted avg       0.78      0.77      0.76      3080

The number of misclassifications: 721
Proportion of misclassifications: 23.41%
Input Text: get card
Actual Label: 11
Predicted Label: 43

Input Text: long card delivery take
Actual Label: 11
Predicted Label: 12

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 41

Input Text: status card ordered
```

```
Actual Label: 11
Predicted Label: 53

Input Text: long new card take arrive
Actual Label: 11
Predicted Label: 12

Input Text: think something went wrong card delivery havent received yet
Actual Label: 11
Predicted Label: 12

Input Text: im starting think card lost still hasnt arrived help
Actual Label: 11
Predicted Label: 18

Input Text: add card account
Actual Label: 13
Predicted Label: 43

Input Text: link credit card
Actual Label: 13
Predicted Label: 47

Input Text: way make old card usable app
Actual Label: 13
Predicted Label: 61

Input Text: good time exchange
Actual Label: 32
Predicted Label: 50

Input Text: exchange rate like app
Actual Label: 32
Predicted Label: 76

Input Text: exchange rate use
Actual Label: 32
Predicted Label: 76

Input Text: much get exchange rate
Actual Label: 32
Predicted Label: 76

Input Text: kind foreign exchange rate get exchange money
Actual Label: 32
Predicted Label: 76

Input Text: made currency exchange think charged
```

Actual Label: 17
Predicted Label: 31

Input Text: rate low sure using right exchange rate
Actual Label: 17
Predicted Label: 76

Input Text: charged
Actual Label: 17
Predicted Label: 15

Input Text: hi dont think exchange rate right need check official interbank exchange please
Actual Label: 17
Predicted Label: 76

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 15

Input Text: im okay fee statement
Actual Label: 34
Predicted Label: 63

Input Text: would like refund extra pound charged
Actual Label: 34
Predicted Label: 63

Input Text: statement extra charges
Actual Label: 34
Predicted Label: 63

Input Text: transaction credited
Actual Label: 34
Predicted Label: 8

Input Text: fee come
Actual Label: 34
Predicted Label: 15

Input Text: many fees statement
Actual Label: 34
Predicted Label: 15

Input Text: euro fee come
Actual Label: 34
Predicted Label: 2

```
Input Text: euro fee statement
Actual Label: 34
Predicted Label: 15

Input Text: extra charge app told aware
Actual Label: 34
Predicted Label: 16

Input Text: new customer happened look app charge familiar could tell extra
charge
Actual Label: 34
Predicted Label: 15
```

The LSTM with baseline architecture and dropout has the best performance.

1.5 LSTM (with Word2Vec)

1.5.1 Set up the Word2Vec model

[ ]:
```
# Import and install the library and file for Word2Vec
import gensim

!pip install gdown # Install google download
!gdown https://drive.google.com/uc?id=1Av37IVBQAAntSe1X3MOAl5gvowQzd2_j #
↪Download the Word2Vec (GoogleNews-vectors-negative300.bin.gz)

# Define the Word2Vec model
word2vec_model = gensim.models.KeyedVectors.
↪load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary=True)
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages
(5.1.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from gdown) (3.14.0)
Requirement already satisfied: requests[socks] in
/usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from gdown) (4.66.4)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests[socks]->gdown) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.2.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7.1)
Downloading…
From (original):
https://drive.google.com/uc?id=1Av37IVBQAAntSe1X3MOAl5gvowQzd2_j
From (redirected): https://drive.google.com/uc?id=1Av37IVBQAAntSe1X3MOAl5gvowQzd
2_j&confirm=t&uuid=4335f280-bffc-48d7-a335-0876702f458b
To: /content/GoogleNews-vectors-negative300.bin.gz
100% 1.65G/1.65G [00:18<00:00, 87.9MB/s]
```

[ ]:
```
# Check the model's dimension
print(f'Word2Vec: {word2vec_model.vectors.shape}')
```

```
Word2Vec: (3000000, 300)
```

[ ]:
```
# Define the embedding matrix
embedding_matrix = np.zeros((voca_size+1, 300)) # For future dimension matching␣
↪with the word_index, add 1 to the vocabulary size, and match 300 from␣
↪Word2Vec
print(f'The shape of embedding matrix: {np.shape(embedding_matrix)}')
```

```
The shape of embedding matrix: (2089, 300)
```

[ ]:
```
# Match words in the word_index to those in the word2vec_model for creating␣
↪embedding matrix indices
def extract_vector(word):
    if word in word2vec_model:
        return word2vec_model[word]
    else:
        return None
```

[ ]:
```
# Match the index and word to creat an embedding matrix
for word, index in word_index.items():
    vector_value = extract_vector(word)
    if vector_value is not None:
        embedding_matrix[index] = vector_value
```

[ ]:
```
# Check the word 'card' vectors in the Word2Vec model
print(word2vec_model['card'])
```

```
[-1.63085938e-01  1.43554688e-01  1.97265625e-01  1.57226562e-01
  4.12597656e-02  2.43164062e-01 -2.21679688e-01 -5.02929688e-02
 -4.88281250e-02 -3.45703125e-01  4.18090820e-03 -8.10546875e-02
 -2.00195312e-01 -2.27539062e-01  4.41894531e-02 -1.96533203e-02
  2.83203125e-01  9.52148438e-02  8.74023438e-02 -2.69531250e-01
  2.27539062e-01  5.34667969e-02 -1.63574219e-02  3.97949219e-02
```

-2.97851562e-02	-1.83105469e-02	-8.64257812e-02	9.96093750e-02
3.33984375e-01	1.70898438e-02	-4.88281250e-02	-9.57031250e-02
2.73437500e-01	1.33789062e-01	-5.90820312e-02	6.93359375e-02
4.51660156e-02	1.31835938e-01	-1.74804688e-01	-2.12402344e-02
-1.53198242e-02	8.78906250e-03	3.78906250e-01	-1.36718750e-01
2.13867188e-01	-2.43164062e-01	2.70996094e-02	4.88281250e-02
-3.56445312e-02	-3.19824219e-02	-3.90625000e-01	2.40234375e-01
9.96093750e-02	-1.36718750e-01	-1.52343750e-01	3.14941406e-02
-1.62109375e-01	-1.44531250e-01	2.33398438e-01	-1.55273438e-01
1.92382812e-01	1.95312500e-01	-1.94335938e-01	1.54296875e-01
-1.16210938e-01	1.23535156e-01	-2.98828125e-01	-1.29882812e-01
-1.09375000e-01	1.28906250e-01	2.08007812e-01	1.16699219e-01
4.12109375e-01	-2.24609375e-01	1.53320312e-01	-6.73828125e-02
1.68945312e-01	1.11328125e-01	2.39257812e-01	2.81250000e-01
-1.22680664e-02	-1.40625000e-01	-9.17968750e-02	-3.32031250e-02
1.07421875e-01	-2.16064453e-02	-4.80957031e-02	-1.40625000e-01
-1.13281250e-01	1.34277344e-02	-1.46484375e-02	1.48925781e-02
-2.71484375e-01	-3.75000000e-01	-2.79296875e-01	-4.10156250e-02
-9.96093750e-02	3.28125000e-01	-1.78222656e-02	-2.13867188e-01
8.44726562e-02	-1.63085938e-01	2.31445312e-01	1.75781250e-01
-1.55273438e-01	-1.49414062e-01	3.55468750e-01	-4.62890625e-01
1.04980469e-01	-1.07910156e-01	7.32421875e-02	-1.77734375e-01
-5.98144531e-02	-6.98852539e-03	-1.87500000e-01	1.84570312e-01
1.28906250e-01	1.04003906e-01	5.03906250e-01	1.36718750e-01
-3.17382812e-02	8.54492188e-03	1.67236328e-02	2.28515625e-01
1.17797852e-02	-2.94921875e-01	2.04467773e-03	2.65625000e-01
-1.51367188e-01	-9.81445312e-02	1.82617188e-01	-2.08984375e-01
-2.83203125e-01	-3.71093750e-01	-1.21307373e-03	8.69140625e-02
-5.71289062e-02	1.93359375e-01	-2.79296875e-01	-3.71093750e-02
-3.98437500e-01	-2.40234375e-01	4.41894531e-02	3.49121094e-02
1.04980469e-01	-7.71484375e-02	1.12792969e-01	-6.29882812e-02
-1.68945312e-01	-7.56835938e-02	8.44726562e-02	2.63671875e-01
-4.17480469e-02	2.24609375e-02	1.41601562e-01	-7.81250000e-02
1.23046875e-01	2.51464844e-02	4.51660156e-03	5.12695312e-02
9.88769531e-03	-2.63671875e-01	3.45703125e-01	-8.34960938e-02
-1.04980469e-01	1.44531250e-01	1.71875000e-01	-3.96484375e-01
-3.35937500e-01	-1.33056641e-02	-3.68652344e-02	-3.28125000e-01
-1.54418945e-02	2.70996094e-02	-2.61718750e-01	1.87988281e-02
6.54296875e-02	-9.03320312e-02	2.41210938e-01	1.03515625e-01
-1.40625000e-01	-1.57226562e-01	-3.00781250e-01	3.22265625e-02
1.05957031e-01	-2.61718750e-01	-1.52343750e-01	4.73632812e-02
-4.37500000e-01	4.41406250e-01	-1.20605469e-01	-1.25000000e-01
-3.39843750e-01	-3.30078125e-01	2.89062500e-01	1.18164062e-01
-1.06933594e-01	1.22680664e-02	-2.65625000e-01	-2.02148438e-01
-1.29882812e-01	1.98242188e-01	-7.66601562e-02	1.31835938e-01
1.89453125e-01	-8.20312500e-02	-1.66992188e-01	1.67968750e-01
-3.88183594e-02	8.05664062e-02	-2.53906250e-02	2.73437500e-01
-9.86328125e-02	1.52343750e-01	1.07910156e-01	3.65234375e-01

```
 -3.75366211e-03  1.23023987e-04 -2.49023438e-01  5.88378906e-02
  7.27539062e-02  7.03125000e-02  8.49609375e-02 -1.63085938e-01
 -2.23632812e-01  8.88671875e-02 -1.00097656e-01  3.54003906e-02
 -1.42822266e-02 -2.35351562e-01 -3.06640625e-01  3.23486328e-03
  2.45117188e-01 -8.74023438e-02 -2.86865234e-02 -4.12597656e-02
  9.27734375e-02 -1.20849609e-02  1.43554688e-01 -6.39648438e-02
  2.17773438e-01 -5.98144531e-02 -6.17675781e-02 -3.37890625e-01
 -2.99072266e-03 -4.27246094e-02  2.67333984e-02  3.56445312e-02
 -6.31713867e-03  7.12890625e-02  1.03515625e-01 -6.20117188e-02
  1.52587891e-03 -4.21875000e-01 -8.48388672e-03  1.31835938e-01
  2.64892578e-02  1.58203125e-01  2.11181641e-02 -5.05371094e-02
  1.15234375e-01  4.46777344e-02 -1.75781250e-01 -3.06640625e-01
 -1.51367188e-01 -1.09375000e-01 -1.50390625e-01  7.91015625e-02
 -1.36718750e-01  5.00488281e-02 -2.23632812e-01 -8.98437500e-02
 -2.81250000e-01  2.13867188e-01  5.20019531e-02  3.32031250e-02
  1.87500000e-01 -2.50000000e-01 -1.50390625e-01  3.76953125e-01
  1.29882812e-01  1.48010254e-03 -9.91210938e-02  1.59179688e-01
 -1.65039062e-01 -1.15722656e-01  8.20312500e-02  8.93554688e-02
  1.38671875e-01  1.38549805e-02  1.08032227e-02  1.62109375e-01
 -9.86328125e-02 -5.02929688e-02  2.18505859e-02 -1.29882812e-01
 -4.68750000e-02 -1.04492188e-01 -1.25000000e-01  1.13281250e-01]
```

[ ]:
```python
# Check the index of the word 'card' in word_index
print(f"The index of card in word_index: {word_index['card']}")
```

```
The index of card in word_index: 1
```

[ ]:
```python
# Check the word 'card' vectors in the embedding matrix
print(embedding_matrix[1])
```

```
[-1.63085938e-01  1.43554688e-01  1.97265625e-01  1.57226562e-01
  4.12597656e-02  2.43164062e-01 -2.21679688e-01 -5.02929688e-02
 -4.88281250e-02 -3.45703125e-01  4.18090820e-03 -8.10546875e-02
 -2.00195312e-01 -2.27539062e-01  4.41894531e-02 -1.96533203e-02
  2.83203125e-01  9.52148438e-02  8.74023438e-02 -2.69531250e-01
  2.27539062e-01  5.34667969e-02 -1.63574219e-02  3.97949219e-02
 -2.97851562e-02 -1.83105469e-02 -8.64257812e-02  9.96093750e-02
  3.33984375e-01  1.70898438e-02 -4.88281250e-02 -9.57031250e-02
  2.73437500e-01  1.33789062e-01 -5.90820312e-02  6.93359375e-02
  4.51660156e-02  1.31835938e-01 -1.74804688e-01 -2.12402344e-02
 -1.53198242e-02  8.78906250e-03  3.78906250e-01 -1.36718750e-01
  2.13867188e-01 -2.43164062e-01  2.70996094e-02  4.88281250e-02
 -3.56445312e-02 -3.19824219e-02 -3.90625000e-01  2.40234375e-01
  9.96093750e-02 -1.36718750e-01 -1.52343750e-01  3.14941406e-02
 -1.62109375e-01 -1.44531250e-01  2.33398438e-01 -1.55273438e-01
  1.92382812e-01  1.95312500e-01 -1.94335938e-01  1.54296875e-01
 -1.16210938e-01  1.23535156e-01 -2.98828125e-01 -1.29882812e-01
 -1.09375000e-01  1.28906250e-01  2.08007812e-01  1.16699219e-01
```

4.12109375e-01	-2.24609375e-01	1.53320312e-01	-6.73828125e-02
1.68945312e-01	1.11328125e-01	2.39257812e-01	2.81250000e-01
-1.22680664e-02	-1.40625000e-01	-9.17968750e-02	-3.32031250e-02
1.07421875e-01	-2.16064453e-02	-4.80957031e-02	-1.40625000e-01
-1.13281250e-01	1.34277344e-02	-1.46484375e-02	1.48925781e-02
-2.71484375e-01	-3.75000000e-01	-2.79296875e-01	-4.10156250e-02
-9.96093750e-02	3.28125000e-01	-1.78222656e-02	-2.13867188e-01
8.44726562e-02	-1.63085938e-01	2.31445312e-01	1.75781250e-01
-1.55273438e-01	-1.49414062e-01	3.55468750e-01	-4.62890625e-01
1.04980469e-01	-1.07910156e-01	7.32421875e-02	-1.77734375e-01
-5.98144531e-02	-6.98852539e-03	-1.87500000e-01	1.84570312e-01
1.28906250e-01	1.04003906e-01	5.03906250e-01	1.36718750e-01
-3.17382812e-02	8.54492188e-03	1.67236328e-02	2.28515625e-01
1.17797852e-02	-2.94921875e-01	2.04467773e-03	2.65625000e-01
-1.51367188e-01	-9.81445312e-02	1.82617188e-01	-2.08984375e-01
-2.83203125e-01	-3.71093750e-01	-1.21307373e-03	8.69140625e-02
-5.71289062e-02	1.93359375e-01	-2.79296875e-01	-3.71093750e-02
-3.98437500e-01	-2.40234375e-01	4.41894531e-02	3.49121094e-02
1.04980469e-01	-7.71484375e-02	1.12792969e-01	-6.29882812e-02
-1.68945312e-01	-7.56835938e-02	8.44726562e-02	2.63671875e-01
-4.17480469e-02	2.24609375e-02	1.41601562e-01	-7.81250000e-02
1.23046875e-01	2.51464844e-02	4.51660156e-03	5.12695312e-02
9.88769531e-03	-2.63671875e-01	3.45703125e-01	-8.34960938e-02
-1.04980469e-01	1.44531250e-01	1.71875000e-01	-3.96484375e-01
-3.35937500e-01	-1.33056641e-02	-3.68652344e-02	-3.28125000e-01
-1.54418945e-02	2.70996094e-02	-2.61718750e-01	1.87988281e-02
6.54296875e-02	-9.03320312e-02	2.41210938e-01	1.03515625e-01
-1.40625000e-01	-1.57226562e-01	-3.00781250e-01	3.22265625e-02
1.05957031e-01	-2.61718750e-01	-1.52343750e-01	4.73632812e-02
-4.37500000e-01	4.41406250e-01	-1.20605469e-01	-1.25000000e-01
-3.39843750e-01	-3.30078125e-01	2.89062500e-01	1.18164062e-01
-1.06933594e-01	1.22680664e-02	-2.65625000e-01	-2.02148438e-01
-1.29882812e-01	1.98242188e-01	-7.66601562e-02	1.31835938e-01
1.89453125e-01	-8.20312500e-02	-1.66992188e-01	1.67968750e-01
-3.88183594e-02	8.05664062e-02	-2.53906250e-02	2.73437500e-01
-9.86328125e-02	1.52343750e-01	1.07910156e-01	3.65234375e-01
-3.75366211e-03	1.23023987e-04	-2.49023438e-01	5.88378906e-02
7.27539062e-02	7.03125000e-02	8.49609375e-02	-1.63085938e-01
-2.23632812e-01	8.88671875e-02	-1.00097656e-01	3.54003906e-02
-1.42822266e-02	-2.35351562e-01	-3.06640625e-01	3.23486328e-03
2.45117188e-01	-8.74023438e-02	-2.86865234e-02	-4.12597656e-02
9.27734375e-02	-1.20849609e-02	1.43554688e-01	-6.39648438e-02
2.17773438e-01	-5.98144531e-02	-6.17675781e-02	-3.37890625e-01
-2.99072266e-03	-4.27246094e-02	2.67333984e-02	3.56445312e-02
-6.31713867e-03	7.12890625e-02	1.03515625e-01	-6.20117188e-02
1.52587891e-03	-4.21875000e-01	-8.48388672e-03	1.31835938e-01
2.64892578e-02	1.58203125e-01	2.11181641e-02	-5.05371094e-02
1.15234375e-01	4.46777344e-02	-1.75781250e-01	-3.06640625e-01

```
 -1.51367188e-01 -1.09375000e-01 -1.50390625e-01  7.91015625e-02
 -1.36718750e-01  5.00488281e-02 -2.23632812e-01 -8.98437500e-02
 -2.81250000e-01  2.13867188e-01  5.20019531e-02  3.32031250e-02
  1.87500000e-01 -2.50000000e-01 -1.50390625e-01  3.76953125e-01
  1.29882812e-01  1.48010254e-03 -9.91210938e-02  1.59179688e-01
 -1.65039062e-01 -1.15722656e-01  8.20312500e-02  8.93554688e-02
  1.38671875e-01  1.38549805e-02  1.08032227e-02  1.62109375e-01
 -9.86328125e-02 -5.02929688e-02  2.18505859e-02 -1.29882812e-01
 -4.68750000e-02 -1.04492188e-01 -1.25000000e-01  1.13281250e-01]
```

[ ]:
```
# Check the word 'topup' vectors in the Word2Vec model
print(word2vec_model['topup'])
```

```
[ 5.17578125e-02 -1.61132812e-01 -1.33789062e-01  1.93359375e-01
 -1.39770508e-02  1.69921875e-01  1.06933594e-01 -2.08007812e-01
  7.81250000e-02  1.11816406e-01  3.82995605e-03  2.55126953e-02
 -2.75390625e-01 -9.71679688e-02 -2.84423828e-02  1.31835938e-01
  2.27539062e-01  2.29492188e-02  1.18652344e-01  6.03027344e-02
 -1.92871094e-02 -6.15234375e-02  1.73828125e-01  1.72851562e-01
  2.50244141e-02 -1.54296875e-01 -1.62109375e-01  1.02539062e-01
  1.50146484e-02 -7.42187500e-02 -2.60009766e-02  4.98046875e-02
 -3.32031250e-02 -1.01074219e-01 -5.34667969e-02 -2.45117188e-01
 -1.04003906e-01 -1.25976562e-01 -7.86132812e-02  6.44531250e-02
 -2.86865234e-02 -3.44238281e-02  3.75976562e-02 -1.87500000e-01
 -4.78515625e-02 -3.53515625e-01 -1.01318359e-02  3.97949219e-02
 -1.40625000e-01 -1.69921875e-01  3.11279297e-02 -5.27343750e-02
  1.09863281e-01 -7.35473633e-03 -4.27246094e-02  3.00292969e-02
 -3.08593750e-01  1.87988281e-02  2.20947266e-02 -1.17675781e-01
 -1.70898438e-01 -2.04101562e-01 -7.51953125e-02  1.94091797e-02
 -1.42578125e-01 -1.42578125e-01 -2.14843750e-01  7.71484375e-02
 -3.09753418e-03  1.10839844e-01 -1.66015625e-02 -2.01416016e-02
  1.69921875e-01 -9.91210938e-02 -4.76074219e-02 -2.30468750e-01
  1.49414062e-01  1.42578125e-01 -2.25830078e-02  2.06298828e-02
 -7.95898438e-02  1.29882812e-01  3.51562500e-02  2.03125000e-01
 -3.39355469e-02  7.12890625e-02 -4.41894531e-02  3.80859375e-02
 -3.80859375e-02  2.55859375e-01 -8.74023438e-02 -2.44140625e-02
 -1.21093750e-01 -1.15722656e-01 -1.11694336e-02  1.20117188e-01
 -7.03125000e-02 -7.12890625e-02  3.30078125e-01  4.63867188e-02
 -5.00488281e-02 -7.03125000e-02 -7.91015625e-02  1.96289062e-01
 -1.24511719e-01 -7.03125000e-02  1.53320312e-01  5.02929688e-02
  2.69531250e-01  1.39160156e-02 -7.22656250e-02 -1.26953125e-01
 -2.19726562e-01  3.78417969e-02 -3.80859375e-02  1.30859375e-01
 -1.10351562e-01 -1.52343750e-01  1.64062500e-01 -7.91015625e-02
  2.05078125e-01 -1.33789062e-01 -3.05175781e-02  1.25976562e-01
  9.52148438e-02  9.57031250e-02 -2.29492188e-01  1.22558594e-01
  1.14257812e-01 -4.95605469e-02 -7.42187500e-02  1.01074219e-01
 -1.31835938e-01 -3.12500000e-01 -1.01562500e-01 -5.81054688e-02
  1.30859375e-01  2.13623047e-02 -1.19140625e-01  2.65625000e-01
```

```
 5.34667969e-02 -1.57226562e-01 -1.21593475e-04  1.58203125e-01
 9.27734375e-02 -4.36401367e-03 -9.91210938e-02 -6.34765625e-02
-1.09375000e-01 -2.30712891e-02  1.40625000e-01 -1.58203125e-01
-1.31835938e-01 -2.61718750e-01  1.78710938e-01 -1.61132812e-01
-1.74560547e-02  9.66796875e-02 -2.59765625e-01  5.73730469e-03
-1.40625000e-01 -5.61523438e-02 -1.52343750e-01 -2.17285156e-02
-1.05957031e-01 -3.75976562e-02  1.26953125e-01 -2.38037109e-03
-2.61718750e-01 -1.48437500e-01 -1.57226562e-01 -9.13085938e-02
 7.47070312e-02 -1.49414062e-01 -9.57031250e-02  7.51953125e-02
 2.38281250e-01 -1.79687500e-01 -1.28906250e-01  5.63964844e-02
-1.40625000e-01 -1.51367188e-01  2.74658203e-03  9.22851562e-02
 8.64257812e-02 -1.16210938e-01 -5.54199219e-02  1.99218750e-01
-3.10546875e-01  7.85827637e-04 -2.13867188e-01 -2.08007812e-01
-1.66015625e-01 -1.18164062e-01 -1.77001953e-02  5.59082031e-02
 1.17187500e-02 -3.61328125e-02 -8.34960938e-02 -9.37500000e-02
 1.06445312e-01  1.48437500e-01 -1.23046875e-01  1.11816406e-01
 3.58886719e-02  9.57031250e-02 -3.11279297e-02 -1.01562500e-01
-6.73828125e-02  4.30297852e-03  1.97265625e-01  1.59179688e-01
 3.61328125e-02  1.09375000e-01 -1.45507812e-01  1.11328125e-01
 2.24609375e-02 -1.13769531e-01 -1.79687500e-01  2.50244141e-02
-5.88378906e-02  2.44140625e-02  2.57568359e-02 -9.33837891e-03
-6.78710938e-02  1.73828125e-01 -1.55273438e-01  2.70996094e-02
-2.50244141e-02 -9.37500000e-02 -3.36914062e-02 -7.27539062e-02
 7.95898438e-02 -4.15039062e-02  5.71289062e-02  6.93359375e-02
-2.06298828e-02  1.57226562e-01 -3.39355469e-02  1.02539062e-01
 7.86132812e-02  2.85644531e-02  5.59082031e-02  3.44238281e-02
 6.98242188e-02 -1.04370117e-02  2.89306641e-02  1.53320312e-01
 1.12304688e-01 -9.22851562e-02 -3.85742188e-02  1.22558594e-01
-1.28906250e-01  1.54296875e-01 -3.58581543e-03  1.07421875e-01
 3.24707031e-02 -3.32031250e-02 -1.87988281e-02  3.41796875e-02
 2.07031250e-01 -1.81640625e-01 -1.67968750e-01  2.14843750e-02
-1.84570312e-01 -9.71679688e-02 -1.19140625e-01 -5.59082031e-02
-3.27148438e-02 -4.76074219e-03 -6.88476562e-02 -4.71191406e-02
-1.16699219e-01  1.67968750e-01  1.00097656e-01 -1.84326172e-02
-1.53198242e-02 -7.42187500e-02 -2.50244141e-02  6.83593750e-02
-1.05957031e-01  2.25585938e-01  9.91821289e-04 -4.07714844e-02
 1.09863281e-01  1.57165527e-03  2.80761719e-02  2.04101562e-01
 1.78710938e-01  2.55859375e-01  4.95605469e-02  2.03857422e-02
 6.44531250e-02 -7.61718750e-02 -9.76562500e-02  2.46582031e-02
-8.88671875e-02 -1.98974609e-02 -1.02539062e-01  1.50299072e-03]
```

[ ]:
```
# Check the index of the word 'topup' in word_index
print(f"The index of topup in word_index: {word_index['topup']}")
```

```
The index of topup in word_index: 19
```

[ ]:
```
# Check the word 'topup' vectors in the embedding matrix
print(embedding_matrix[20])
```

```
[ 0.04980469  0.06640625  0.03833008  0.02355957 -0.02148438  0.20898438
  0.06396484 -0.02282715 -0.04101562 -0.26757812  0.1015625   0.10693359
 -0.06591797  0.16992188 -0.0078125  -0.07861328 -0.06591797  0.12060547
 -0.00390625 -0.02770996  0.19238281  0.13183594  0.16113281 -0.07324219
 -0.22167969 -0.05102539 -0.12255859  0.06298828  0.01080322 -0.12695312
  0.04614258 -0.01794434 -0.03222656 -0.21484375 -0.01696777  0.0098877
 -0.00976562 -0.05175781  0.12011719  0.04980469  0.01867676  0.05712891
 -0.04492188 -0.16113281 -0.08105469 -0.09960938 -0.19824219 -0.00109863
  0.01239014  0.23144531 -0.06738281  0.08105469 -0.0177002  -0.12402344
 -0.14746094 -0.10253906 -0.23046875 -0.03149414  0.03125    -0.09033203
  0.08251953 -0.09326172 -0.21679688  0.06103516  0.046875   -0.03466797
 -0.05908203  0.12695312 -0.0025177   0.08251953  0.0703125  -0.01037598
  0.08447266  0.006073   -0.12988281 -0.06689453  0.18359375  0.21191406
  0.01495361 -0.04907227 -0.01525879  0.07080078 -0.04418945 -0.01153564
 -0.05273438  0.06982422 -0.14453125 -0.06103516 -0.12402344  0.10058594
  0.17675781  0.12353516 -0.03710938 -0.38085938  0.15136719 -0.22753906
  0.03149414  0.09228516  0.01525879  0.07666016 -0.078125   -0.07080078
  0.05688477  0.10058594 -0.11425781 -0.02893066 -0.19921875 -0.08642578
  0.08349609  0.00552368 -0.19433594  0.05249023 -0.08251953 -0.12597656
  0.12792969  0.09814453 -0.12255859  0.10644531  0.0703125   0.13378906
  0.01550293  0.16894531 -0.10644531 -0.078125   -0.09716797 -0.02685547
 -0.07470703 -0.19628906  0.23632812  0.07470703  0.04711914 -0.00427246
  0.0133667  -0.04882812 -0.02416992 -0.03588867  0.14550781 -0.02282715
  0.10546875  0.08349609 -0.00830078 -0.04931641 -0.09521484  0.11914062
  0.07470703 -0.04223633 -0.10742188 -0.22167969  0.05541992 -0.08398438
  0.00098419  0.09667969 -0.13867188  0.13476562  0.17871094  0.03149414
 -0.06494141 -0.03613281 -0.27148438 -0.08691406  0.20800781  0.10546875
 -0.06079102 -0.09863281  0.06884766 -0.19433594  0.03857422 -0.04589844
  0.1484375  -0.140625    0.00564575  0.03564453 -0.06542969 -0.13476562
  0.12060547 -0.01446533  0.2109375   0.21679688 -0.20507812  0.15332031
 -0.28515625 -0.13769531  0.19140625 -0.05126953 -0.23242188 -0.08691406
 -0.14257812 -0.06982422  0.15820312 -0.23535156 -0.06982422  0.01220703
  0.28515625 -0.09667969 -0.08837891  0.0625      0.15917969 -0.11767578
 -0.18164062 -0.35546875 -0.05102539  0.12255859 -0.125      -0.01550293
 -0.09521484  0.00643921 -0.13769531 -0.11865234 -0.02819824  0.02758789
 -0.1171875  -0.0324707  -0.0098877   0.12255859  0.125      -0.01220703
  0.21972656  0.14160156 -0.09033203 -0.05566406 -0.03759766  0.04785156
 -0.04980469 -0.15625    -0.09912109 -0.08691406 -0.10693359 -0.01293945
  0.00540161  0.04882812  0.01586914  0.10058594  0.03588867 -0.10351562
  0.03540039 -0.14453125 -0.13671875 -0.15820312  0.00994873  0.00726318
 -0.10791016  0.04394531 -0.00314331  0.04077148 -0.03466797 -0.03271484
  0.140625    0.04492188  0.12988281 -0.09326172  0.13183594  0.04248047
  0.06396484  0.30078125  0.0559082   0.0039978   0.1328125   0.0390625
 -0.1796875  -0.05053711 -0.16992188  0.04858398 -0.14941406  0.17675781
```

```
 0.09130859  0.08496094  0.06738281  0.03393555 -0.08300781 -0.06982422
 0.06225586  0.12353516  0.05004883  0.06738281  0.05029297 -0.0324707
 0.04003906 -0.20703125 -0.07861328 -0.00389099 -0.08398438 -0.11621094
 0.05078125  0.24023438 -0.13183594  0.02600098 -0.23339844 -0.11474609
 0.07568359 -0.12695312 -0.06787109  0.08496094 -0.00476074 -0.10595703
 0.00970459  0.14648438 -0.04711914  0.20703125  0.01196289  0.03564453]
```

1.5.2 LSTM (baseline)

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
hidden_unit = 30
nlabel = 77

model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 300, weights=[embedding_matrix],␣
 ↪input_length=max_length_train_text, trainable=False)
model.add(e)
model.add(layers.LSTM(hidden_unit))
model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Summary the model
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 29, 300)           626700

 lstm_6 (LSTM)               (None, 30)                39720

 dense_6 (Dense)             (None, 77)                2387

=================================================================
Total params: 668807 (2.55 MB)
Trainable params: 42107 (164.48 KB)
Non-trainable params: 626700 (2.39 MB)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
```

```
model_checkpoint_path = folder_path + 'LSTM_word2vec_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Define early stopping
es =  tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3) # Random␣
 ↪number of patience
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

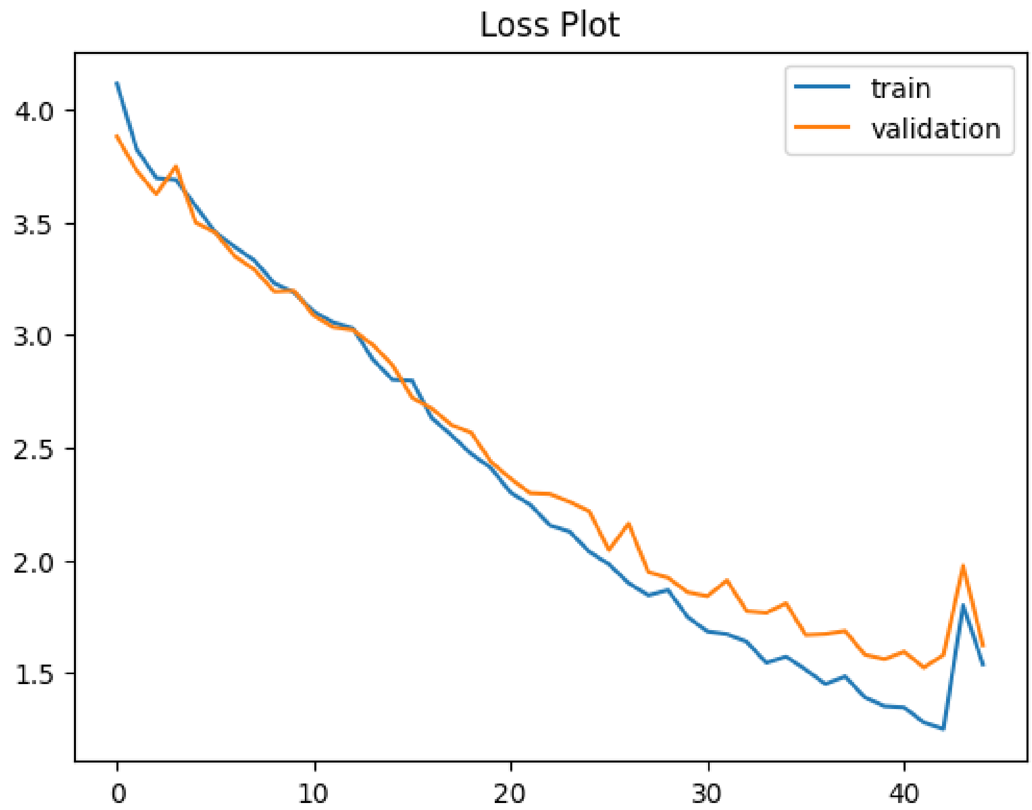
```
Epoch 1/100
230/230 [==============================] - 17s 60ms/step - loss: 4.1156 -
accuracy: 0.0242 - val_loss: 3.8803 - val_accuracy: 0.0397
Epoch 2/100
230/230 [==============================] - 10s 43ms/step - loss: 3.8250 -
accuracy: 0.0403 - val_loss: 3.7300 - val_accuracy: 0.0332
Epoch 3/100
230/230 [==============================] - 12s 54ms/step - loss: 3.6964 -
accuracy: 0.0448 - val_loss: 3.6242 - val_accuracy: 0.0577
Epoch 4/100
230/230 [==============================] - 11s 47ms/step - loss: 3.6885 -
accuracy: 0.0483 - val_loss: 3.7484 - val_accuracy: 0.0457
Epoch 5/100
```

```
230/230 [==============================] - 8s 36ms/step - loss: 3.5730 -
accuracy: 0.0559 - val_loss: 3.4992 - val_accuracy: 0.0626
Epoch 6/100
230/230 [==============================] - 10s 43ms/step - loss: 3.4550 -
accuracy: 0.0595 - val_loss: 3.4535 - val_accuracy: 0.0735
Epoch 7/100
230/230 [==============================] - 8s 37ms/step - loss: 3.3899 -
accuracy: 0.0670 - val_loss: 3.3481 - val_accuracy: 0.0751
Epoch 8/100
230/230 [==============================] - 8s 36ms/step - loss: 3.3292 -
accuracy: 0.0788 - val_loss: 3.2886 - val_accuracy: 0.0947
Epoch 9/100
230/230 [==============================] - 7s 30ms/step - loss: 3.2287 -
accuracy: 0.0896 - val_loss: 3.1921 - val_accuracy: 0.0974
Epoch 10/100
230/230 [==============================] - 7s 30ms/step - loss: 3.1885 -
accuracy: 0.0961 - val_loss: 3.1957 - val_accuracy: 0.1051
Epoch 11/100
230/230 [==============================] - 5s 20ms/step - loss: 3.1033 -
accuracy: 0.1089 - val_loss: 3.0856 - val_accuracy: 0.1083
Epoch 12/100
230/230 [==============================] - 12s 51ms/step - loss: 3.0548 -
accuracy: 0.1075 - val_loss: 3.0339 - val_accuracy: 0.1165
Epoch 13/100
230/230 [==============================] - 5s 21ms/step - loss: 3.0292 -
accuracy: 0.1201 - val_loss: 3.0214 - val_accuracy: 0.1394
Epoch 14/100
230/230 [==============================] - 5s 22ms/step - loss: 2.8912 -
accuracy: 0.1465 - val_loss: 2.9558 - val_accuracy: 0.1486
Epoch 15/100
230/230 [==============================] - 7s 31ms/step - loss: 2.8010 -
accuracy: 0.1663 - val_loss: 2.8664 - val_accuracy: 0.1584
Epoch 16/100
230/230 [==============================] - 5s 22ms/step - loss: 2.7962 -
accuracy: 0.1687 - val_loss: 2.7211 - val_accuracy: 0.1987
Epoch 17/100
230/230 [==============================] - 8s 33ms/step - loss: 2.6303 -
accuracy: 0.1983 - val_loss: 2.6730 - val_accuracy: 0.2003
Epoch 18/100
230/230 [==============================] - 5s 23ms/step - loss: 2.5544 -
accuracy: 0.2111 - val_loss: 2.5998 - val_accuracy: 0.2123
Epoch 19/100
230/230 [==============================] - 5s 23ms/step - loss: 2.4715 -
accuracy: 0.2275 - val_loss: 2.5657 - val_accuracy: 0.2308
Epoch 20/100
230/230 [==============================] - 7s 30ms/step - loss: 2.4096 -
accuracy: 0.2365 - val_loss: 2.4372 - val_accuracy: 0.2580
Epoch 21/100
```

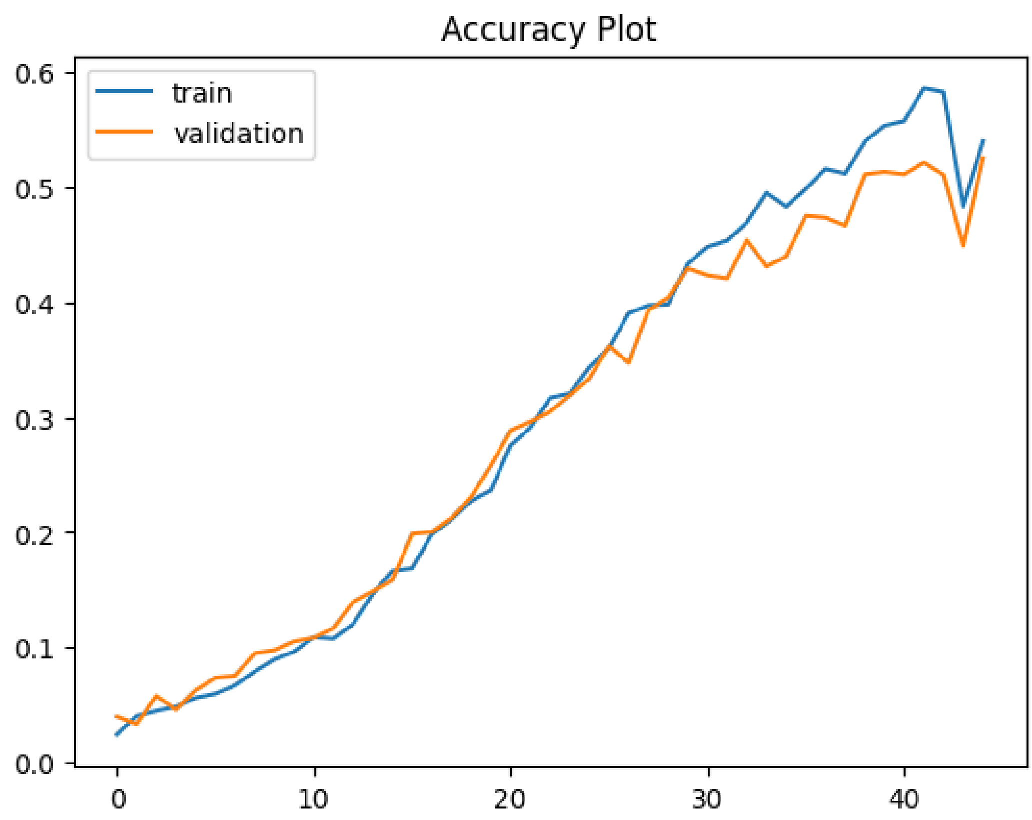
```
230/230 [==============================] - 5s 23ms/step - loss: 2.3002 -
accuracy: 0.2755 - val_loss: 2.3618 - val_accuracy: 0.2885
Epoch 22/100
230/230 [==============================] - 6s 27ms/step - loss: 2.2456 -
accuracy: 0.2908 - val_loss: 2.2972 - val_accuracy: 0.2961
Epoch 23/100
230/230 [==============================] - 6s 26ms/step - loss: 2.1545 -
accuracy: 0.3168 - val_loss: 2.2933 - val_accuracy: 0.3048
Epoch 24/100
230/230 [==============================] - 5s 22ms/step - loss: 2.1261 -
accuracy: 0.3207 - val_loss: 2.2585 - val_accuracy: 0.3190
Epoch 25/100
230/230 [==============================] - 7s 31ms/step - loss: 2.0376 -
accuracy: 0.3435 - val_loss: 2.2165 - val_accuracy: 0.3337
Epoch 26/100
230/230 [==============================] - 5s 23ms/step - loss: 1.9811 -
accuracy: 0.3601 - val_loss: 2.0442 - val_accuracy: 0.3620
Epoch 27/100
230/230 [==============================] - 5s 23ms/step - loss: 1.8978 -
accuracy: 0.3906 - val_loss: 2.1619 - val_accuracy: 0.3473
Epoch 28/100
230/230 [==============================] - 7s 29ms/step - loss: 1.8436 -
accuracy: 0.3972 - val_loss: 1.9467 - val_accuracy: 0.3930
Epoch 29/100
230/230 [==============================] - 5s 22ms/step - loss: 1.8666 -
accuracy: 0.3982 - val_loss: 1.9213 - val_accuracy: 0.4039
Epoch 30/100
230/230 [==============================] - 6s 25ms/step - loss: 1.7474 -
accuracy: 0.4336 - val_loss: 1.8573 - val_accuracy: 0.4295
Epoch 31/100
230/230 [==============================] - 6s 26ms/step - loss: 1.6830 -
accuracy: 0.4481 - val_loss: 1.8397 - val_accuracy: 0.4235
Epoch 32/100
230/230 [==============================] - 5s 21ms/step - loss: 1.6706 -
accuracy: 0.4536 - val_loss: 1.9097 - val_accuracy: 0.4208
Epoch 33/100
230/230 [==============================] - 7s 30ms/step - loss: 1.6372 -
accuracy: 0.4692 - val_loss: 1.7732 - val_accuracy: 0.4540
Epoch 34/100
230/230 [==============================] - 5s 22ms/step - loss: 1.5439 -
accuracy: 0.4954 - val_loss: 1.7658 - val_accuracy: 0.4311
Epoch 35/100
230/230 [==============================] - 7s 30ms/step - loss: 1.5710 -
accuracy: 0.4833 - val_loss: 1.8079 - val_accuracy: 0.4398
Epoch 36/100
230/230 [==============================] - 7s 32ms/step - loss: 1.5137 -
accuracy: 0.4986 - val_loss: 1.6672 - val_accuracy: 0.4752
Epoch 37/100
```

```
230/230 [==============================] - 5s 21ms/step - loss: 1.4493 -
accuracy: 0.5159 - val_loss: 1.6713 - val_accuracy: 0.4736
Epoch 38/100
230/230 [==============================] - 7s 29ms/step - loss: 1.4833 -
accuracy: 0.5117 - val_loss: 1.6851 - val_accuracy: 0.4665
Epoch 39/100
230/230 [==============================] - 5s 23ms/step - loss: 1.3904 -
accuracy: 0.5399 - val_loss: 1.5800 - val_accuracy: 0.5112
Epoch 40/100
230/230 [==============================] - 5s 21ms/step - loss: 1.3515 -
accuracy: 0.5535 - val_loss: 1.5596 - val_accuracy: 0.5133
Epoch 41/100
230/230 [==============================] - 9s 38ms/step - loss: 1.3457 -
accuracy: 0.5576 - val_loss: 1.5929 - val_accuracy: 0.5112
Epoch 42/100
230/230 [==============================] - 5s 21ms/step - loss: 1.2789 -
accuracy: 0.5862 - val_loss: 1.5229 - val_accuracy: 0.5215
Epoch 43/100
230/230 [==============================] - 7s 31ms/step - loss: 1.2500 -
accuracy: 0.5828 - val_loss: 1.5785 - val_accuracy: 0.5106
Epoch 44/100
230/230 [==============================] - 8s 33ms/step - loss: 1.7998 -
accuracy: 0.4834 - val_loss: 1.9750 - val_accuracy: 0.4491
Epoch 45/100
230/230 [==============================] - 9s 41ms/step - loss: 1.5368 -
accuracy: 0.5402 - val_loss: 1.6225 - val_accuracy: 0.5253
Training time: 317.1865930557251 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 4s 16ms/step - loss: 1.6655 - accuracy:
0.5049
Test Loss: 1.6655056476593018
Test Accuracy: 50.49
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 3s 16ms/step
Precision: 49.75
Recall: 50.49
F1 Score: 46.57
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[ ]:
```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support
```

0	0.60	0.15	0.24	40
1	0.29	0.57	0.39	40
2	0.97	0.95	0.96	40
3	0.00	0.00	0.00	40
4	0.83	0.85	0.84	40
5	0.37	0.62	0.46	40
6	0.65	0.78	0.70	40
7	0.50	0.72	0.59	40
8	0.55	0.82	0.66	40
9	0.86	0.62	0.72	40
10	0.44	0.30	0.36	40
11	0.43	0.82	0.56	40
12	0.65	0.28	0.39	40
13	0.36	0.70	0.47	40
14	0.46	0.78	0.57	40
15	0.58	0.85	0.69	40
16	0.49	0.50	0.49	40
17	0.88	0.90	0.89	40
18	0.71	0.75	0.73	40
19	0.82	0.90	0.86	40
20	0.50	0.65	0.57	40
21	0.00	0.00	0.00	40
22	0.44	0.75	0.56	40
23	0.00	0.00	0.00	40
24	0.92	0.82	0.87	40
25	0.65	0.78	0.70	40
26	0.36	0.25	0.29	40
27	0.25	0.03	0.05	40
28	0.78	0.62	0.69	40
29	0.24	0.80	0.37	40
30	0.00	0.00	0.00	40
31	0.87	0.85	0.86	40
32	0.88	0.90	0.89	40
33	0.64	0.85	0.73	40
34	0.79	0.78	0.78	40
35	0.47	0.62	0.54	40
36	0.60	0.65	0.63	40
37	0.00	0.00	0.00	40
38	0.00	0.00	0.00	40
39	0.46	0.68	0.55	40
40	0.20	0.03	0.04	40
41	0.43	0.45	0.44	40
42	0.26	0.17	0.21	40
43	0.60	0.23	0.33	40
44	0.00	0.00	0.00	40
45	0.58	0.70	0.64	40
46	0.67	0.40	0.50	40
47	0.45	0.82	0.58	40

```
          48       0.76      0.40      0.52        40
          49       0.50      0.03      0.05        40
          50       0.67      0.50      0.57        40
          51       0.54      0.93      0.69        40
          52       0.21      0.07      0.11        40
          53       0.63      0.55      0.59        40
          54       0.46      0.40      0.43        40
          55       0.50      0.03      0.05        40
          56       0.60      0.07      0.13        40
          57       0.92      0.85      0.88        40
          58       0.47      0.40      0.43        40
          59       0.49      0.42      0.45        40
          60       0.72      0.85      0.78        40
          61       0.38      0.15      0.21        40
          62       0.67      0.15      0.24        40
          63       0.57      0.62      0.60        40
          64       0.55      0.75      0.63        40
          65       0.81      0.55      0.66        40
          66       0.47      0.53      0.49        40
          67       0.74      0.42      0.54        40
          68       0.00      0.00      0.00        40
          69       0.00      0.00      0.00        40
          70       0.73      1.00      0.84        40
          71       0.00      0.00      0.00        40
          72       0.00      0.00      0.00        40
          73       1.00      0.82      0.90        40
          74       0.10      0.97      0.18        40
          75       0.44      0.82      0.57        40
          76       0.92      0.88      0.90        40

    accuracy                           0.50      3080
   macro avg       0.50      0.50      0.47      3080
weighted avg       0.50      0.50      0.47      3080

The number of misclassifications: 1525
Proportion of misclassifications: 49.51%
Input Text: locate card
Actual Label: 11
Predicted Label: 41

Input Text: get card
Actual Label: 11
Predicted Label: 43

Input Text: received card
Actual Label: 11
Predicted Label: 43
```

```
Input Text: tracking number card mailed
Actual Label: 11
Predicted Label: 22

Input Text: ordered card still havent received two weeks
Actual Label: 11
Predicted Label: 9

Input Text: im starting think card lost still hasnt arrived help
Actual Label: 11
Predicted Label: 41

Input Text: tracking info available
Actual Label: 11
Predicted Label: 42

Input Text: received new card dont see app anywhere
Actual Label: 13
Predicted Label: 12

Input Text: add card account
Actual Label: 13
Predicted Label: 10

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 11

Input Text: hello found card misplaced need reactive
Actual Label: 13
Predicted Label: 41

Input Text: found card add app
Actual Label: 13
Predicted Label: 40

Input Text: link credit card
Actual Label: 13
Predicted Label: 39

Input Text: reactivate lost card found morning jacket pocket
Actual Label: 13
Predicted Label: 12

Input Text: app doesnt show card received
Actual Label: 13
Predicted Label: 11
```

```
Input Text: please show find location link card
Actual Label: 13
Predicted Label: 41

Input Text: way make old card usable app
Actual Label: 13
Predicted Label: 29

Input Text: need go app enter card info
Actual Label: 13
Predicted Label: 42

Input Text: found lost stolen card way link card account app
Actual Label: 13
Predicted Label: 41

Input Text: good time exchange
Actual Label: 32
Predicted Label: 33

Input Text: exchange rate like app
Actual Label: 32
Predicted Label: 17

Input Text: currencies exchange rate calculated
Actual Label: 32
Predicted Label: 31

Input Text: kind foreign exchange rate get exchange money
Actual Label: 32
Predicted Label: 31

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: charged
Actual Label: 17
Predicted Label: 63

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 34

Input Text: wrong exchange rate used bought something foriegn currency
Actual Label: 17
Predicted Label: 76
```

```
Input Text: would like refund extra pound charged
Actual Label: 34
Predicted Label: 19

Input Text: explain random charge
Actual Label: 34
Predicted Label: 63

Input Text: remember purchasing anything £ statement please tell
Actual Label: 34
Predicted Label: 45

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

1.5.3 LSTM (with dropout)

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
hidden_unit = 30
nlabel = 77

dropout_model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 300, weights=[embedding_matrix],␣
↪input_length=max_length_train_text, trainable=False)
dropout_model.add(e)
dropout_model.add(layers.LSTM(hidden_unit, dropout=0.2))
dropout_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
dropout_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy'])

# Summary the model
dropout_model.summary()
```

```
Model: "sequential_7"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 29, 300)           626700

 lstm_7 (LSTM)               (None, 30)                39720

 dense_7 (Dense)             (None, 77)                2387

=================================================================
Total params: 668807 (2.55 MB)
Trainable params: 42107 (164.48 KB)
Non-trainable params: 626700 (2.39 MB)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'dropout_LSTM_word2vec_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = dropout_model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

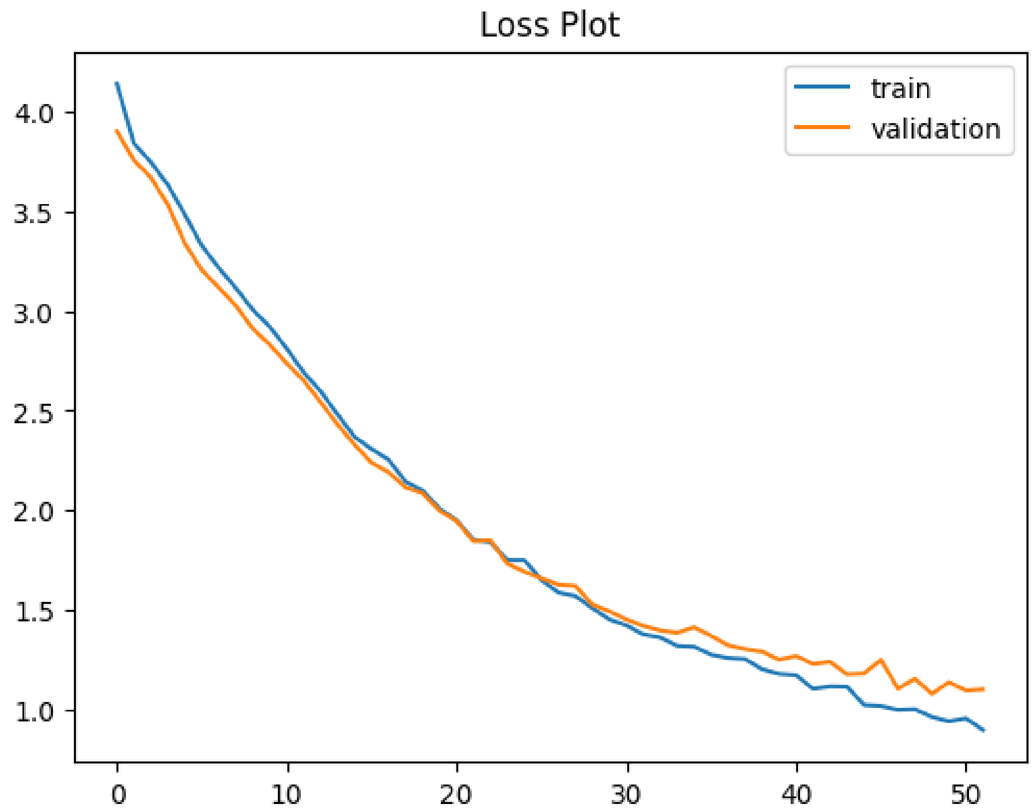
```
Epoch 1/100
230/230 [==============================] - 10s 32ms/step - loss: 4.1376 -
accuracy: 0.0193 - val_loss: 3.9000 - val_accuracy: 0.0321
Epoch 2/100
230/230 [==============================] - 6s 25ms/step - loss: 3.8377 -
accuracy: 0.0366 - val_loss: 3.7534 - val_accuracy: 0.0359
Epoch 3/100
230/230 [==============================] - 10s 45ms/step - loss: 3.7450 -
accuracy: 0.0395 - val_loss: 3.6682 - val_accuracy: 0.0555
Epoch 4/100
230/230 [==============================] - 12s 50ms/step - loss: 3.6308 -
accuracy: 0.0502 - val_loss: 3.5310 - val_accuracy: 0.0670
Epoch 5/100
230/230 [==============================] - 13s 55ms/step - loss: 3.4824 -
accuracy: 0.0682 - val_loss: 3.3377 - val_accuracy: 0.0773
Epoch 6/100
230/230 [==============================] - 20s 87ms/step - loss: 3.3260 -
accuracy: 0.0791 - val_loss: 3.2037 - val_accuracy: 0.0936
Epoch 7/100
230/230 [==============================] - 14s 63ms/step - loss: 3.2143 -
accuracy: 0.0913 - val_loss: 3.1157 - val_accuracy: 0.0931
Epoch 8/100
230/230 [==============================] - 11s 47ms/step - loss: 3.1142 -
accuracy: 0.0976 - val_loss: 3.0257 - val_accuracy: 0.1181
Epoch 9/100
230/230 [==============================] - 10s 45ms/step - loss: 3.0052 -
```

```
accuracy: 0.1194 - val_loss: 2.9100 - val_accuracy: 0.1399
Epoch 10/100
230/230 [==============================] - 11s 49ms/step - loss: 2.9193 -
accuracy: 0.1371 - val_loss: 2.8299 - val_accuracy: 0.1492
Epoch 11/100
230/230 [==============================] - 11s 49ms/step - loss: 2.8110 -
accuracy: 0.1563 - val_loss: 2.7365 - val_accuracy: 0.1932
Epoch 12/100
230/230 [==============================] - 10s 42ms/step - loss: 2.6905 -
accuracy: 0.2002 - val_loss: 2.6498 - val_accuracy: 0.2150
Epoch 13/100
230/230 [==============================] - 13s 55ms/step - loss: 2.5966 -
accuracy: 0.2156 - val_loss: 2.5415 - val_accuracy: 0.2270
Epoch 14/100
230/230 [==============================] - 11s 49ms/step - loss: 2.4795 -
accuracy: 0.2476 - val_loss: 2.4292 - val_accuracy: 0.2657
Epoch 15/100
230/230 [==============================] - 9s 38ms/step - loss: 2.3678 -
accuracy: 0.2746 - val_loss: 2.3284 - val_accuracy: 0.2831
Epoch 16/100
230/230 [==============================] - 8s 36ms/step - loss: 2.3066 -
accuracy: 0.2905 - val_loss: 2.2382 - val_accuracy: 0.3239
Epoch 17/100
230/230 [==============================] - 6s 26ms/step - loss: 2.2535 -
accuracy: 0.3101 - val_loss: 2.1902 - val_accuracy: 0.3408
Epoch 18/100
230/230 [==============================] - 7s 29ms/step - loss: 2.1428 -
accuracy: 0.3371 - val_loss: 2.1148 - val_accuracy: 0.3353
Epoch 19/100
230/230 [==============================] - 7s 30ms/step - loss: 2.1001 -
accuracy: 0.3518 - val_loss: 2.0857 - val_accuracy: 0.3511
Epoch 20/100
230/230 [==============================] - 6s 25ms/step - loss: 2.0080 -
accuracy: 0.3764 - val_loss: 1.9976 - val_accuracy: 0.3892
Epoch 21/100
230/230 [==============================] - 8s 34ms/step - loss: 1.9501 -
accuracy: 0.4006 - val_loss: 1.9471 - val_accuracy: 0.4192
Epoch 22/100
230/230 [==============================] - 5s 24ms/step - loss: 1.8516 -
accuracy: 0.4308 - val_loss: 1.8448 - val_accuracy: 0.4491
Epoch 23/100
230/230 [==============================] - 8s 33ms/step - loss: 1.8403 -
accuracy: 0.4406 - val_loss: 1.8490 - val_accuracy: 0.4273
Epoch 24/100
230/230 [==============================] - 6s 25ms/step - loss: 1.7508 -
accuracy: 0.4669 - val_loss: 1.7319 - val_accuracy: 0.4932
Epoch 25/100
230/230 [==============================] - 6s 28ms/step - loss: 1.7503 -
```

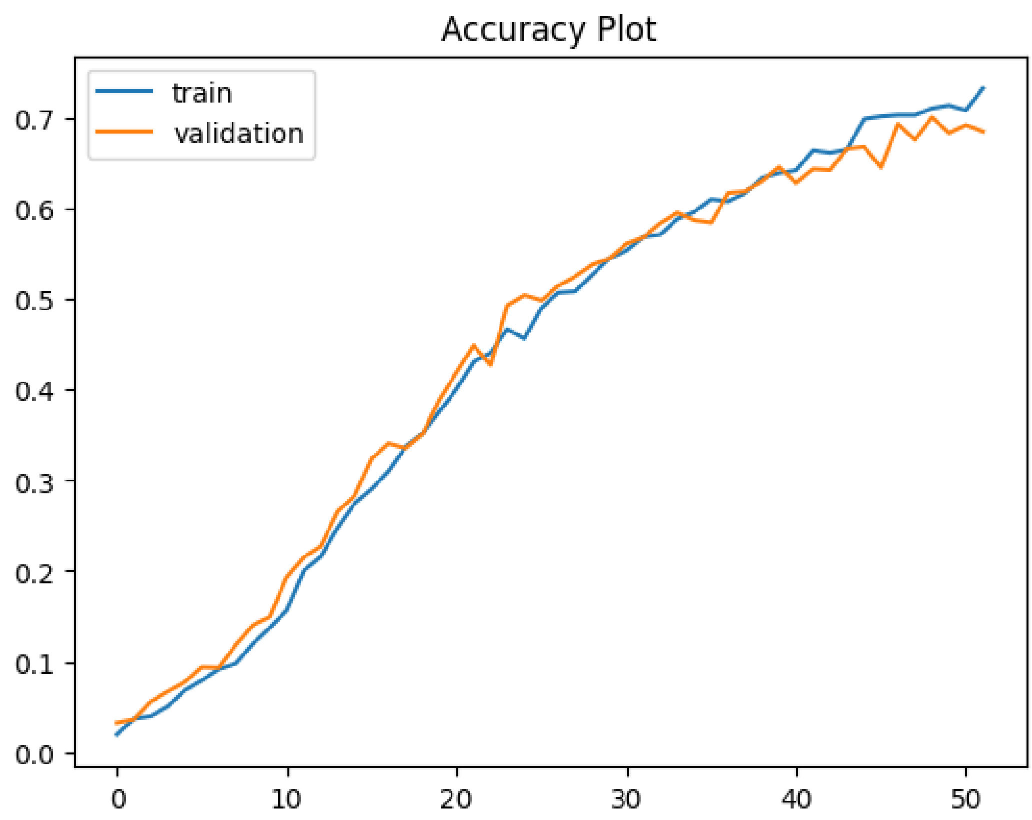
```
accuracy: 0.4562 - val_loss: 1.6921 - val_accuracy: 0.5046
Epoch 26/100
230/230 [==============================] - 7s 32ms/step - loss: 1.6505 -
accuracy: 0.4906 - val_loss: 1.6588 - val_accuracy: 0.4986
Epoch 27/100
230/230 [==============================] - 6s 24ms/step - loss: 1.5878 -
accuracy: 0.5074 - val_loss: 1.6278 - val_accuracy: 0.5150
Epoch 28/100
230/230 [==============================] - 8s 33ms/step - loss: 1.5702 -
accuracy: 0.5087 - val_loss: 1.6215 - val_accuracy: 0.5253
Epoch 29/100
230/230 [==============================] - 6s 24ms/step - loss: 1.5098 -
accuracy: 0.5272 - val_loss: 1.5272 - val_accuracy: 0.5384
Epoch 30/100
230/230 [==============================] - 8s 34ms/step - loss: 1.4530 -
accuracy: 0.5452 - val_loss: 1.4939 - val_accuracy: 0.5449
Epoch 31/100
230/230 [==============================] - 6s 26ms/step - loss: 1.4235 -
accuracy: 0.5536 - val_loss: 1.4532 - val_accuracy: 0.5612
Epoch 32/100
230/230 [==============================] - 9s 38ms/step - loss: 1.3776 -
accuracy: 0.5689 - val_loss: 1.4207 - val_accuracy: 0.5683
Epoch 33/100
230/230 [==============================] - 9s 38ms/step - loss: 1.3623 -
accuracy: 0.5715 - val_loss: 1.3973 - val_accuracy: 0.5841
Epoch 34/100
230/230 [==============================] - 5s 23ms/step - loss: 1.3205 -
accuracy: 0.5886 - val_loss: 1.3861 - val_accuracy: 0.5955
Epoch 35/100
230/230 [==============================] - 8s 35ms/step - loss: 1.3162 -
accuracy: 0.5965 - val_loss: 1.4134 - val_accuracy: 0.5874
Epoch 36/100
230/230 [==============================] - 6s 24ms/step - loss: 1.2758 -
accuracy: 0.6103 - val_loss: 1.3703 - val_accuracy: 0.5846
Epoch 37/100
230/230 [==============================] - 8s 34ms/step - loss: 1.2594 -
accuracy: 0.6079 - val_loss: 1.3232 - val_accuracy: 0.6173
Epoch 38/100
230/230 [==============================] - 6s 28ms/step - loss: 1.2538 -
accuracy: 0.6172 - val_loss: 1.3042 - val_accuracy: 0.6189
Epoch 39/100
230/230 [==============================] - 6s 26ms/step - loss: 1.2036 -
accuracy: 0.6342 - val_loss: 1.2918 - val_accuracy: 0.6309
Epoch 40/100
230/230 [==============================] - 7s 31ms/step - loss: 1.1799 -
accuracy: 0.6395 - val_loss: 1.2506 - val_accuracy: 0.6462
Epoch 41/100
230/230 [==============================] - 5s 24ms/step - loss: 1.1737 -
```

```
accuracy: 0.6425 - val_loss: 1.2694 - val_accuracy: 0.6287
Epoch 42/100
230/230 [==============================] - 8s 35ms/step - loss: 1.1052 -
accuracy: 0.6644 - val_loss: 1.2296 - val_accuracy: 0.6440
Epoch 43/100
230/230 [==============================] - 6s 26ms/step - loss: 1.1176 -
accuracy: 0.6617 - val_loss: 1.2405 - val_accuracy: 0.6429
Epoch 44/100
230/230 [==============================] - 8s 33ms/step - loss: 1.1162 -
accuracy: 0.6654 - val_loss: 1.1781 - val_accuracy: 0.6663
Epoch 45/100
230/230 [==============================] - 6s 25ms/step - loss: 1.0243 -
accuracy: 0.6992 - val_loss: 1.1823 - val_accuracy: 0.6685
Epoch 46/100
230/230 [==============================] - 6s 26ms/step - loss: 1.0189 -
accuracy: 0.7020 - val_loss: 1.2504 - val_accuracy: 0.6456
Epoch 47/100
230/230 [==============================] - 7s 32ms/step - loss: 0.9995 -
accuracy: 0.7035 - val_loss: 1.1060 - val_accuracy: 0.6935
Epoch 48/100
230/230 [==============================] - 6s 25ms/step - loss: 1.0028 -
accuracy: 0.7035 - val_loss: 1.1554 - val_accuracy: 0.6761
Epoch 49/100
230/230 [==============================] - 8s 34ms/step - loss: 0.9643 -
accuracy: 0.7107 - val_loss: 1.0804 - val_accuracy: 0.7011
Epoch 50/100
230/230 [==============================] - 6s 25ms/step - loss: 0.9423 -
accuracy: 0.7140 - val_loss: 1.1377 - val_accuracy: 0.6837
Epoch 51/100
230/230 [==============================] - 7s 29ms/step - loss: 0.9571 -
accuracy: 0.7085 - val_loss: 1.0972 - val_accuracy: 0.6924
Epoch 52/100
230/230 [==============================] - 7s 30ms/step - loss: 0.8994 -
accuracy: 0.7332 - val_loss: 1.1023 - val_accuracy: 0.6854
Training time: 421.2397985458374 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 9ms/step - loss: 1.1089 - accuracy:
0.7000
Test Loss: 1.108872652053833
Test Accuracy: 70.0
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```python
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 1s 9ms/step
Precision: 70.48
Recall: 70.0
F1 Score: 68.02
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[ ]:
```python
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support
```

0	0.92	0.90	0.91	40
1	0.90	0.95	0.93	40
2	1.00	0.97	0.99	40
3	0.72	0.33	0.45	40
4	0.96	0.68	0.79	40
5	0.47	0.68	0.55	40
6	0.81	0.85	0.83	40
7	0.50	0.35	0.41	40
8	0.80	0.80	0.80	40
9	0.95	0.93	0.94	40
10	0.78	0.53	0.63	40
11	0.52	0.65	0.58	40
12	0.90	0.47	0.62	40
13	0.62	0.88	0.73	40
14	0.43	0.70	0.53	40
15	0.71	0.88	0.79	40
16	0.56	0.70	0.62	40
17	0.84	0.80	0.82	40
18	0.67	0.72	0.70	40
19	0.80	0.90	0.85	40
20	0.72	0.72	0.73	40
21	0.72	0.78	0.75	40
22	0.64	0.80	0.71	40
23	0.00	0.00	0.00	40
24	0.81	0.95	0.87	40
25	0.58	0.78	0.67	40
26	0.76	0.88	0.81	40
27	0.75	0.75	0.75	40
28	0.73	0.68	0.70	40
29	0.41	0.85	0.55	40
30	0.79	0.85	0.82	40
31	0.57	0.80	0.67	40
32	0.93	0.95	0.94	40
33	0.77	0.75	0.76	40
34	0.88	0.70	0.78	40
35	0.56	0.72	0.63	40
36	0.85	0.85	0.85	40
37	0.00	0.00	0.00	40
38	0.61	0.78	0.68	40
39	0.47	0.60	0.53	40
40	0.64	0.95	0.77	40
41	0.49	0.70	0.58	40
42	0.71	0.93	0.80	40
43	0.40	0.53	0.45	40
44	0.94	0.82	0.88	40
45	0.77	0.85	0.81	40
46	0.76	0.85	0.80	40
47	0.71	0.60	0.65	40

48	0.68	0.65	0.67	40
49	0.82	0.45	0.58	40
50	0.82	0.57	0.68	40
51	0.70	0.82	0.76	40
52	0.74	0.57	0.65	40
53	0.68	0.53	0.59	40
54	0.73	0.82	0.78	40
55	0.95	0.97	0.96	40
56	0.82	0.82	0.82	40
57	0.97	0.85	0.91	40
58	0.69	0.60	0.64	40
59	0.52	0.78	0.62	40
60	0.76	0.88	0.81	40
61	0.82	0.57	0.68	40
62	0.63	0.55	0.59	40
63	0.69	0.78	0.73	40
64	0.91	0.78	0.84	40
65	0.61	0.70	0.65	40
66	0.83	0.25	0.38	40
67	0.75	0.68	0.71	40
68	0.50	0.03	0.05	40
69	0.00	0.00	0.00	40
70	0.83	1.00	0.91	40
71	0.97	0.80	0.88	40
72	1.00	0.05	0.10	40
73	0.95	0.90	0.92	40
74	0.35	0.97	0.51	40
75	0.84	0.80	0.82	40
76	0.84	0.68	0.75	40
accuracy			0.70	3080
macro avg	0.70	0.70	0.68	3080
weighted avg	0.70	0.70	0.68	3080

```
The number of misclassifications: 924
Proportion of misclassifications: 30.0%
Input Text: locate card
Actual Label: 11
Predicted Label: 43

Input Text: still received new card ordered week ago
Actual Label: 11
Predicted Label: 13

Input Text: get card
Actual Label: 11
Predicted Label: 39
```

Input Text: know tracking number new card sent
Actual Label: 11
Predicted Label: 13

Input Text: received card
Actual Label: 11
Predicted Label: 43

Input Text: still waiting card
Actual Label: 11
Predicted Label: 43

Input Text: track card
Actual Label: 11
Predicted Label: 43

Input Text: still dont card weeks
Actual Label: 11
Predicted Label: 43

Input Text: ive waiting longer expected card
Actual Label: 11
Predicted Label: 43

Input Text: hasnt card delivered
Actual Label: 11
Predicted Label: 43

Input Text: card still hasnt arrived weeks lost
Actual Label: 11
Predicted Label: 13

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 13

Input Text: status card ordered
Actual Label: 11
Predicted Label: 43

Input Text: im starting think card lost still hasnt arrived help
Actual Label: 11
Predicted Label: 13

Input Text: would like reactivate card
Actual Label: 13
Predicted Label: 0

```
Input Text: add card account
Actual Label: 13
Predicted Label: 24

Input Text: link credit card
Actual Label: 13
Predicted Label: 11

Input Text: link replacement card
Actual Label: 13
Predicted Label: 11

Input Text: link another card account
Actual Label: 13
Predicted Label: 43

Input Text: good time exchange
Actual Label: 32
Predicted Label: 31

Input Text: kind foreign exchange rate get exchange money
Actual Label: 32
Predicted Label: 50

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: rate exchange card payment incorrect
Actual Label: 17
Predicted Label: 76

Input Text: bought something overseas wrong exchange rate statement
Actual Label: 17
Predicted Label: 76

Input Text: exchange rate card payment wrong
Actual Label: 17
Predicted Label: 76

Input Text: charged
Actual Label: 17
Predicted Label: 15

Input Text: believe card payment exchange rate incorrect
Actual Label: 17
Predicted Label: 76
```

```
Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 25

Input Text: exchange rate totally wrong card payment
Actual Label: 17
Predicted Label: 76

Input Text: extra dollar charged account
Actual Label: 34
Predicted Label: 19


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

1.5.4 Hyperparameter tuning

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Define the model for hyperparameter tuning
def model_builder(hp):
  model = keras.models.Sequential()
  e = layers.Embedding(voca_size+1, 300, weights=[embedding_matrix],␣
↪input_length=max_length_train_text, trainable=False)
  model.add(e)
  hp_units = hp.Int('units', min_value = 20, max_value = 50, step = 10) # Set␣
↪up the hyperparameters
  model.add(layers.LSTM(units = hp_units)) # We will check the optimal hidden␣
↪unit for the LSTM layer
  model.add(layers.Dense(nlabel, activation='softmax'))

  hp_learning_rate = hp.Choice('learning_rate', values = [0.01, 0.001, 0.0001])␣
↪# Set up the hyperparameters
```

```
    model.compile(optimizer = keras.optimizers.Adam(learning_rate =␣
↪hp_learning_rate), # We will check the optimal learning rate
                  loss = 'sparse_categorical_crossentropy',
                  metrics = ['accuracy'])
    return model
```

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Specify the tuner
tuner = kt.Hyperband(model_builder,
                     objective = 'val_accuracy',
                     max_epochs = 100)
```

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Set up a callback for early stopping
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
```

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Run the tuner
tuner.search(X_train_padded, y_train, epochs = 100, validation_data =␣
↪(X_val_padded, y_val), callbacks = [stop_early])

# Get the optimal hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials = 1)[0]

print(f"The optimal number of units: {best_hps.get('units')}. The optimal␣
↪learning rate: {best_hps.get('learning_rate')}.")
```

```
Trial 12 Complete [00h 00m 13s]
val_accuracy: 0.03647251054644585

Best val_accuracy So Far: 0.17038649320602417
Total elapsed time: 00h 24m 36s
The optimal number of units: 50. The optimal learning rate: 0.01.
```

1.5.5 Tuned LSTM

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
nlabel = 77

tuned_model = keras.models.Sequential()
```

```
e = layers.Embedding(voca_size+1, 300, weights=[embedding_matrix],␣
 ↪input_length=max_length_train_text, trainable=False)
tuned_model.add(e)
tuned_model.add(layers.LSTM(50))
tuned_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
tuned_model.compile(optimizer=keras.optimizers.Adam(learning_rate = 0.01),␣
 ↪loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Summary the model
tuned_model.summary()
```

```
Model: "sequential_8"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 29, 300)           626700

 lstm_8 (LSTM)               (None, 50)                70200

 dense_8 (Dense)             (None, 77)                3927

=================================================================
Total params: 700827 (2.67 MB)
Trainable params: 74127 (289.56 KB)
Non-trainable params: 626700 (2.39 MB)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'tuned_LSTM_word2vec_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()
```

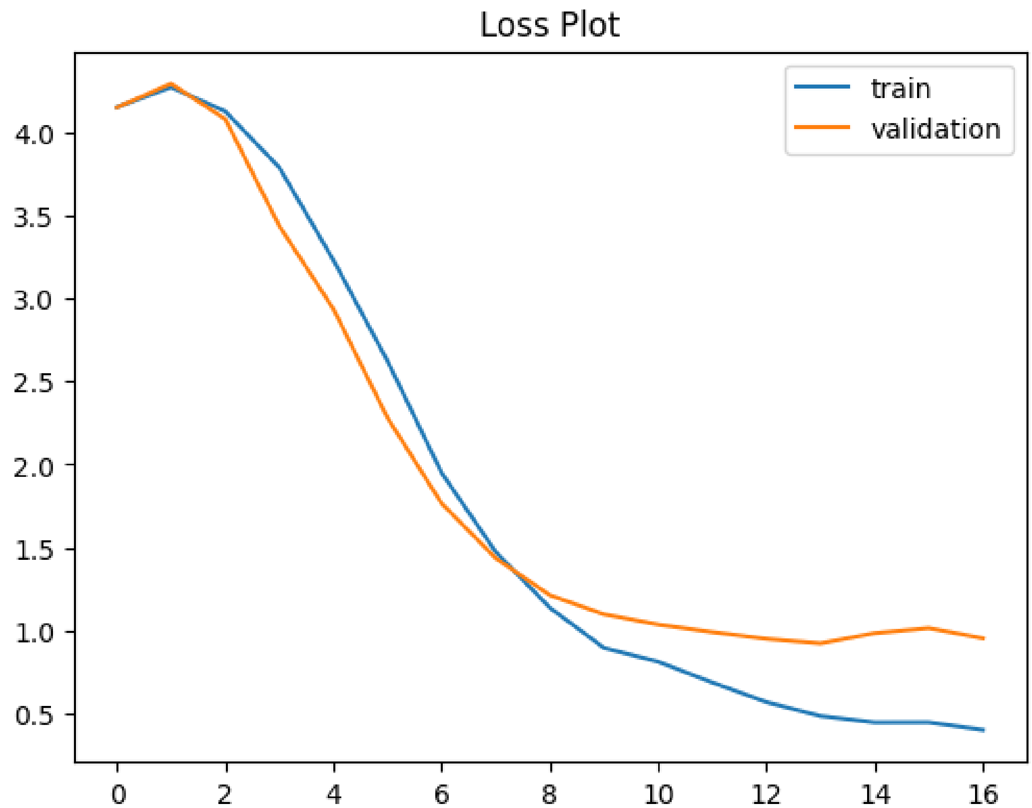
```python
# Fit the model
history = tuned_model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 14s 49ms/step - loss: 4.1499 -
accuracy: 0.0271 - val_loss: 4.1483 - val_accuracy: 0.0299
Epoch 2/100
230/230 [==============================] - 11s 48ms/step - loss: 4.2695 -
accuracy: 0.0267 - val_loss: 4.2910 - val_accuracy: 0.0240
Epoch 3/100
230/230 [==============================] - 7s 29ms/step - loss: 4.1253 -
accuracy: 0.0253 - val_loss: 4.0790 - val_accuracy: 0.0289
Epoch 4/100
230/230 [==============================] - 10s 45ms/step - loss: 3.7872 -
accuracy: 0.0501 - val_loss: 3.4356 - val_accuracy: 0.0806
Epoch 5/100
230/230 [==============================] - 8s 33ms/step - loss: 3.2299 -
accuracy: 0.1010 - val_loss: 2.9360 - val_accuracy: 0.1595
Epoch 6/100
230/230 [==============================] - 10s 44ms/step - loss: 2.6225 -
accuracy: 0.2332 - val_loss: 2.2807 - val_accuracy: 0.3005
Epoch 7/100
230/230 [==============================] - 9s 38ms/step - loss: 1.9497 -
accuracy: 0.3935 - val_loss: 1.7657 - val_accuracy: 0.4605
Epoch 8/100
230/230 [==============================] - 12s 52ms/step - loss: 1.4739 -
accuracy: 0.5501 - val_loss: 1.4363 - val_accuracy: 0.5781
Epoch 9/100
230/230 [==============================] - 11s 47ms/step - loss: 1.1382 -
accuracy: 0.6572 - val_loss: 1.2136 - val_accuracy: 0.6587
Epoch 10/100
230/230 [==============================] - 8s 35ms/step - loss: 0.8966 -
accuracy: 0.7374 - val_loss: 1.0990 - val_accuracy: 0.6973
Epoch 11/100
```

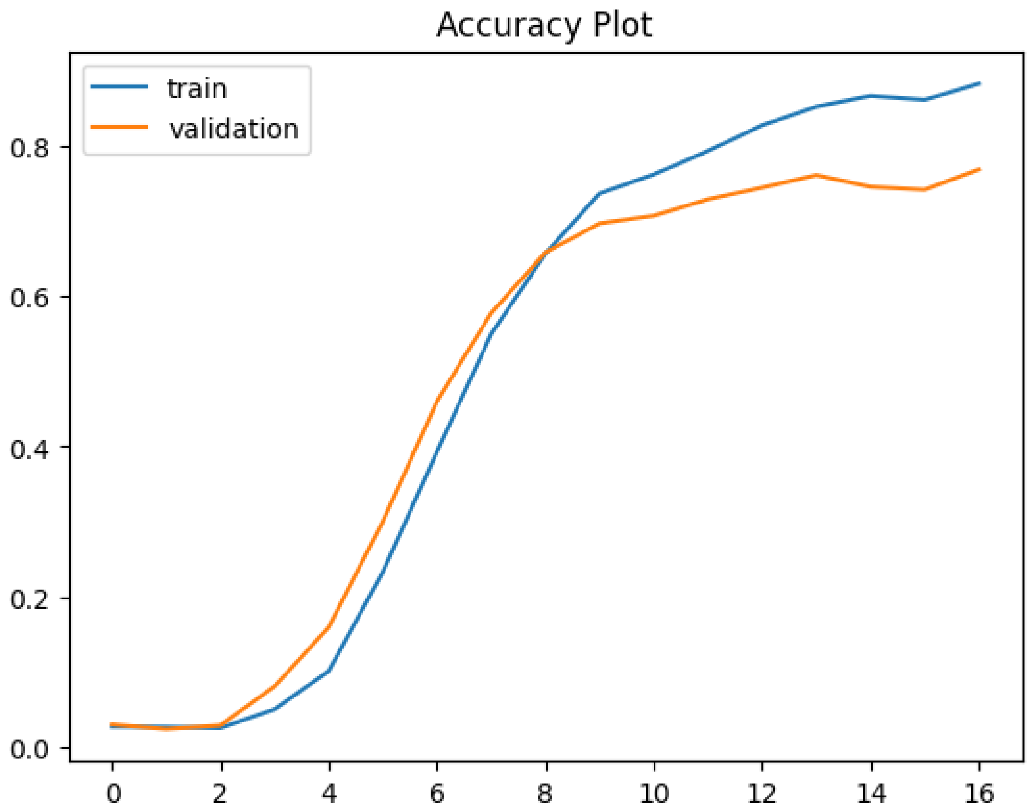
```
230/230 [==============================] - 10s 42ms/step - loss: 0.8136 -
accuracy: 0.7626 - val_loss: 1.0372 - val_accuracy: 0.7077
Epoch 12/100
230/230 [==============================] - 9s 40ms/step - loss: 0.6888 -
accuracy: 0.7939 - val_loss: 0.9918 - val_accuracy: 0.7295
Epoch 13/100
230/230 [==============================] - 8s 36ms/step - loss: 0.5707 -
accuracy: 0.8281 - val_loss: 0.9511 - val_accuracy: 0.7452
Epoch 14/100
230/230 [==============================] - 10s 44ms/step - loss: 0.4869 -
accuracy: 0.8528 - val_loss: 0.9241 - val_accuracy: 0.7616
Epoch 15/100
230/230 [==============================] - 7s 30ms/step - loss: 0.4483 -
accuracy: 0.8673 - val_loss: 0.9850 - val_accuracy: 0.7463
Epoch 16/100
230/230 [==============================] - 11s 47ms/step - loss: 0.4489 -
accuracy: 0.8620 - val_loss: 1.0156 - val_accuracy: 0.7425
Epoch 17/100
230/230 [==============================] - 7s 32ms/step - loss: 0.4037 -
accuracy: 0.8836 - val_loss: 0.9544 - val_accuracy: 0.7692
Training time: 161.79024744033813 seconds
```

[ ]:

```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 3s 22ms/step - loss: 0.8925 - accuracy:
0.7731
Test Loss: 0.892530083656311
Test Accuracy: 77.31
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 2s 12ms/step
Precision: 78.87
Recall: 77.31
F1 Score: 77.25
```

[ ]:
```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94        40
           1       0.85      0.97      0.91        40
           2       0.97      0.97      0.97        40
           3       0.59      0.68      0.63        40
           4       0.85      0.70      0.77        40
           5       0.66      0.68      0.67        40
```

6	0.74	0.78	0.76	40
7	0.71	0.72	0.72	40
8	0.86	0.80	0.83	40
9	0.87	1.00	0.93	40
10	0.69	0.50	0.58	40
11	0.67	0.78	0.72	40
12	0.68	0.75	0.71	40
13	0.92	0.90	0.91	40
14	0.55	0.82	0.66	40
15	0.74	0.85	0.79	40
16	0.56	0.55	0.56	40
17	0.78	0.90	0.84	40
18	0.91	0.72	0.81	40
19	0.77	0.93	0.84	40
20	0.50	0.72	0.59	40
21	0.97	0.78	0.86	40
22	0.67	0.65	0.66	40
23	0.97	0.85	0.91	40
24	0.85	0.97	0.91	40
25	0.75	0.82	0.79	40
26	0.73	0.93	0.81	40
27	0.88	0.72	0.79	40
28	0.82	0.70	0.76	40
29	0.86	0.78	0.82	40
30	0.88	0.93	0.90	40
31	0.92	0.82	0.87	40
32	0.85	0.88	0.86	40
33	0.77	0.85	0.81	40
34	0.79	0.78	0.78	40
35	0.68	0.68	0.68	40
36	0.79	0.78	0.78	40
37	0.54	0.80	0.65	40
38	0.87	0.97	0.92	40
39	0.93	0.70	0.80	40
40	0.81	0.75	0.78	40
41	0.73	0.55	0.63	40
42	0.97	0.88	0.92	40
43	0.73	0.68	0.70	40
44	0.98	1.00	0.99	40
45	0.78	0.72	0.75	40
46	0.76	0.78	0.77	40
47	0.76	0.78	0.77	40
48	0.56	0.72	0.63	40
49	0.78	0.72	0.75	40
50	1.00	0.62	0.77	40
51	0.84	0.90	0.87	40
52	0.81	0.88	0.84	40
53	0.64	0.57	0.61	40

```
          54       0.84      0.78      0.81        40
          55       0.97      0.88      0.92        40
          56       0.88      0.70      0.78        40
          57       1.00      0.90      0.95        40
          58       0.63      0.68      0.65        40
          59       0.83      0.60      0.70        40
          60       0.85      0.85      0.85        40
          61       0.60      0.68      0.64        40
          62       0.76      0.55      0.64        40
          63       0.75      0.82      0.79        40
          64       0.75      0.82      0.79        40
          65       0.88      0.55      0.68        40
          66       0.58      0.72      0.64        40
          67       0.85      0.57      0.69        40
          68       0.81      0.75      0.78        40
          69       0.62      0.25      0.36        40
          70       0.84      0.95      0.89        40
          71       0.93      0.97      0.95        40
          72       0.86      0.47      0.61        40
          73       0.97      0.88      0.92        40
          74       0.44      0.88      0.58        40
          75       0.74      0.88      0.80        40
          76       0.86      0.80      0.83        40

    accuracy                           0.77      3080
   macro avg       0.79      0.77      0.77      3080
weighted avg       0.79      0.77      0.77      3080

The number of misclassifications: 699
Proportion of misclassifications: 22.69%
Input Text: ordered card arrived help please
Actual Label: 11
Predicted Label: 12

Input Text: get card
Actual Label: 11
Predicted Label: 43

Input Text: waiting longer expected bank card could provide information arrive
Actual Label: 11
Predicted Label: 27

Input Text: hasnt card delivered
Actual Label: 11
Predicted Label: 12

Input Text: status card ordered
Actual Label: 11
```

Predicted Label: 14

Input Text: expecting new card wondering havent received yet
Actual Label: 11
Predicted Label: 40

Input Text: know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: im starting think card lost still hasnt arrived help
Actual Label: 11
Predicted Label: 41

Input Text: tracking info available
Actual Label: 11
Predicted Label: 13

Input Text: add card account
Actual Label: 13
Predicted Label: 18

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 41

Input Text: hello found card misplaced need reactive
Actual Label: 13
Predicted Label: 49

Input Text: already one cards link
Actual Label: 13
Predicted Label: 40

Input Text: good time exchange
Actual Label: 32
Predicted Label: 33

Input Text: currencies exchange rate calculated
Actual Label: 32
Predicted Label: 31

Input Text: im trying figure current exchange rate
Actual Label: 32
Predicted Label: 76

Input Text: kind foreign exchange rate get exchange money
Actual Label: 32

```
Predicted Label: 31

Input Text: rate get determined
Actual Label: 32
Predicted Label: 17

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: rate low sure using right exchange rate
Actual Label: 17
Predicted Label: 32

Input Text: charged
Actual Label: 17
Predicted Label: 15

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 63

Input Text: explain random charge
Actual Label: 34
Predicted Label: 63

Input Text: transaction credited
Actual Label: 34
Predicted Label: 8

Input Text: fee come
Actual Label: 34
Predicted Label: 19

Input Text: extra charge
Actual Label: 34
Predicted Label: 15

Input Text: extra pound charge card
Actual Label: 34
Predicted Label: 15

Input Text: euro fee come
Actual Label: 34
Predicted Label: 17

Input Text: euro fee statement
Actual Label: 34
```

```
Predicted Label: 15

Input Text: reason accounts charged extra dollar
Actual Label: 34
Predicted Label: 15
```

1.5.6 Tuned LSTM (with dropout)

```
[ ]: # Define the output dimension for the embedding layer and hidden units
nlabel = 77

dropout_tuned_model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 300, weights=[embedding_matrix],␣
 ↪input_length=max_length_train_text, trainable=False)
dropout_tuned_model.add(e)
dropout_tuned_model.add(layers.LSTM(50, dropout=0.2))
dropout_tuned_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
dropout_tuned_model.compile(optimizer=keras.optimizers.Adam(learning_rate = 0.
 ↪01), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Summary the model
dropout_tuned_model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 29, 300)           626700

 lstm_9 (LSTM)               (None, 50)                70200

 dense_9 (Dense)             (None, 77)                3927

=================================================================
Total params: 700827 (2.67 MB)
Trainable params: 74127 (289.56 KB)
Non-trainable params: 626700 (2.39 MB)
_________________________________________________________________
```

```
[ ]: # Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'dropout_tuned_LSTM_word2vec_model.keras'
```

```
# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Get the weights of the embedding layer
embedding_weights = dropout_tuned_model.layers[0].get_weights()[0]

# Define the file path to save the weights
embedding_weights_file = '/content/drive/MyDrive/1. NLP CW/embedding_weights.
 ↪npy'

# Save the weights as a file
np.save(embedding_weights_file, embedding_weights)
```

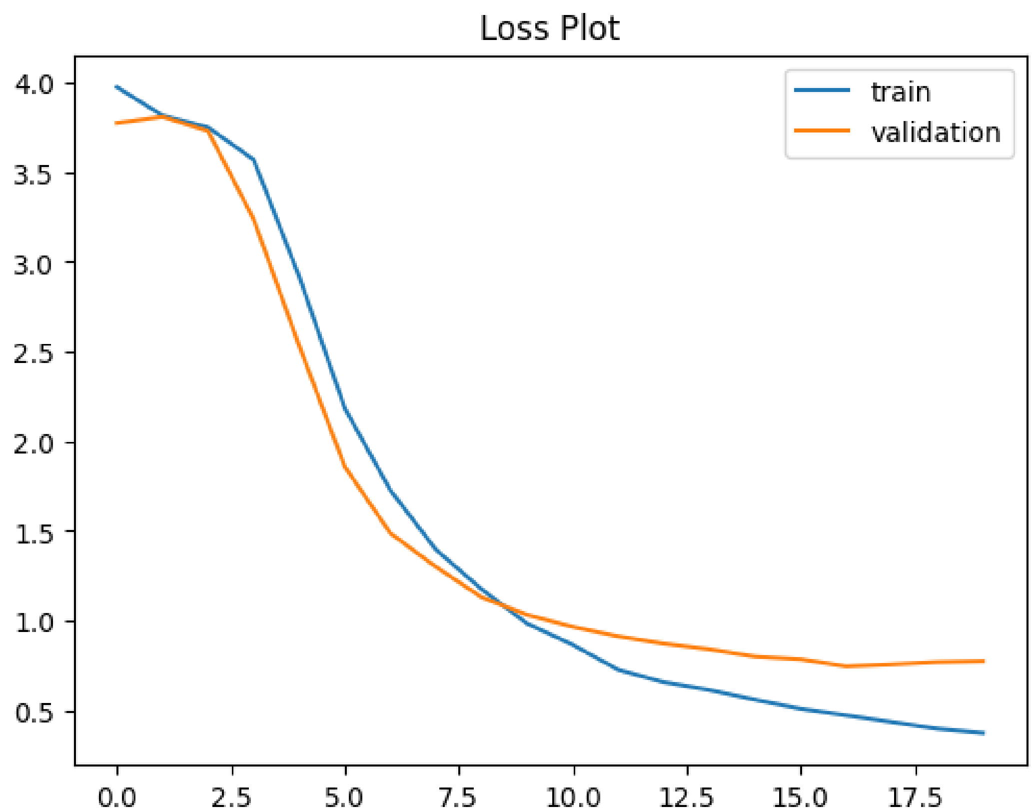
[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = dropout_tuned_model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 16s 53ms/step - loss: 3.9733 -
accuracy: 0.0348 - val_loss: 3.7733 - val_accuracy: 0.0474
Epoch 2/100
230/230 [==============================] - 10s 43ms/step - loss: 3.8154 -
accuracy: 0.0566 - val_loss: 3.8087 - val_accuracy: 0.0637
Epoch 3/100
230/230 [==============================] - 10s 41ms/step - loss: 3.7495 -
```

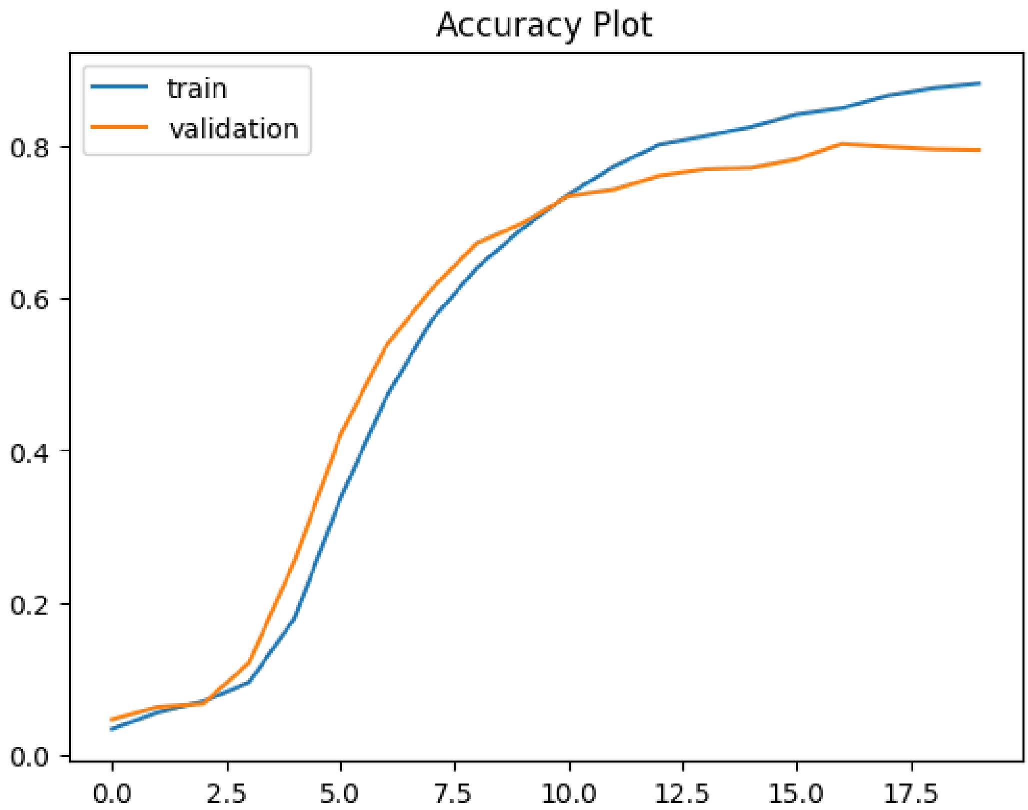
```
accuracy: 0.0709 - val_loss: 3.7288 - val_accuracy: 0.0680
Epoch 4/100
230/230 [==============================] - 11s 48ms/step - loss: 3.5694 -
accuracy: 0.0957 - val_loss: 3.2373 - val_accuracy: 0.1214
Epoch 5/100
230/230 [==============================] - 8s 37ms/step - loss: 2.9182 -
accuracy: 0.1797 - val_loss: 2.5324 - val_accuracy: 0.2548
Epoch 6/100
230/230 [==============================] - 11s 48ms/step - loss: 2.1847 -
accuracy: 0.3357 - val_loss: 1.8590 - val_accuracy: 0.4192
Epoch 7/100
230/230 [==============================] - 12s 54ms/step - loss: 1.7277 -
accuracy: 0.4684 - val_loss: 1.4882 - val_accuracy: 0.5362
Epoch 8/100
230/230 [==============================] - 9s 41ms/step - loss: 1.3967 -
accuracy: 0.5702 - val_loss: 1.3007 - val_accuracy: 0.6113
Epoch 9/100
230/230 [==============================] - 11s 49ms/step - loss: 1.1765 -
accuracy: 0.6394 - val_loss: 1.1318 - val_accuracy: 0.6717
Epoch 10/100
230/230 [==============================] - 11s 48ms/step - loss: 0.9851 -
accuracy: 0.6911 - val_loss: 1.0336 - val_accuracy: 0.6979
Epoch 11/100
230/230 [==============================] - 9s 38ms/step - loss: 0.8667 -
accuracy: 0.7352 - val_loss: 0.9671 - val_accuracy: 0.7333
Epoch 12/100
230/230 [==============================] - 11s 46ms/step - loss: 0.7273 -
accuracy: 0.7721 - val_loss: 0.9134 - val_accuracy: 0.7420
Epoch 13/100
230/230 [==============================] - 10s 43ms/step - loss: 0.6583 -
accuracy: 0.8008 - val_loss: 0.8731 - val_accuracy: 0.7599
Epoch 14/100
230/230 [==============================] - 9s 38ms/step - loss: 0.6147 -
accuracy: 0.8119 - val_loss: 0.8417 - val_accuracy: 0.7686
Epoch 15/100
230/230 [==============================] - 12s 51ms/step - loss: 0.5607 -
accuracy: 0.8238 - val_loss: 0.8010 - val_accuracy: 0.7703
Epoch 16/100
230/230 [==============================] - 9s 39ms/step - loss: 0.5102 -
accuracy: 0.8403 - val_loss: 0.7857 - val_accuracy: 0.7817
Epoch 17/100
230/230 [==============================] - 11s 48ms/step - loss: 0.4737 -
accuracy: 0.8486 - val_loss: 0.7466 - val_accuracy: 0.8019
Epoch 18/100
230/230 [==============================] - 11s 49ms/step - loss: 0.4363 -
accuracy: 0.8650 - val_loss: 0.7574 - val_accuracy: 0.7980
Epoch 19/100
230/230 [==============================] - 9s 38ms/step - loss: 0.3991 -
```

```
accuracy: 0.8748 - val_loss: 0.7701 - val_accuracy: 0.7948
Epoch 20/100
230/230 [==============================] - 11s 48ms/step - loss: 0.3757 -
accuracy: 0.8808 - val_loss: 0.7752 - val_accuracy: 0.7937
Training time: 210.85999464988708 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 4s 25ms/step - loss: 0.7372 - accuracy:
0.7984
Test Loss: 0.7371559143066406
Test Accuracy: 79.84
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 2s 13ms/step
Precision: 81.46
Recall: 79.84
F1 Score: 79.54
```

[ ]:
```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.93        40
           1       0.90      0.95      0.93        40
           2       0.95      1.00      0.98        40
           3       0.58      0.70      0.64        40
           4       1.00      0.85      0.92        40
           5       0.70      0.65      0.68        40
```

6	1.00	0.75	0.86	40
7	0.79	0.68	0.73	40
8	0.73	0.90	0.81	40
9	0.85	0.97	0.91	40
10	0.79	0.57	0.67	40
11	0.87	0.68	0.76	40
12	0.65	0.90	0.76	40
13	0.93	0.95	0.94	40
14	0.81	0.85	0.83	40
15	0.80	0.80	0.80	40
16	0.50	0.57	0.53	40
17	0.79	0.93	0.85	40
18	0.92	0.82	0.87	40
19	0.88	0.95	0.92	40
20	0.67	0.90	0.77	40
21	0.97	0.95	0.96	40
22	0.67	0.72	0.70	40
23	1.00	0.88	0.93	40
24	0.88	0.93	0.90	40
25	0.83	0.88	0.85	40
26	0.78	0.72	0.75	40
27	0.97	0.70	0.81	40
28	0.88	0.75	0.81	40
29	0.88	0.75	0.81	40
30	0.91	0.97	0.94	40
31	1.00	0.85	0.92	40
32	0.90	0.93	0.91	40
33	0.81	0.85	0.83	40
34	0.79	0.85	0.82	40
35	0.71	0.72	0.72	40
36	0.84	0.78	0.81	40
37	0.63	0.82	0.72	40
38	0.82	0.93	0.87	40
39	0.83	0.88	0.85	40
40	0.74	0.97	0.84	40
41	0.82	0.70	0.76	40
42	0.80	0.88	0.83	40
43	0.82	0.68	0.74	40
44	1.00	1.00	1.00	40
45	0.95	0.88	0.91	40
46	0.71	0.88	0.79	40
47	0.71	0.85	0.77	40
48	0.72	0.72	0.73	40
49	0.88	0.75	0.81	40
50	0.86	0.78	0.82	40
51	0.93	0.68	0.78	40
52	0.61	0.95	0.75	40
53	0.74	0.50	0.60	40

54	0.79	0.78	0.78	40
55	0.93	0.93	0.93	40
56	0.77	0.75	0.76	40
57	0.88	0.88	0.88	40
58	0.56	0.85	0.67	40
59	0.80	0.70	0.75	40
60	0.93	0.95	0.94	40
61	0.77	0.57	0.66	40
62	0.87	0.65	0.74	40
63	0.76	0.88	0.81	40
64	0.79	0.85	0.82	40
65	0.79	0.75	0.77	40
66	0.66	0.68	0.67	40
67	0.68	0.65	0.67	40
68	0.75	0.30	0.43	40
69	0.64	0.23	0.33	40
70	0.95	0.97	0.96	40
71	0.95	1.00	0.98	40
72	0.93	0.33	0.48	40
73	1.00	0.85	0.92	40
74	0.43	0.93	0.58	40
75	0.69	0.82	0.75	40
76	0.97	0.82	0.89	40
accuracy			0.80	3080
macro avg	0.81	0.80	0.80	3080
weighted avg	0.81	0.80	0.80	3080

```
The number of misclassifications: 621
Proportion of misclassifications: 20.16%
Input Text: locate card
Actual Label: 11
Predicted Label: 13

Input Text: way know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: get card
Actual Label: 11
Predicted Label: 12

Input Text: received card
Actual Label: 11
Predicted Label: 12

Input Text: long card delivery take
Actual Label: 11
```

```
Predicted Label: 12

Input Text: hasnt card delivered
Actual Label: 11
Predicted Label: 12

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 41

Input Text: status card ordered
Actual Label: 11
Predicted Label: 12

Input Text: im still waiting delivery new card taking long
Actual Label: 11
Predicted Label: 9

Input Text: know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: im still waiting card delivered
Actual Label: 11
Predicted Label: 12

Input Text: im starting think card lost still hasnt arrived help
Actual Label: 11
Predicted Label: 41

Input Text: tracking info available
Actual Label: 11
Predicted Label: 68

Input Text: add card account
Actual Label: 13
Predicted Label: 39

Input Text: link another card account
Actual Label: 13
Predicted Label: 39

Input Text: good time exchange
Actual Label: 32
Predicted Label: 17

Input Text: currencies exchange rate calculated
Actual Label: 32
```

```
Predicted Label: 17

Input Text: im trying figure current exchange rate
Actual Label: 32
Predicted Label: 17

Input Text: rate low sure using right exchange rate
Actual Label: 17
Predicted Label: 32

Input Text: charged
Actual Label: 17
Predicted Label: 34

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 15

Input Text: explain random charge
Actual Label: 34
Predicted Label: 16

Input Text: transaction credited
Actual Label: 34
Predicted Label: 8

Input Text: fee come
Actual Label: 34
Predicted Label: 15

Input Text: euro fee come
Actual Label: 34
Predicted Label: 29

Input Text: euro fee statement
Actual Label: 34
Predicted Label: 22

Input Text: two weeks transaction reversed
Actual Label: 34
Predicted Label: 63

Input Text: made withdrawal account posted
Actual Label: 46
Predicted Label: 20

Input Text: wheres accounting cash withdrawal
Actual Label: 46
```

```
Predicted Label: 20

Input Text: account charged withdraw tried make decline
Actual Label: 46
Predicted Label: 19
```

1.6 LSTM (with GloVe)

1.6.1 Set up the GloVe model

[ ]:
```
# Define the embedding indez as a dictionary
embeddings_index = dict()

# Load the GLoVe file to the colab notebook
f = open('/content/drive/MyDrive/1. NLP CW/glove.6B.100d.txt', encoding="utf8")
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()

print(f'Total vectors in the GloVe file: {len(embeddings_index)}')
```

```
Total vectors in the GloVe file: 400001
```

[ ]:
```
# Check the dimension of a word
print(f"The dimension of a word in embedding_index:␣
↪{len(embeddings_index['card'])}")
```

```
The dimension of a word in embedding_index: 100
```

[ ]:
```
# Define the embedding matrix
embedding_matrix = np.zeros((voca_size+1, 100)) # For future dimension matching␣
↪with the word_index, add 1 to the vocabulary size, and match 100 from GloVe
print(f'The shape of embedding matrix: {np.shape(embedding_matrix)}')
```

```
The shape of embedding matrix: (2089, 100)
```

[ ]:
```
# Check the index of the word 'card' in word_index
print(f"The index of card in word_index: {word_index['card']}")
```

```
The index of card in word_index: 1
```

[ ]:
```
# Check the word 'card' vectors in the embedding index
print(embeddings_index['card'])
```

```
[ 1.6292e-01 -3.1798e-01  4.2328e-01 -8.6767e-01  4.5101e-01  5.7857e-01
  2.6645e-02 -1.2648e-01  3.3465e-01 -4.2047e-02 -4.0596e-02  1.6478e-01
 -6.7344e-01 -3.3751e-01  3.5913e-01  5.7383e-01  8.4620e-01  3.6374e-01
  3.0630e-01 -6.8050e-02 -6.7610e-01 -1.9147e-01 -1.4594e-01  3.2621e-03
  6.6949e-01 -3.3588e-01  1.7868e-01 -3.9360e-01  1.7700e-01 -3.3642e-01
  1.9288e-01  1.0030e+00 -2.1794e-01  2.4271e-01  1.0935e+00 -1.0303e-01
 -7.9197e-01 -1.3506e-01  1.2156e-01 -9.8377e-01  1.0300e+00 -1.0242e+00
  6.0269e-01 -1.5986e-01 -2.6773e-01 -5.5630e-01  2.5834e-01 -8.5021e-02
 -1.5221e-01 -3.3717e-01  2.6358e-02  2.3171e-01 -1.8056e-01  5.7107e-01
  3.8556e-01 -1.5732e+00 -1.4902e-01  3.7826e-02  1.8485e+00  7.0210e-01
 -1.1697e-01  7.7822e-02  7.4620e-02  9.9570e-02 -2.1427e-01 -6.0061e-01
  9.4903e-02  8.0589e-01  5.5333e-01 -3.1359e-01 -9.0991e-01  5.3645e-02
 -1.4494e-01 -4.8532e-01  1.0335e-01  1.2182e+00 -2.2199e-01 -1.4934e-02
 -1.1355e+00  3.2790e-01  1.1733e+00 -5.2838e-01 -6.6953e-01 -6.2109e-01
 -1.3660e+00 -4.4052e-01 -2.9538e-01 -7.1655e-01  5.9920e-01 -3.4550e-04
 -8.2363e-01  9.3572e-01  6.2134e-01 -2.6649e-01  9.9595e-02 -1.1545e-01
  6.0000e-01  7.2834e-02  6.6487e-01 -6.4510e-01]
```

[ ]:
```
# Check the word 'topup' vectors in the embedding index
print(embeddings_index['topup'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-153-c4db3aaada12> in <cell line: 2>()
      1 # Check the word 'topup' vectors in the embedding index
----> 2 print(embeddings_index['topup'])

KeyError: 'topup'
```

There is no word 'topup'.

[ ]:
```
# Match the index and word to creat an embedding matrix
for word, index in word_index.items():
    vector_value = embeddings_index.get(word)
    if vector_value is not None:
        embedding_matrix[index] = vector_value
```

[ ]:
```
# Check the word 'card' vectors in the embedding matrix
print(embedding_matrix[1])
```

```
[ 1.62919998e-01 -3.17979991e-01  4.23280001e-01 -8.67670000e-01
  4.51009989e-01  5.78570008e-01  2.66449992e-02 -1.26479998e-01
  3.34650010e-01 -4.20470014e-02 -4.05960009e-02  1.64780006e-01
 -6.73439980e-01 -3.37509990e-01  3.59129995e-01  5.73830009e-01
  8.46199989e-01  3.63739997e-01  3.06300014e-01 -6.80499971e-02
 -6.76100016e-01 -1.91469997e-01 -1.45940006e-01  3.26210004e-03
  6.69489980e-01 -3.35880011e-01  1.78680003e-01 -3.93599987e-01
```

```
 1.77000001e-01 -3.36420000e-01  1.92880005e-01  1.00300002e+00
-2.17940003e-01  2.42709994e-01  1.09350002e+00 -1.03030004e-01
-7.91970015e-01 -1.35059997e-01  1.21560000e-01 -9.83770013e-01
 1.02999997e+00 -1.02419996e+00  6.02689981e-01 -1.59860000e-01
-2.67729998e-01 -5.56299984e-01  2.58340001e-01 -8.50209966e-02
-1.52209997e-01 -3.37170005e-01  2.63579991e-02  2.31710002e-01
-1.80559993e-01  5.71070015e-01  3.85560006e-01 -1.57319999e+00
-1.49020001e-01  3.78260016e-02  1.84850001e+00  7.02099979e-01
-1.16970003e-01  7.78219998e-02  7.46200010e-02  9.95699987e-02
-2.14269996e-01 -6.00610018e-01  9.49029997e-02  8.05890024e-01
 5.53330004e-01 -3.13589990e-01 -9.09910023e-01  5.36449999e-02
-1.44940004e-01 -4.85320002e-01  1.03349999e-01  1.21819997e+00
-2.21990004e-01 -1.49339996e-02 -1.13549995e+00  3.27899992e-01
 1.17330003e+00 -5.28379977e-01 -6.69529974e-01 -6.21089995e-01
-1.36600006e+00 -4.40519989e-01 -2.95379996e-01 -7.16549993e-01
 5.99200010e-01 -3.45500011e-04 -8.23629975e-01  9.35720026e-01
 6.21339977e-01 -2.66490012e-01  9.95950028e-02 -1.15450002e-01
 6.00000024e-01  7.28340000e-02  6.64870024e-01 -6.45099998e-01]
```

1.6.2 LSTM (baseline)

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
hidden_unit = 30
nlabel = 77

model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 100, weights=[embedding_matrix],␣
 ↪input_length=max_length_train_text, trainable=False) # Using 100 dimension␣
 ↪for GloVe
model.add(e)
model.add(layers.LSTM(hidden_unit))
model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy']) #, run_eagerly=True

# Summary the model
model.summary()
```

```
Model: "sequential_10"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_10 (Embedding)    (None, 29, 100)           208900

 lstm_10 (LSTM)              (None, 30)                15720
```

```
 dense_10 (Dense)            (None, 77)                2387

=================================================================
Total params: 227007 (886.75 KB)
Trainable params: 18107 (70.73 KB)
Non-trainable params: 208900 (816.02 KB)
_________________________________________________________________
```

[ ]:
```python
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'LSTM_glove_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```python
# Define early stopping
es =  tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3) # Random␣
 ↪number of patience
```

[ ]:
```python
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/100
```

```
230/230 [==============================] - 11s 31ms/step - loss: 4.0977 -
accuracy: 0.0253 - val_loss: 3.8938 - val_accuracy: 0.0267
Epoch 2/100
230/230 [==============================] - 4s 19ms/step - loss: 3.8196 -
accuracy: 0.0376 - val_loss: 3.7330 - val_accuracy: 0.0435
Epoch 3/100
230/230 [==============================] - 4s 19ms/step - loss: 3.6594 -
accuracy: 0.0561 - val_loss: 3.5586 - val_accuracy: 0.0648
Epoch 4/100
230/230 [==============================] - 6s 26ms/step - loss: 3.4984 -
accuracy: 0.0724 - val_loss: 3.4428 - val_accuracy: 0.0768
Epoch 5/100
230/230 [==============================] - 4s 19ms/step - loss: 3.3714 -
accuracy: 0.0779 - val_loss: 3.3062 - val_accuracy: 0.0936
Epoch 6/100
230/230 [==============================] - 4s 19ms/step - loss: 3.2583 -
accuracy: 0.0873 - val_loss: 3.2377 - val_accuracy: 0.0925
Epoch 7/100
230/230 [==============================] - 5s 22ms/step - loss: 3.1858 -
accuracy: 0.1032 - val_loss: 3.1524 - val_accuracy: 0.1132
Epoch 8/100
230/230 [==============================] - 5s 20ms/step - loss: 3.1273 -
accuracy: 0.1171 - val_loss: 3.1484 - val_accuracy: 0.1083
Epoch 9/100
230/230 [==============================] - 4s 18ms/step - loss: 3.0075 -
accuracy: 0.1261 - val_loss: 3.0145 - val_accuracy: 0.1366
Epoch 10/100
230/230 [==============================] - 4s 19ms/step - loss: 2.9693 -
accuracy: 0.1262 - val_loss: 2.9630 - val_accuracy: 0.1334
Epoch 11/100
230/230 [==============================] - 6s 27ms/step - loss: 2.8887 -
accuracy: 0.1364 - val_loss: 2.9393 - val_accuracy: 0.1366
Epoch 12/100
230/230 [==============================] - 5s 20ms/step - loss: 2.8243 -
accuracy: 0.1481 - val_loss: 2.8853 - val_accuracy: 0.1497
Epoch 13/100
230/230 [==============================] - 4s 19ms/step - loss: 2.8182 -
accuracy: 0.1501 - val_loss: 2.8631 - val_accuracy: 0.1595
Epoch 14/100
230/230 [==============================] - 6s 28ms/step - loss: 2.7213 -
accuracy: 0.1695 - val_loss: 2.7291 - val_accuracy: 0.1796
Epoch 15/100
230/230 [==============================] - 4s 19ms/step - loss: 2.6767 -
accuracy: 0.1853 - val_loss: 2.6960 - val_accuracy: 0.1791
Epoch 16/100
230/230 [==============================] - 4s 19ms/step - loss: 2.6447 -
accuracy: 0.1854 - val_loss: 2.6630 - val_accuracy: 0.1981
Epoch 17/100
```

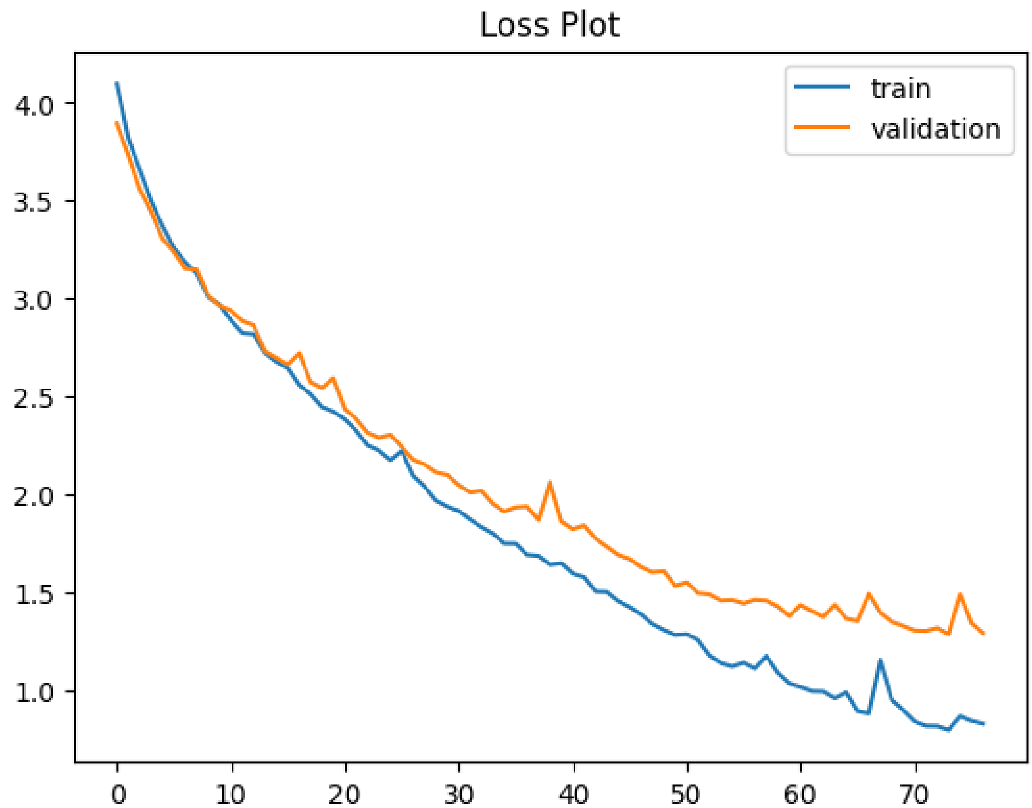
```
230/230 [==============================] - 6s 25ms/step - loss: 2.5574 -
accuracy: 0.2058 - val_loss: 2.7203 - val_accuracy: 0.1747
Epoch 18/100
230/230 [==============================] - 4s 19ms/step - loss: 2.5111 -
accuracy: 0.2171 - val_loss: 2.5729 - val_accuracy: 0.2248
Epoch 19/100
230/230 [==============================] - 4s 16ms/step - loss: 2.4451 -
accuracy: 0.2340 - val_loss: 2.5427 - val_accuracy: 0.2172
Epoch 20/100
230/230 [==============================] - 5s 23ms/step - loss: 2.4228 -
accuracy: 0.2322 - val_loss: 2.5926 - val_accuracy: 0.2041
Epoch 21/100
230/230 [==============================] - 5s 21ms/step - loss: 2.3821 -
accuracy: 0.2520 - val_loss: 2.4346 - val_accuracy: 0.2727
Epoch 22/100
230/230 [==============================] - 4s 18ms/step - loss: 2.3258 -
accuracy: 0.2723 - val_loss: 2.3848 - val_accuracy: 0.2657
Epoch 23/100
230/230 [==============================] - 7s 32ms/step - loss: 2.2492 -
accuracy: 0.2883 - val_loss: 2.3145 - val_accuracy: 0.2782
Epoch 24/100
230/230 [==============================] - 7s 29ms/step - loss: 2.2229 -
accuracy: 0.2875 - val_loss: 2.2886 - val_accuracy: 0.2825
Epoch 25/100
230/230 [==============================] - 5s 23ms/step - loss: 2.1747 -
accuracy: 0.3068 - val_loss: 2.3042 - val_accuracy: 0.2896
Epoch 26/100
230/230 [==============================] - 6s 25ms/step - loss: 2.2218 -
accuracy: 0.3072 - val_loss: 2.2399 - val_accuracy: 0.3114
Epoch 27/100
230/230 [==============================] - 4s 18ms/step - loss: 2.0938 -
accuracy: 0.3252 - val_loss: 2.1766 - val_accuracy: 0.3217
Epoch 28/100
230/230 [==============================] - 5s 20ms/step - loss: 2.0396 -
accuracy: 0.3357 - val_loss: 2.1520 - val_accuracy: 0.3304
Epoch 29/100
230/230 [==============================] - 5s 23ms/step - loss: 1.9676 -
accuracy: 0.3560 - val_loss: 2.1121 - val_accuracy: 0.3386
Epoch 30/100
230/230 [==============================] - 5s 21ms/step - loss: 1.9361 -
accuracy: 0.3601 - val_loss: 2.0977 - val_accuracy: 0.3375
Epoch 31/100
230/230 [==============================] - 4s 18ms/step - loss: 1.9150 -
accuracy: 0.3695 - val_loss: 2.0477 - val_accuracy: 0.3457
Epoch 32/100
230/230 [==============================] - 5s 21ms/step - loss: 1.8701 -
accuracy: 0.3891 - val_loss: 2.0085 - val_accuracy: 0.3778
Epoch 33/100
```

```
230/230 [==============================] - 6s 25ms/step - loss: 1.8323 -
accuracy: 0.3998 - val_loss: 2.0175 - val_accuracy: 0.3702
Epoch 34/100
230/230 [==============================] - 4s 19ms/step - loss: 1.7985 -
accuracy: 0.4123 - val_loss: 1.9501 - val_accuracy: 0.3832
Epoch 35/100
230/230 [==============================] - 4s 19ms/step - loss: 1.7482 -
accuracy: 0.4300 - val_loss: 1.9111 - val_accuracy: 0.3958
Epoch 36/100
230/230 [==============================] - 6s 27ms/step - loss: 1.7472 -
accuracy: 0.4306 - val_loss: 1.9328 - val_accuracy: 0.4012
Epoch 37/100
230/230 [==============================] - 4s 19ms/step - loss: 1.6903 -
accuracy: 0.4510 - val_loss: 1.9369 - val_accuracy: 0.4056
Epoch 38/100
230/230 [==============================] - 4s 19ms/step - loss: 1.6844 -
accuracy: 0.4468 - val_loss: 1.8676 - val_accuracy: 0.4175
Epoch 39/100
230/230 [==============================] - 6s 28ms/step - loss: 1.6393 -
accuracy: 0.4631 - val_loss: 2.0641 - val_accuracy: 0.3767
Epoch 40/100
230/230 [==============================] - 4s 18ms/step - loss: 1.6475 -
accuracy: 0.4624 - val_loss: 1.8607 - val_accuracy: 0.4279
Epoch 41/100
230/230 [==============================] - 5s 20ms/step - loss: 1.5949 -
accuracy: 0.4741 - val_loss: 1.8212 - val_accuracy: 0.4197
Epoch 42/100
230/230 [==============================] - 6s 26ms/step - loss: 1.5768 -
accuracy: 0.4857 - val_loss: 1.8403 - val_accuracy: 0.4415
Epoch 43/100
230/230 [==============================] - 7s 29ms/step - loss: 1.5028 -
accuracy: 0.5140 - val_loss: 1.7729 - val_accuracy: 0.4529
Epoch 44/100
230/230 [==============================] - 4s 19ms/step - loss: 1.5005 -
accuracy: 0.5244 - val_loss: 1.7320 - val_accuracy: 0.4780
Epoch 45/100
230/230 [==============================] - 6s 28ms/step - loss: 1.4529 -
accuracy: 0.5357 - val_loss: 1.6892 - val_accuracy: 0.4823
Epoch 46/100
230/230 [==============================] - 4s 20ms/step - loss: 1.4240 -
accuracy: 0.5520 - val_loss: 1.6691 - val_accuracy: 0.4932
Epoch 47/100
230/230 [==============================] - 4s 18ms/step - loss: 1.3856 -
accuracy: 0.5652 - val_loss: 1.6269 - val_accuracy: 0.5046
Epoch 48/100
230/230 [==============================] - 6s 26ms/step - loss: 1.3377 -
accuracy: 0.5784 - val_loss: 1.6021 - val_accuracy: 0.5161
Epoch 49/100
```

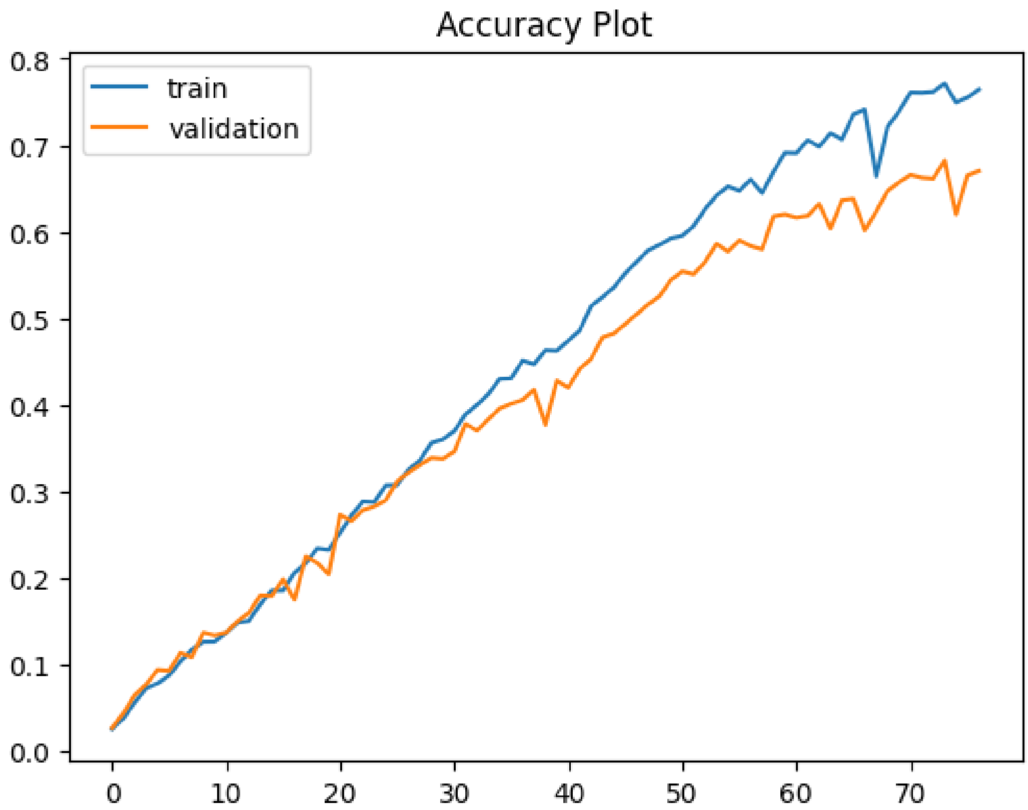
```
230/230 [==============================] - 4s 18ms/step - loss: 1.3058 -
accuracy: 0.5849 - val_loss: 1.6068 - val_accuracy: 0.5253
Epoch 50/100
230/230 [==============================] - 4s 18ms/step - loss: 1.2801 -
accuracy: 0.5923 - val_loss: 1.5313 - val_accuracy: 0.5444
Epoch 51/100
230/230 [==============================] - 5s 24ms/step - loss: 1.2841 -
accuracy: 0.5953 - val_loss: 1.5498 - val_accuracy: 0.5542
Epoch 52/100
230/230 [==============================] - 5s 20ms/step - loss: 1.2555 -
accuracy: 0.6066 - val_loss: 1.4948 - val_accuracy: 0.5509
Epoch 53/100
230/230 [==============================] - 4s 18ms/step - loss: 1.1757 -
accuracy: 0.6262 - val_loss: 1.4880 - val_accuracy: 0.5651
Epoch 54/100
230/230 [==============================] - 5s 21ms/step - loss: 1.1375 -
accuracy: 0.6418 - val_loss: 1.4567 - val_accuracy: 0.5863
Epoch 55/100
230/230 [==============================] - 6s 26ms/step - loss: 1.1200 -
accuracy: 0.6525 - val_loss: 1.4595 - val_accuracy: 0.5770
Epoch 56/100
230/230 [==============================] - 4s 19ms/step - loss: 1.1394 -
accuracy: 0.6474 - val_loss: 1.4413 - val_accuracy: 0.5901
Epoch 57/100
230/230 [==============================] - 5s 20ms/step - loss: 1.1099 -
accuracy: 0.6604 - val_loss: 1.4607 - val_accuracy: 0.5836
Epoch 58/100
230/230 [==============================] - 6s 27ms/step - loss: 1.1738 -
accuracy: 0.6447 - val_loss: 1.4562 - val_accuracy: 0.5797
Epoch 59/100
230/230 [==============================] - 4s 19ms/step - loss: 1.0876 -
accuracy: 0.6695 - val_loss: 1.4259 - val_accuracy: 0.6179
Epoch 60/100
230/230 [==============================] - 4s 18ms/step - loss: 1.0330 -
accuracy: 0.6914 - val_loss: 1.3771 - val_accuracy: 0.6200
Epoch 61/100
230/230 [==============================] - 6s 26ms/step - loss: 1.0152 -
accuracy: 0.6910 - val_loss: 1.4332 - val_accuracy: 0.6162
Epoch 62/100
230/230 [==============================] - 4s 18ms/step - loss: 0.9939 -
accuracy: 0.7056 - val_loss: 1.4023 - val_accuracy: 0.6184
Epoch 63/100
230/230 [==============================] - 5s 20ms/step - loss: 0.9923 -
accuracy: 0.6983 - val_loss: 1.3735 - val_accuracy: 0.6320
Epoch 64/100
230/230 [==============================] - 6s 25ms/step - loss: 0.9576 -
accuracy: 0.7137 - val_loss: 1.4352 - val_accuracy: 0.6037
Epoch 65/100
```

```
230/230 [==============================] - 5s 23ms/step - loss: 0.9878 -
accuracy: 0.7065 - val_loss: 1.3652 - val_accuracy: 0.6364
Epoch 66/100
230/230 [==============================] - 4s 19ms/step - loss: 0.8919 -
accuracy: 0.7356 - val_loss: 1.3525 - val_accuracy: 0.6380
Epoch 67/100
230/230 [==============================] - 5s 22ms/step - loss: 0.8803 -
accuracy: 0.7411 - val_loss: 1.4911 - val_accuracy: 0.6015
Epoch 68/100
230/230 [==============================] - 5s 23ms/step - loss: 1.1530 -
accuracy: 0.6639 - val_loss: 1.3941 - val_accuracy: 0.6233
Epoch 69/100
230/230 [==============================] - 4s 17ms/step - loss: 0.9495 -
accuracy: 0.7213 - val_loss: 1.3490 - val_accuracy: 0.6473
Epoch 70/100
230/230 [==============================] - 5s 20ms/step - loss: 0.8977 -
accuracy: 0.7389 - val_loss: 1.3274 - val_accuracy: 0.6570
Epoch 71/100
230/230 [==============================] - 6s 24ms/step - loss: 0.8398 -
accuracy: 0.7610 - val_loss: 1.3034 - val_accuracy: 0.6658
Epoch 72/100
230/230 [==============================] - 4s 17ms/step - loss: 0.8164 -
accuracy: 0.7607 - val_loss: 1.3009 - val_accuracy: 0.6625
Epoch 73/100
230/230 [==============================] - 4s 18ms/step - loss: 0.8154 -
accuracy: 0.7616 - val_loss: 1.3166 - val_accuracy: 0.6614
Epoch 74/100
230/230 [==============================] - 6s 27ms/step - loss: 0.7957 -
accuracy: 0.7712 - val_loss: 1.2847 - val_accuracy: 0.6821
Epoch 75/100
230/230 [==============================] - 4s 19ms/step - loss: 0.8666 -
accuracy: 0.7495 - val_loss: 1.4881 - val_accuracy: 0.6200
Epoch 76/100
230/230 [==============================] - 4s 18ms/step - loss: 0.8429 -
accuracy: 0.7552 - val_loss: 1.3415 - val_accuracy: 0.6652
Epoch 77/100
230/230 [==============================] - 5s 21ms/step - loss: 0.8279 -
accuracy: 0.7642 - val_loss: 1.2907 - val_accuracy: 0.6701
Training time: 385.54614877700806 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 7ms/step - loss: 1.2908 - accuracy:
0.6769
Test Loss: 1.2908092737197876
Test Accuracy: 67.69
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 2s 9ms/step
Precision: 67.88
Recall: 67.69
F1 Score: 66.03
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[ ]:

```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support
```

0	0.87	0.85	0.86	40
1	0.90	0.95	0.93	40
2	0.97	0.97	0.97	40
3	0.79	0.65	0.71	40
4	0.77	0.68	0.72	40
5	0.46	0.68	0.55	40
6	0.80	0.90	0.85	40
7	0.62	0.45	0.52	40
8	0.71	0.85	0.77	40
9	0.90	0.93	0.91	40
10	0.50	0.05	0.09	40
11	0.45	0.75	0.57	40
12	0.64	0.17	0.27	40
13	0.74	0.80	0.77	40
14	0.37	0.55	0.44	40
15	0.74	0.80	0.77	40
16	0.64	0.62	0.63	40
17	0.85	0.82	0.84	40
18	0.66	0.62	0.64	40
19	0.80	0.90	0.85	40
20	0.74	0.57	0.65	40
21	0.70	0.75	0.72	40
22	0.40	0.62	0.49	40
23	0.96	0.68	0.79	40
24	0.66	0.88	0.75	40
25	0.53	0.68	0.59	40
26	0.64	0.75	0.69	40
27	0.75	0.68	0.71	40
28	0.70	0.75	0.72	40
29	0.60	0.80	0.69	40
30	0.82	0.93	0.87	40
31	0.89	0.80	0.84	40
32	0.88	0.88	0.88	40
33	0.71	0.90	0.79	40
34	0.78	0.72	0.75	40
35	0.61	0.50	0.55	40
36	0.78	0.62	0.69	40
37	0.75	0.23	0.35	40
38	0.58	0.75	0.65	40
39	0.57	0.78	0.66	40
40	0.44	0.97	0.61	40
41	0.51	0.65	0.57	40
42	0.91	0.75	0.82	40
43	0.46	0.45	0.46	40
44	0.92	0.88	0.90	40
45	0.72	0.72	0.73	40
46	0.64	0.75	0.69	40
47	0.61	0.75	0.67	40

```
          48       0.73      0.47      0.58        40
          49       0.73      0.28      0.40        40
          50       0.63      0.60      0.62        40
          51       0.74      0.93      0.82        40
          52       0.73      0.60      0.66        40
          53       0.85      0.70      0.77        40
          54       0.61      0.62      0.62        40
          55       0.85      0.82      0.84        40
          56       0.76      0.47      0.58        40
          57       0.97      0.85      0.91        40
          58       0.82      0.68      0.74        40
          59       0.56      0.57      0.57        40
          60       0.91      0.75      0.82        40
          61       0.65      0.65      0.65        40
          62       0.58      0.53      0.55        40
          63       0.84      0.93      0.88        40
          64       0.63      0.78      0.70        40
          65       0.63      0.65      0.64        40
          66       0.50      0.68      0.57        40
          67       0.44      0.40      0.42        40
          68       0.33      0.05      0.09        40
          69       0.00      0.00      0.00        40
          70       0.92      0.82      0.87        40
          71       0.80      0.93      0.86        40
          72       0.00      0.00      0.00        40
          73       0.82      0.82      0.82        40
          74       0.37      0.97      0.53        40
          75       0.64      0.75      0.69        40
          76       0.74      0.65      0.69        40

    accuracy                           0.68      3080
   macro avg       0.68      0.68      0.66      3080
weighted avg       0.68      0.68      0.66      3080

The number of misclassifications: 995
Proportion of misclassifications: 32.31%
Input Text: locate card
Actual Label: 11
Predicted Label: 43

Input Text: card arrived yet
Actual Label: 11
Predicted Label: 43

Input Text: get card
Actual Label: 11
Predicted Label: 43
```

```
Input Text: know tracking number new card sent
Actual Label: 11
Predicted Label: 9

Input Text: received card
Actual Label: 11
Predicted Label: 43

Input Text: still waiting card
Actual Label: 11
Predicted Label: 43

Input Text: normal wait week new card
Actual Label: 11
Predicted Label: 9

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 39

Input Text: card arrived yet
Actual Label: 11
Predicted Label: 43

Input Text: tracking info available
Actual Label: 11
Predicted Label: 71

Input Text: add card account
Actual Label: 13
Predicted Label: 39

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 39

Input Text: hello found card misplaced need reactive
Actual Label: 13
Predicted Label: 0

Input Text: view card received app
Actual Label: 13
Predicted Label: 41

Input Text: found card add app
Actual Label: 13
Predicted Label: 39
```

Input Text: link credit card
Actual Label: 13
Predicted Label: 41

Input Text: reactivate lost card found morning jacket pocket
Actual Label: 13
Predicted Label: 42

Input Text: app doesnt show card received
Actual Label: 13
Predicted Label: 16

Input Text: exchange rates offer
Actual Label: 32
Predicted Label: 31

Input Text: international exchange rates
Actual Label: 32
Predicted Label: 31

Input Text: good time exchange
Actual Label: 32
Predicted Label: 31

Input Text: currencies exchange rate calculated
Actual Label: 32
Predicted Label: 31

Input Text: rate get determined
Actual Label: 32
Predicted Label: 76

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 32

Input Text: rate exchange card payment incorrect
Actual Label: 17
Predicted Label: 76

Input Text: exchange rate card payment wrong
Actual Label: 17
Predicted Label: 76

Input Text: charged
Actual Label: 17
Predicted Label: 34

```
Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 45

Input Text: exchange rate totally wrong card payment
Actual Label: 17
Predicted Label: 76

Input Text: wrong rate applied item bought currency different mine changed
Actual Label: 17
Predicted Label: 76


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

1.6.3 LSTM (with dropout)

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
hidden_unit = 30
nlabel = 77

dropout_model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 100, weights=[embedding_matrix],␣
↪input_length=max_length_train_text, trainable=False) # Using 100 dimension␣
↪for GloVe
dropout_model.add(e)
dropout_model.add(layers.LSTM(hidden_unit, dropout=0.2))
dropout_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
dropout_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
↪metrics=['accuracy']) #, run_eagerly=True

# Summary the model
```

```
dropout_model.summary()
```

```
Model: "sequential_11"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 29, 100)           208900

 lstm_11 (LSTM)              (None, 30)                15720

 dense_11 (Dense)            (None, 77)                2387

=================================================================
Total params: 227007 (886.75 KB)
Trainable params: 18107 (70.73 KB)
Non-trainable params: 208900 (816.02 KB)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'dropout_LSTM_glove_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = dropout_model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()
```

```python
# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

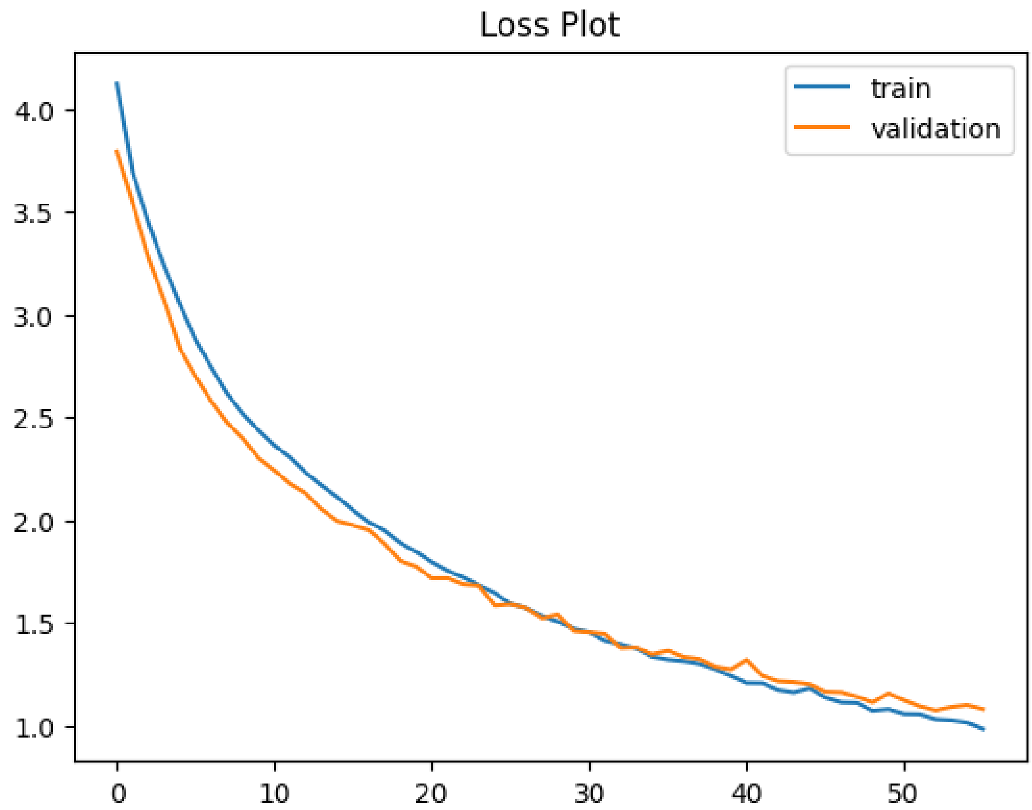
```
Epoch 1/100
230/230 [==============================] - 9s 25ms/step - loss: 4.1219 -
accuracy: 0.0373 - val_loss: 3.7914 - val_accuracy: 0.0572
Epoch 2/100
230/230 [==============================] - 4s 20ms/step - loss: 3.6903 -
accuracy: 0.0694 - val_loss: 3.5403 - val_accuracy: 0.0925
Epoch 3/100
230/230 [==============================] - 6s 28ms/step - loss: 3.4460 -
accuracy: 0.1055 - val_loss: 3.2752 - val_accuracy: 0.1285
Epoch 4/100
230/230 [==============================] - 5s 20ms/step - loss: 3.2353 -
accuracy: 0.1286 - val_loss: 3.0684 - val_accuracy: 0.1671
Epoch 5/100
230/230 [==============================] - 5s 20ms/step - loss: 3.0482 -
accuracy: 0.1516 - val_loss: 2.8333 - val_accuracy: 0.2047
Epoch 6/100
230/230 [==============================] - 7s 29ms/step - loss: 2.8758 -
accuracy: 0.1838 - val_loss: 2.6965 - val_accuracy: 0.2384
Epoch 7/100
230/230 [==============================] - 4s 19ms/step - loss: 2.7410 -
accuracy: 0.2053 - val_loss: 2.5757 - val_accuracy: 0.2515
Epoch 8/100
230/230 [==============================] - 5s 20ms/step - loss: 2.6136 -
accuracy: 0.2291 - val_loss: 2.4723 - val_accuracy: 0.2711
Epoch 9/100
230/230 [==============================] - 6s 27ms/step - loss: 2.5150 -
accuracy: 0.2584 - val_loss: 2.3961 - val_accuracy: 0.2874
Epoch 10/100
230/230 [==============================] - 5s 20ms/step - loss: 2.4349 -
accuracy: 0.2784 - val_loss: 2.2997 - val_accuracy: 0.3261
Epoch 11/100
230/230 [==============================] - 4s 19ms/step - loss: 2.3619 -
accuracy: 0.2991 - val_loss: 2.2400 - val_accuracy: 0.3381
Epoch 12/100
230/230 [==============================] - 6s 27ms/step - loss: 2.3036 -
accuracy: 0.3166 - val_loss: 2.1751 - val_accuracy: 0.3756
Epoch 13/100
230/230 [==============================] - 5s 21ms/step - loss: 2.2289 -
accuracy: 0.3357 - val_loss: 2.1284 - val_accuracy: 0.3827
Epoch 14/100
230/230 [==============================] - 5s 22ms/step - loss: 2.1670 -
accuracy: 0.3493 - val_loss: 2.0505 - val_accuracy: 0.4017
```

```
Epoch 15/100
230/230 [==============================] - 6s 27ms/step - loss: 2.1111 -
accuracy: 0.3641 - val_loss: 1.9942 - val_accuracy: 0.4023
Epoch 16/100
230/230 [==============================] - 5s 20ms/step - loss: 2.0460 -
accuracy: 0.3814 - val_loss: 1.9737 - val_accuracy: 0.4088
Epoch 17/100
230/230 [==============================] - 5s 21ms/step - loss: 1.9883 -
accuracy: 0.4048 - val_loss: 1.9501 - val_accuracy: 0.4241
Epoch 18/100
230/230 [==============================] - 6s 27ms/step - loss: 1.9483 -
accuracy: 0.4097 - val_loss: 1.8844 - val_accuracy: 0.4230
Epoch 19/100
230/230 [==============================] - 4s 19ms/step - loss: 1.8871 -
accuracy: 0.4268 - val_loss: 1.7997 - val_accuracy: 0.4600
Epoch 20/100
230/230 [==============================] - 5s 20ms/step - loss: 1.8451 -
accuracy: 0.4345 - val_loss: 1.7730 - val_accuracy: 0.4823
Epoch 21/100
230/230 [==============================] - 7s 28ms/step - loss: 1.7948 -
accuracy: 0.4594 - val_loss: 1.7160 - val_accuracy: 0.4997
Epoch 22/100
230/230 [==============================] - 5s 20ms/step - loss: 1.7520 -
accuracy: 0.4713 - val_loss: 1.7165 - val_accuracy: 0.4829
Epoch 23/100
230/230 [==============================] - 5s 21ms/step - loss: 1.7203 -
accuracy: 0.4790 - val_loss: 1.6868 - val_accuracy: 0.5101
Epoch 24/100
230/230 [==============================] - 6s 27ms/step - loss: 1.6815 -
accuracy: 0.4814 - val_loss: 1.6795 - val_accuracy: 0.5095
Epoch 25/100
230/230 [==============================] - 5s 21ms/step - loss: 1.6439 -
accuracy: 0.5052 - val_loss: 1.5830 - val_accuracy: 0.5427
Epoch 26/100
230/230 [==============================] - 5s 22ms/step - loss: 1.5916 -
accuracy: 0.5203 - val_loss: 1.5895 - val_accuracy: 0.5406
Epoch 27/100
230/230 [==============================] - 6s 27ms/step - loss: 1.5704 -
accuracy: 0.5313 - val_loss: 1.5709 - val_accuracy: 0.5455
Epoch 28/100
230/230 [==============================] - 4s 19ms/step - loss: 1.5318 -
accuracy: 0.5433 - val_loss: 1.5212 - val_accuracy: 0.5623
Epoch 29/100
230/230 [==============================] - 5s 22ms/step - loss: 1.5055 -
accuracy: 0.5506 - val_loss: 1.5391 - val_accuracy: 0.5520
Epoch 30/100
230/230 [==============================] - 6s 26ms/step - loss: 1.4706 -
accuracy: 0.5658 - val_loss: 1.4595 - val_accuracy: 0.5890
```

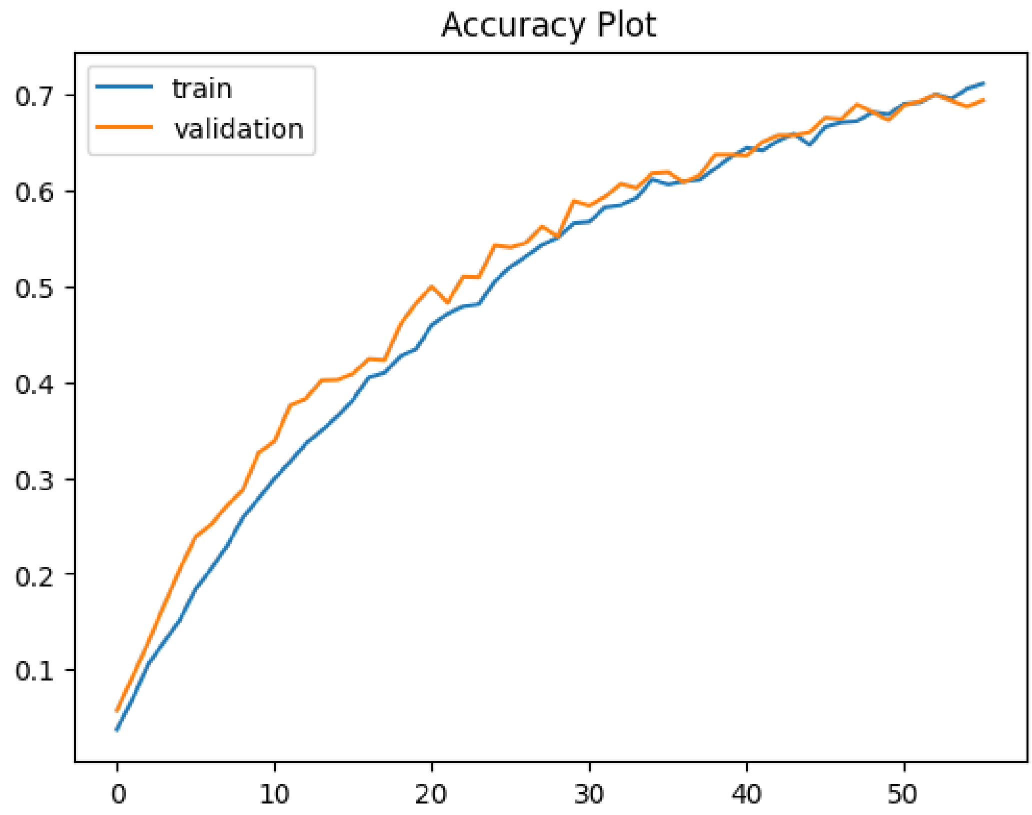
```
Epoch 31/100
230/230 [==============================] - 4s 20ms/step - loss: 1.4534 -
accuracy: 0.5672 - val_loss: 1.4532 - val_accuracy: 0.5841
Epoch 32/100
230/230 [==============================] - 5s 21ms/step - loss: 1.4122 -
accuracy: 0.5824 - val_loss: 1.4450 - val_accuracy: 0.5934
Epoch 33/100
230/230 [==============================] - 6s 27ms/step - loss: 1.3942 -
accuracy: 0.5847 - val_loss: 1.3758 - val_accuracy: 0.6070
Epoch 34/100
230/230 [==============================] - 5s 20ms/step - loss: 1.3746 -
accuracy: 0.5922 - val_loss: 1.3792 - val_accuracy: 0.6026
Epoch 35/100
230/230 [==============================] - 5s 22ms/step - loss: 1.3337 -
accuracy: 0.6116 - val_loss: 1.3454 - val_accuracy: 0.6179
Epoch 36/100
230/230 [==============================] - 6s 27ms/step - loss: 1.3192 -
accuracy: 0.6063 - val_loss: 1.3635 - val_accuracy: 0.6189
Epoch 37/100
230/230 [==============================] - 7s 29ms/step - loss: 1.3119 -
accuracy: 0.6096 - val_loss: 1.3324 - val_accuracy: 0.6086
Epoch 38/100
230/230 [==============================] - 6s 25ms/step - loss: 1.2998 -
accuracy: 0.6112 - val_loss: 1.3223 - val_accuracy: 0.6157
Epoch 39/100
230/230 [==============================] - 5s 23ms/step - loss: 1.2733 -
accuracy: 0.6231 - val_loss: 1.2850 - val_accuracy: 0.6375
Epoch 40/100
230/230 [==============================] - 5s 21ms/step - loss: 1.2412 -
accuracy: 0.6349 - val_loss: 1.2724 - val_accuracy: 0.6375
Epoch 41/100
230/230 [==============================] - 6s 26ms/step - loss: 1.2065 -
accuracy: 0.6446 - val_loss: 1.3188 - val_accuracy: 0.6364
Epoch 42/100
230/230 [==============================] - 5s 24ms/step - loss: 1.2054 -
accuracy: 0.6418 - val_loss: 1.2413 - val_accuracy: 0.6505
Epoch 43/100
230/230 [==============================] - 4s 19ms/step - loss: 1.1724 -
accuracy: 0.6518 - val_loss: 1.2153 - val_accuracy: 0.6576
Epoch 44/100
230/230 [==============================] - 7s 31ms/step - loss: 1.1606 -
accuracy: 0.6593 - val_loss: 1.2107 - val_accuracy: 0.6576
Epoch 45/100
230/230 [==============================] - 5s 20ms/step - loss: 1.1813 -
accuracy: 0.6476 - val_loss: 1.1999 - val_accuracy: 0.6609
Epoch 46/100
230/230 [==============================] - 5s 21ms/step - loss: 1.1356 -
accuracy: 0.6665 - val_loss: 1.1637 - val_accuracy: 0.6761
```

```
Epoch 47/100
230/230 [==============================] - 6s 27ms/step - loss: 1.1122 -
accuracy: 0.6711 - val_loss: 1.1611 - val_accuracy: 0.6745
Epoch 48/100
230/230 [==============================] - 5s 21ms/step - loss: 1.1101 -
accuracy: 0.6725 - val_loss: 1.1394 - val_accuracy: 0.6897
Epoch 49/100
230/230 [==============================] - 5s 21ms/step - loss: 1.0707 -
accuracy: 0.6817 - val_loss: 1.1129 - val_accuracy: 0.6821
Epoch 50/100
230/230 [==============================] - 6s 27ms/step - loss: 1.0784 -
accuracy: 0.6796 - val_loss: 1.1557 - val_accuracy: 0.6734
Epoch 51/100
230/230 [==============================] - 5s 21ms/step - loss: 1.0550 -
accuracy: 0.6904 - val_loss: 1.1224 - val_accuracy: 0.6886
Epoch 52/100
230/230 [==============================] - 4s 19ms/step - loss: 1.0537 -
accuracy: 0.6918 - val_loss: 1.0932 - val_accuracy: 0.6930
Epoch 53/100
230/230 [==============================] - 7s 30ms/step - loss: 1.0294 -
accuracy: 0.7005 - val_loss: 1.0722 - val_accuracy: 0.7001
Epoch 54/100
230/230 [==============================] - 5s 20ms/step - loss: 1.0248 -
accuracy: 0.6956 - val_loss: 1.0894 - val_accuracy: 0.6935
Epoch 55/100
230/230 [==============================] - 5s 21ms/step - loss: 1.0138 -
accuracy: 0.7062 - val_loss: 1.0988 - val_accuracy: 0.6875
Epoch 56/100
230/230 [==============================] - 7s 29ms/step - loss: 0.9829 -
accuracy: 0.7115 - val_loss: 1.0786 - val_accuracy: 0.6941
Training time: 301.18429708480835 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 6ms/step - loss: 1.0427 - accuracy:
0.7166
Test Loss: 1.042669415473938
Test Accuracy: 71.66
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
```

```
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 1s 7ms/step
Precision: 72.08
Recall: 71.66
F1 Score: 70.69
```

[ ]:

```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91        40
           1       0.77      0.85      0.81        40
           2       0.86      0.95      0.90        40
           3       0.82      0.70      0.76        40
           4       0.66      0.68      0.67        40
           5       0.59      0.75      0.66        40
```

6	0.85	0.85	0.85	40
7	0.79	0.47	0.59	40
8	0.77	0.82	0.80	40
9	1.00	0.93	0.96	40
10	0.00	0.00	0.00	40
11	0.41	0.82	0.55	40
12	0.58	0.35	0.44	40
13	0.80	0.88	0.83	40
14	0.66	0.68	0.67	40
15	0.74	0.78	0.76	40
16	0.52	0.62	0.57	40
17	0.94	0.82	0.88	40
18	0.86	0.78	0.82	40
19	0.88	0.90	0.89	40
20	0.61	0.75	0.67	40
21	0.80	0.80	0.80	40
22	0.61	0.50	0.55	40
23	1.00	0.88	0.93	40
24	0.79	0.95	0.86	40
25	0.54	0.68	0.60	40
26	0.65	0.85	0.74	40
27	0.93	0.68	0.78	40
28	0.83	0.72	0.77	40
29	0.76	0.70	0.73	40
30	0.83	0.97	0.90	40
31	0.81	0.85	0.83	40
32	0.85	0.97	0.91	40
33	0.84	0.80	0.82	40
34	0.70	0.82	0.76	40
35	0.59	0.68	0.63	40
36	0.77	0.82	0.80	40
37	0.49	0.68	0.57	40
38	0.86	0.90	0.88	40
39	0.47	0.60	0.53	40
40	0.65	0.93	0.76	40
41	0.63	0.68	0.65	40
42	0.91	0.80	0.85	40
43	0.33	0.38	0.35	40
44	0.84	0.90	0.87	40
45	0.94	0.72	0.82	40
46	0.88	0.75	0.81	40
47	0.70	0.65	0.68	40
48	0.64	0.53	0.58	40
49	0.84	0.65	0.73	40
50	0.74	0.70	0.72	40
51	0.69	0.88	0.77	40
52	0.70	0.47	0.57	40
53	0.54	0.72	0.62	40

```
          54       0.60      0.60      0.60        40
          55       0.81      0.88      0.84        40
          56       0.94      0.40      0.56        40
          57       0.85      0.88      0.86        40
          58       0.77      0.75      0.76        40
          59       0.57      0.65      0.60        40
          60       0.71      0.80      0.75        40
          61       0.57      0.72      0.64        40
          62       0.75      0.68      0.71        40
          63       0.97      0.78      0.86        40
          64       0.73      0.88      0.80        40
          65       0.61      0.70      0.65        40
          66       0.56      0.35      0.43        40
          67       0.61      0.82      0.70        40
          68       0.79      0.55      0.65        40
          69       0.36      0.23      0.28        40
          70       0.72      0.97      0.83        40
          71       0.83      0.95      0.88        40
          72       0.50      0.03      0.05        40
          73       0.97      0.85      0.91        40
          74       0.43      0.57      0.49        40
          75       0.78      0.53      0.63        40
          76       0.91      0.75      0.82        40

    accuracy                           0.72      3080
   macro avg       0.72      0.72      0.71      3080
weighted avg       0.72      0.72      0.71      3080

The number of misclassifications: 873
Proportion of misclassifications: 28.34%
Input Text: locate card
Actual Label: 11
Predicted Label: 0

Input Text: waiting longer expected bank card could provide information arrive
Actual Label: 11
Predicted Label: 12

Input Text: ive waiting longer expected card
Actual Label: 11
Predicted Label: 12

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 41

Input Text: status card ordered
Actual Label: 11
```

```
Predicted Label: 14

Input Text: long new card take arrive
Actual Label: 11
Predicted Label: 12

Input Text: tracking info available
Actual Label: 11
Predicted Label: 30

Input Text: add card account
Actual Label: 13
Predicted Label: 39

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 41

Input Text: hello found card misplaced need reactive
Actual Label: 13
Predicted Label: 42

Input Text: view card received app
Actual Label: 13
Predicted Label: 41

Input Text: app doesnt show card received
Actual Label: 13
Predicted Label: 41

Input Text: good time exchange
Actual Label: 32
Predicted Label: 31

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: rate low sure using right exchange rate
Actual Label: 17
Predicted Label: 32

Input Text: charged
Actual Label: 17
Predicted Label: 34

Input Text: charged wrong currency exchange purchase
Actual Label: 17
```

```
Predicted Label: 31

Input Text: exchange rate seems transaction
Actual Label: 17
Predicted Label: 32

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 16

Input Text: check exchange rate applied transaction
Actual Label: 17
Predicted Label: 32

Input Text: would like refund extra pound charged
Actual Label: 34
Predicted Label: 19

Input Text: transaction credited
Actual Label: 34
Predicted Label: 66

Input Text: fee come
Actual Label: 34
Predicted Label: 15

Input Text: extra pound charge card
Actual Label: 34
Predicted Label: 57

Input Text: euro fee come
Actual Label: 34
Predicted Label: 50

Input Text: euro fee statement
Actual Label: 34
Predicted Label: 31

Input Text: two weeks transaction reversed
Actual Label: 34
Predicted Label: 8

Input Text: withdrawl still pending
Actual Label: 46
Predicted Label: 66

Input Text: hi wondering help used city centre atm get cash machine declined
card account shows transaction still pending didnt receive money please cancel
```

```
transaction
Actual Label: 46
Predicted Label: 76

Input Text: long til cash goes
Actual Label: 46
Predicted Label: 26
```

1.6.4 Hyperparameter tuning

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Define the model for hyperparameter tuning
def model_builder(hp):
  model = keras.models.Sequential()
  e = layers.Embedding(voca_size+1, 100, weights=[embedding_matrix],␣
↪input_length=max_length_train_text, trainable=False)
  model.add(e)
  hp_units = hp.Int('units', min_value = 20, max_value = 50, step = 10) # Set␣
↪up the hyperparameters
  model.add(layers.LSTM(units = hp_units)) # We will check the optimal hidden␣
↪unit for the LSTM layer
  model.add(layers.Dense(nlabel, activation='softmax'))

  hp_learning_rate = hp.Choice('learning_rate', values = [0.01, 0.001, 0.0001])␣
↪# Set up the hyperparameters
  model.compile(optimizer = keras.optimizers.Adam(learning_rate =␣
↪hp_learning_rate), # We will check the optimal learning rate
                loss = 'sparse_categorical_crossentropy',
                metrics = ['accuracy'])
  return model
```

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Specify the tuner
tuner = kt.Hyperband(model_builder,
                     objective = 'val_accuracy',
                     max_epochs = 100)
```

[ ]:
```
# The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Set up a callback for early stopping
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
```

```
[ ]: # The code for hyperparameter tuning is derived from the Tensorflow website.
# (https://www.tensorflow.org/tutorials/keras/keras_tuner)

# Run the tuner
tuner.search(X_train_padded, y_train, epochs = 100, validation_data =␣
 ↪(X_val_padded, y_val), callbacks = [stop_early])

# Get the optimal hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials = 1)[0]

print(f"The optimal number of units: {best_hps.get('units')}. The optimal␣
 ↪learning rate: {best_hps.get('learning_rate')}.")
```

```
Trial 12 Complete [00h 00m 13s]
val_accuracy: 0.02994011901319027

Best val_accuracy So Far: 0.2449646145105362
Total elapsed time: 00h 09m 37s
The optimal number of units: 50. The optimal learning rate: 0.01.
```

1.6.5 Tuned LSTM

```
[ ]: # Define the output dimension for the embedding layer and hidden units
nlabel = 77

tuned_model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 100, weights=[embedding_matrix],␣
 ↪input_length=max_length_train_text, trainable=False) # Using 100 dimension␣
 ↪for GloVe
tuned_model.add(e)
tuned_model.add(layers.LSTM(50))
tuned_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
tuned_model.compile(optimizer=keras.optimizers.Adam(learning_rate = 0.01),␣
 ↪loss='sparse_categorical_crossentropy', metrics=['accuracy']) #,␣
 ↪run_eagerly=True

# Summary the model
tuned_model.summary()
```

```
Model: "sequential_12"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_12 (Embedding)    (None, 29, 100)           208900
```

```
 lstm_12 (LSTM)              (None, 50)               30200

 dense_12 (Dense)            (None, 77)               3927

=================================================================
Total params: 243027 (949.32 KB)
Trainable params: 34127 (133.31 KB)
Non-trainable params: 208900 (816.02 KB)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'tuned_LSTM_glove_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

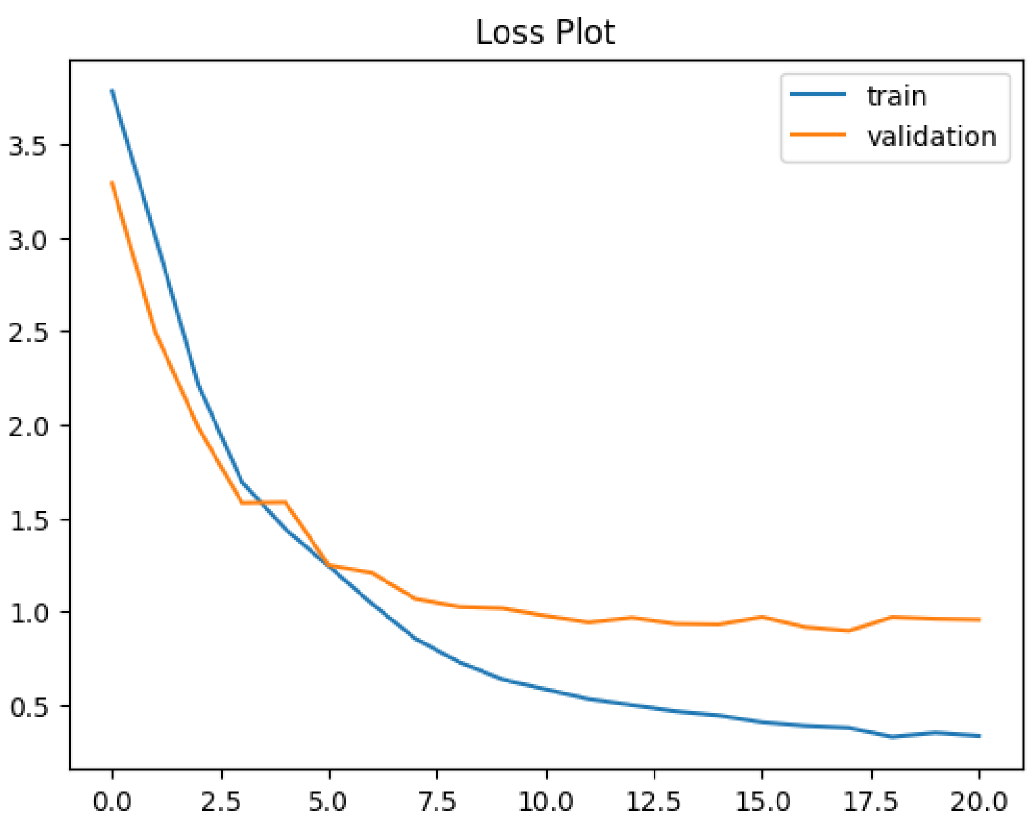
[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = tuned_model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 10s 34ms/step - loss: 3.7835 -
accuracy: 0.0475 - val_loss: 3.2911 - val_accuracy: 0.1083
Epoch 2/100
```

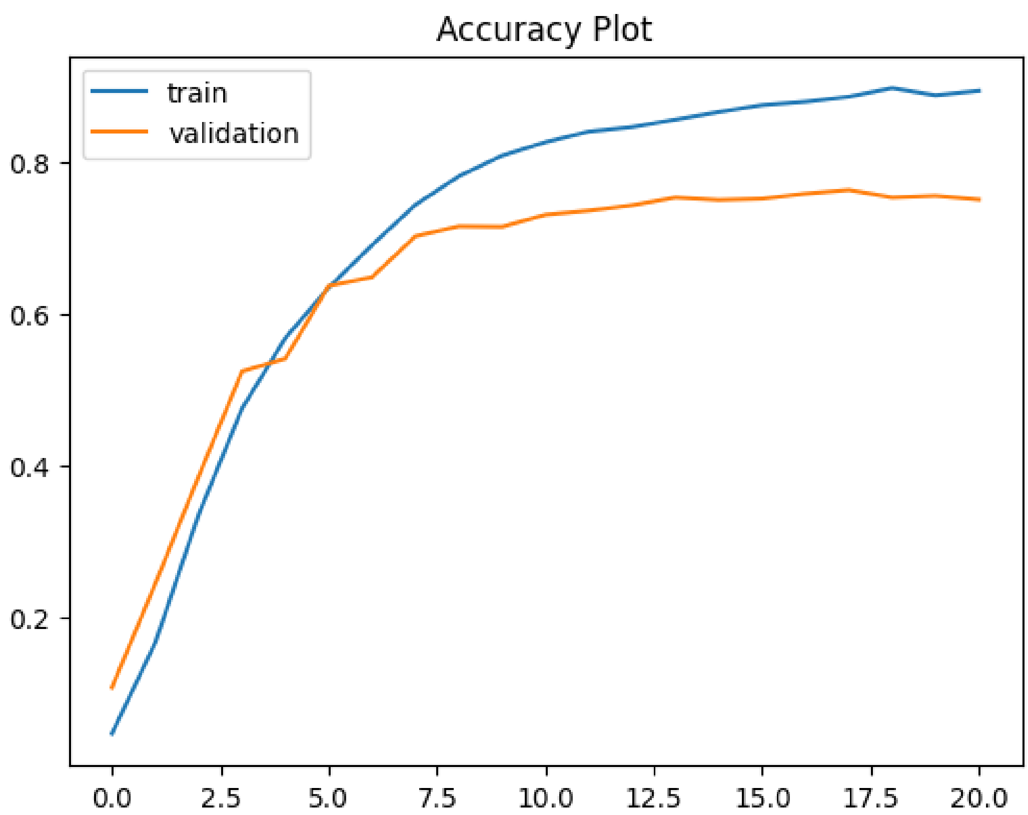
```
230/230 [==============================] - 5s 22ms/step - loss: 3.0044 -
accuracy: 0.1673 - val_loss: 2.4936 - val_accuracy: 0.2450
Epoch 3/100
230/230 [==============================] - 5s 21ms/step - loss: 2.2075 -
accuracy: 0.3361 - val_loss: 1.9810 - val_accuracy: 0.3865
Epoch 4/100
230/230 [==============================] - 8s 33ms/step - loss: 1.6925 -
accuracy: 0.4758 - val_loss: 1.5815 - val_accuracy: 0.5248
Epoch 5/100
230/230 [==============================] - 5s 20ms/step - loss: 1.4416 -
accuracy: 0.5687 - val_loss: 1.5846 - val_accuracy: 0.5416
Epoch 6/100
230/230 [==============================] - 5s 21ms/step - loss: 1.2428 -
accuracy: 0.6352 - val_loss: 1.2479 - val_accuracy: 0.6375
Epoch 7/100
230/230 [==============================] - 7s 30ms/step - loss: 1.0417 -
accuracy: 0.6911 - val_loss: 1.2065 - val_accuracy: 0.6489
Epoch 8/100
230/230 [==============================] - 5s 21ms/step - loss: 0.8548 -
accuracy: 0.7442 - val_loss: 1.0684 - val_accuracy: 0.7028
Epoch 9/100
230/230 [==============================] - 5s 24ms/step - loss: 0.7309 -
accuracy: 0.7818 - val_loss: 1.0257 - val_accuracy: 0.7158
Epoch 10/100
230/230 [==============================] - 6s 27ms/step - loss: 0.6378 -
accuracy: 0.8091 - val_loss: 1.0178 - val_accuracy: 0.7153
Epoch 11/100
230/230 [==============================] - 5s 20ms/step - loss: 0.5842 -
accuracy: 0.8267 - val_loss: 0.9773 - val_accuracy: 0.7311
Epoch 12/100
230/230 [==============================] - 6s 27ms/step - loss: 0.5322 -
accuracy: 0.8407 - val_loss: 0.9424 - val_accuracy: 0.7365
Epoch 13/100
230/230 [==============================] - 6s 25ms/step - loss: 0.4995 -
accuracy: 0.8467 - val_loss: 0.9674 - val_accuracy: 0.7436
Epoch 14/100
230/230 [==============================] - 6s 28ms/step - loss: 0.4665 -
accuracy: 0.8565 - val_loss: 0.9347 - val_accuracy: 0.7539
Epoch 15/100
230/230 [==============================] - 6s 27ms/step - loss: 0.4443 -
accuracy: 0.8665 - val_loss: 0.9314 - val_accuracy: 0.7507
Epoch 16/100
230/230 [==============================] - 5s 21ms/step - loss: 0.4079 -
accuracy: 0.8756 - val_loss: 0.9710 - val_accuracy: 0.7523
Epoch 17/100
230/230 [==============================] - 6s 26ms/step - loss: 0.3888 -
accuracy: 0.8801 - val_loss: 0.9167 - val_accuracy: 0.7588
Epoch 18/100
```

```
230/230 [==============================] - 6s 24ms/step - loss: 0.3782 -
accuracy: 0.8866 - val_loss: 0.8966 - val_accuracy: 0.7637
Epoch 19/100
230/230 [==============================] - 5s 22ms/step - loss: 0.3296 -
accuracy: 0.8980 - val_loss: 0.9706 - val_accuracy: 0.7539
Epoch 20/100
230/230 [==============================] - 6s 28ms/step - loss: 0.3516 -
accuracy: 0.8885 - val_loss: 0.9608 - val_accuracy: 0.7561
Epoch 21/100
230/230 [==============================] - 5s 23ms/step - loss: 0.3347 -
accuracy: 0.8945 - val_loss: 0.9567 - val_accuracy: 0.7512
Training time: 123.43249177932739 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```python
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```python
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 7ms/step - loss: 0.8815 - accuracy:
0.7769
Test Loss: 0.8814808130264282
Test Accuracy: 77.69
```

[ ]:
```python
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 1s 7ms/step
Precision: 79.27
Recall: 77.69
F1 Score: 77.84
```

[ ]:
```python
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support
```

0	1.00	0.93	0.96	40
1	0.92	0.88	0.90	40
2	0.95	0.90	0.92	40
3	0.95	0.88	0.91	40
4	0.91	0.78	0.84	40
5	0.56	0.68	0.61	40
6	0.84	0.80	0.82	40
7	0.63	0.55	0.59	40
8	0.85	0.88	0.86	40
9	0.95	0.97	0.96	40
10	0.76	0.70	0.73	40
11	0.54	0.88	0.67	40
12	0.69	0.50	0.58	40
13	0.81	0.95	0.87	40
14	0.63	0.78	0.70	40
15	0.79	0.82	0.80	40
16	0.55	0.68	0.61	40
17	0.84	0.93	0.88	40
18	1.00	0.65	0.79	40
19	0.83	0.85	0.84	40
20	0.71	0.72	0.72	40
21	0.97	0.80	0.88	40
22	0.86	0.60	0.71	40
23	0.95	0.88	0.91	40
24	0.90	0.90	0.90	40
25	0.66	0.82	0.73	40
26	0.68	0.70	0.69	40
27	0.93	0.68	0.78	40
28	0.71	0.75	0.73	40
29	0.93	0.65	0.76	40
30	1.00	0.95	0.97	40
31	0.91	0.80	0.85	40
32	0.93	0.95	0.94	40
33	0.71	0.93	0.80	40
34	0.77	0.82	0.80	40
35	0.57	0.75	0.65	40
36	0.89	0.82	0.86	40
37	0.65	0.75	0.70	40
38	0.81	0.97	0.89	40
39	0.70	0.70	0.70	40
40	0.79	0.95	0.86	40
41	0.87	0.65	0.74	40
42	0.97	0.85	0.91	40
43	0.53	0.65	0.58	40
44	0.93	0.95	0.94	40
45	0.85	0.70	0.77	40
46	0.84	0.78	0.81	40

47	0.63	0.80	0.70	40
48	0.73	0.55	0.63	40
49	0.94	0.80	0.86	40
50	0.77	0.68	0.72	40
51	0.74	0.88	0.80	40
52	0.84	0.78	0.81	40
53	0.79	0.78	0.78	40
54	0.71	0.60	0.65	40
55	0.92	0.88	0.90	40
56	0.83	0.62	0.71	40
57	0.89	0.82	0.86	40
58	0.96	0.65	0.78	40
59	0.79	0.78	0.78	40
60	0.79	0.82	0.80	40
61	0.84	0.68	0.75	40
62	0.70	0.75	0.72	40
63	0.81	0.85	0.83	40
64	0.72	0.85	0.78	40
65	0.67	0.65	0.66	40
66	0.55	0.72	0.62	40
67	0.69	0.62	0.66	40
68	0.75	0.75	0.75	40
69	0.56	0.38	0.45	40
70	0.89	0.97	0.93	40
71	0.93	0.93	0.93	40
72	0.85	0.72	0.78	40
73	0.92	0.90	0.91	40
74	0.48	0.72	0.57	40
75	0.55	0.75	0.63	40
76	0.78	0.72	0.75	40
accuracy			0.78	3080
macro avg	0.79	0.78	0.78	3080
weighted avg	0.79	0.78	0.78	3080

```
The number of misclassifications: 687
Proportion of misclassifications: 22.31%
Input Text: locate card
Actual Label: 11
Predicted Label: 13

Input Text: way know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: get card
Actual Label: 11
Predicted Label: 43
```

```
Input Text: long card delivery take
Actual Label: 11
Predicted Label: 12

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 41

Input Text: add card account
Actual Label: 13
Predicted Label: 39

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 11

Input Text: good time exchange
Actual Label: 32
Predicted Label: 33

Input Text: currencies exchange rate calculated
Actual Label: 32
Predicted Label: 17

Input Text: charged
Actual Label: 17
Predicted Label: 34

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 16

Input Text: paid something foreign currency noticed exchange rate incorrect
Actual Label: 17
Predicted Label: 76

Input Text: explain random charge
Actual Label: 34
Predicted Label: 72

Input Text: transaction credited
Actual Label: 34
Predicted Label: 47

Input Text: fee come
Actual Label: 34
Predicted Label: 15
```

```
Input Text: extra charge
Actual Label: 34
Predicted Label: 64

Input Text: euro fee come
Actual Label: 34
Predicted Label: 31

Input Text: new customer happened look app charge familiar could tell extra
charge
Actual Label: 34
Predicted Label: 28

Input Text: two weeks transaction reversed
Actual Label: 34
Predicted Label: 27

Input Text: hey tried get money earlier machine didnt work saw transaction still
seems progress please check whats going seems something broken dont want charged
money havent actually received
Actual Label: 46
Predicted Label: 63

Input Text: long til cash goes
Actual Label: 46
Predicted Label: 5

Input Text: hii tried get money machine working transaction still seems progress
please check whats going oni dont want charged money received
Actual Label: 46
Predicted Label: 63

Input Text: made withdrawal account posted
Actual Label: 46
Predicted Label: 6

Input Text: tried take money card didnt work later saw transaction still
progress whats goign
Actual Label: 46
Predicted Label: 25

Input Text: wheres accounting cash withdrawal
Actual Label: 46
Predicted Label: 76

Input Text: long take post atm drawl
Actual Label: 46
```

```
Predicted Label: 4

Input Text: cash withdrawal atm still yet showing confirmed account
Actual Label: 46
Predicted Label: 20

Input Text: whats pending transaction card declined atm account says still
pending cancel payment
Actual Label: 46
Predicted Label: 51

Input Text: incoming payment account deactivated still processed
Actual Label: 36
Predicted Label: 45

Input Text: currencies exchanges
Actual Label: 36
Predicted Label: 32
```

1.6.6 Tuned LSTM (with dropout)

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
nlabel = 77

dropout_tuned_model = keras.models.Sequential()
e = layers.Embedding(voca_size+1, 100, weights=[embedding_matrix],␣
 ↪input_length=max_length_train_text, trainable=False) # Using 100 dimension␣
 ↪for GloVe
dropout_tuned_model.add(e)
dropout_tuned_model.add(layers.LSTM(50, dropout=0.2))
dropout_tuned_model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
dropout_tuned_model.compile(optimizer=keras.optimizers.Adam(learning_rate = 0.
 ↪01), loss='sparse_categorical_crossentropy', metrics=['accuracy']) #,␣
 ↪run_eagerly=True

# Summary the model
dropout_tuned_model.summary()
```

```
Model: "sequential_13"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_13 (Embedding)    (None, 29, 100)           208900
```

```
 lstm_13 (LSTM)              (None, 50)                30200

 dense_13 (Dense)            (None, 77)                3927

=================================================================
Total params: 243027 (949.32 KB)
Trainable params: 34127 (133.31 KB)
Non-trainable params: 208900 (816.02 KB)
_________________________________________________________________
```

[ ]:
```python
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/LSTM/'

# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'dropout_tuned_LSTM_glove_model.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

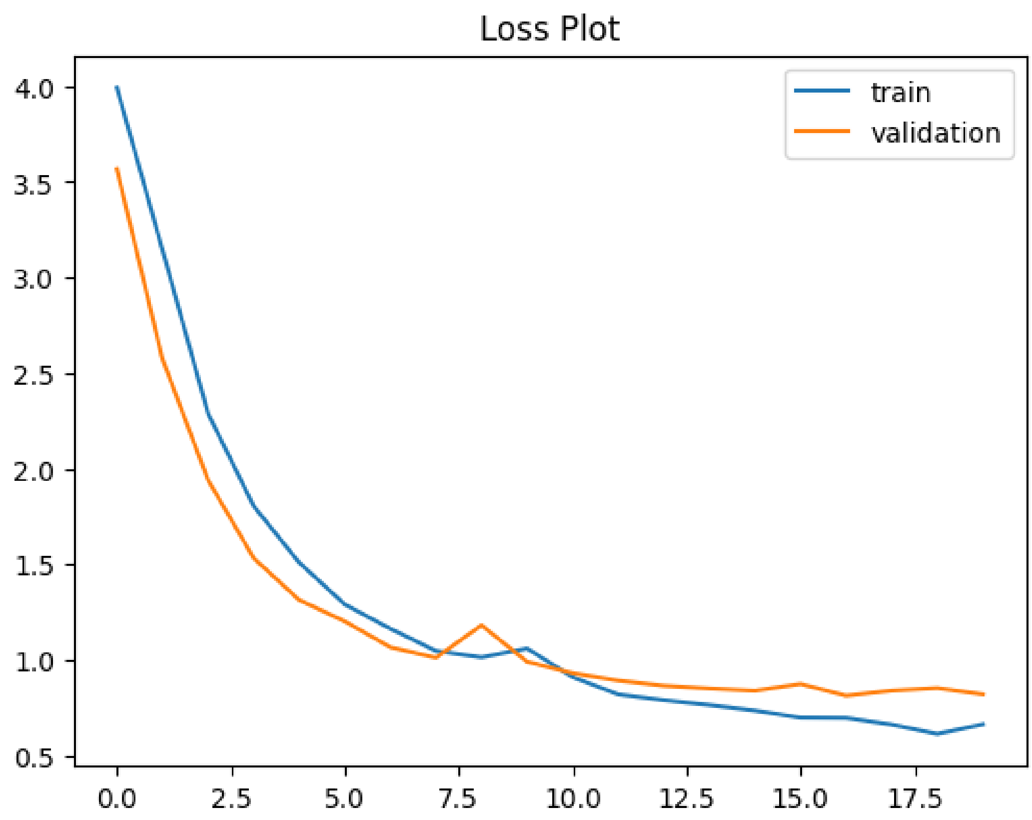
[ ]:
```python
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = dropout_tuned_model.fit(
    X_train_padded, y_train,
    epochs = 100,
    validation_data = (X_val_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 11s 32ms/step - loss: 3.9919 -
accuracy: 0.0328 - val_loss: 3.5665 - val_accuracy: 0.0675
Epoch 2/100
```

```
230/230 [==============================] - 5s 23ms/step - loss: 3.1440 -
accuracy: 0.1525 - val_loss: 2.5732 - val_accuracy: 0.2580
Epoch 3/100
230/230 [==============================] - 7s 32ms/step - loss: 2.2885 -
accuracy: 0.3294 - val_loss: 1.9406 - val_accuracy: 0.4066
Epoch 4/100
230/230 [==============================] - 5s 23ms/step - loss: 1.8043 -
accuracy: 0.4413 - val_loss: 1.5322 - val_accuracy: 0.5291
Epoch 5/100
230/230 [==============================] - 6s 25ms/step - loss: 1.5087 -
accuracy: 0.5449 - val_loss: 1.3131 - val_accuracy: 0.6091
Epoch 6/100
230/230 [==============================] - 6s 26ms/step - loss: 1.2901 -
accuracy: 0.6024 - val_loss: 1.2017 - val_accuracy: 0.6358
Epoch 7/100
230/230 [==============================] - 5s 22ms/step - loss: 1.1630 -
accuracy: 0.6401 - val_loss: 1.0661 - val_accuracy: 0.6924
Epoch 8/100
230/230 [==============================] - 7s 30ms/step - loss: 1.0470 -
accuracy: 0.6771 - val_loss: 1.0115 - val_accuracy: 0.6962
Epoch 9/100
230/230 [==============================] - 5s 23ms/step - loss: 1.0137 -
accuracy: 0.6917 - val_loss: 1.1810 - val_accuracy: 0.6538
Epoch 10/100
230/230 [==============================] - 5s 22ms/step - loss: 1.0596 -
accuracy: 0.6762 - val_loss: 0.9902 - val_accuracy: 0.7039
Epoch 11/100
230/230 [==============================] - 7s 32ms/step - loss: 0.9101 -
accuracy: 0.7207 - val_loss: 0.9309 - val_accuracy: 0.7267
Epoch 12/100
230/230 [==============================] - 5s 23ms/step - loss: 0.8195 -
accuracy: 0.7437 - val_loss: 0.8921 - val_accuracy: 0.7414
Epoch 13/100
230/230 [==============================] - 6s 28ms/step - loss: 0.7892 -
accuracy: 0.7522 - val_loss: 0.8647 - val_accuracy: 0.7420
Epoch 14/100
230/230 [==============================] - 6s 27ms/step - loss: 0.7638 -
accuracy: 0.7601 - val_loss: 0.8502 - val_accuracy: 0.7583
Epoch 15/100
230/230 [==============================] - 5s 21ms/step - loss: 0.7349 -
accuracy: 0.7682 - val_loss: 0.8390 - val_accuracy: 0.7605
Epoch 16/100
230/230 [==============================] - 7s 32ms/step - loss: 0.6979 -
accuracy: 0.7812 - val_loss: 0.8738 - val_accuracy: 0.7572
Epoch 17/100
230/230 [==============================] - 5s 23ms/step - loss: 0.6964 -
accuracy: 0.7830 - val_loss: 0.8131 - val_accuracy: 0.7681
Epoch 18/100
```

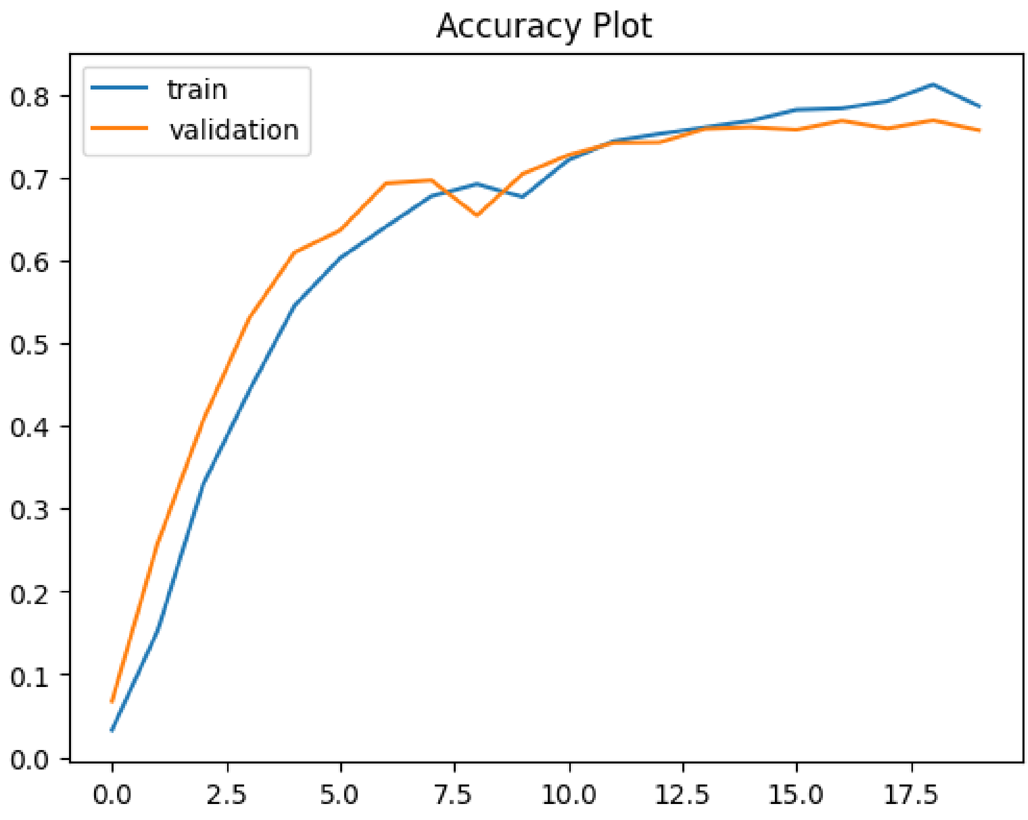
```
230/230 [==============================] - 6s 25ms/step - loss: 0.6616 -
accuracy: 0.7919 - val_loss: 0.8395 - val_accuracy: 0.7588
Epoch 19/100
230/230 [==============================] - 7s 31ms/step - loss: 0.6132 -
accuracy: 0.8117 - val_loss: 0.8524 - val_accuracy: 0.7686
Epoch 20/100
230/230 [==============================] - 5s 22ms/step - loss: 0.6627 -
accuracy: 0.7859 - val_loss: 0.8202 - val_accuracy: 0.7567
Training time: 123.8074312210083 seconds
```

[ ]:
```
# Plot the loss
plt.title('Loss Plot')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='validation')
plt.legend()
plt.show()
```



[ ]:
```
# Plot the accuracy
plt.title('Accuracy Plot')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='validation')
```

```
plt.legend()
plt.show()
```



[ ]:
```
# Load the saved model
saved_model = tf.keras.models.load_model(model_checkpoint_path)

# Evaluate the model with the test set
loss, accuracy = saved_model.evaluate(X_test_padded, y_test_array)

print("Test Loss:", loss)
print("Test Accuracy:", round((accuracy*100), 2))
```

```
97/97 [==============================] - 2s 8ms/step - loss: 0.7693 - accuracy:
0.7786
Test Loss: 0.7692508697509766
Test Accuracy: 77.86
```

[ ]:
```
# Check predictions with the test set
y_test_prob = saved_model.predict(X_test_padded)

# Convert probabilities to class labels
```

```
y_test_pred = np.argmax(y_test_prob, axis=1)

# Calculate precision, recall, and f1 score
precision = precision_score(y_test_array, y_test_pred, average='weighted')
recall = recall_score(y_test_array, y_test_pred, average='weighted')
f1 = f1_score(y_test_array, y_test_pred, average='weighted')

print("Precision:", round((precision*100), 2))
print("Recall:", round((recall*100), 2))
print("F1 Score:", round((f1*100), 2))
```

```
97/97 [==============================] - 2s 11ms/step
Precision: 79.88
Recall: 77.86
F1 Score: 77.85
```

[ ]:
```
# Error analysis

# Print classification report
print(classification_report(y_test_array, y_test_pred))

# Check misclassified data
misclassified_data = np.where(y_test_pred != y_test_array)[0]
print(f"The number of misclassifications: {len(misclassified_data)}")

# Check the ratio of misclassifications
misclassification_ratio = (len(misclassified_data) / len(y_test_array)) * 100
# Round the number
rounded_ratio = round(misclassification_ratio, 2)
print(f"Proportion of misclassifications: {rounded_ratio}%")

# Iterate over misclassified data for error analysis
for idx in misclassified_data[:30]:
    input_text = X_test[idx]
    true_label = y_test[idx]
    predicted_label = y_test_pred[idx]

    # Print information about the misclassified data
    print("Input Text:", input_text)
    print("Actual Label:", true_label)
    print("Predicted Label:", predicted_label)
    print()
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        40
           1       0.89      0.97      0.93        40
           2       0.93      0.97      0.95        40
```

3	0.76	0.95	0.84	40
4	0.91	0.78	0.84	40
5	0.48	0.75	0.59	40
6	0.83	0.85	0.84	40
7	0.76	0.70	0.73	40
8	0.88	0.88	0.88	40
9	1.00	0.90	0.95	40
10	0.77	0.82	0.80	40
11	0.79	0.68	0.73	40
12	0.62	0.88	0.73	40
13	0.90	0.88	0.89	40
14	0.76	0.85	0.80	40
15	0.88	0.88	0.88	40
16	0.72	0.65	0.68	40
17	0.77	0.93	0.84	40
18	0.97	0.78	0.86	40
19	0.88	0.88	0.88	40
20	0.88	0.75	0.81	40
21	0.82	0.35	0.49	40
22	0.59	0.75	0.66	40
23	0.89	0.85	0.87	40
24	0.89	0.97	0.93	40
25	0.70	0.80	0.74	40
26	0.69	0.85	0.76	40
27	0.88	0.75	0.81	40
28	0.80	0.70	0.75	40
29	0.89	0.62	0.74	40
30	0.97	0.85	0.91	40
31	0.92	0.88	0.90	40
32	0.92	0.88	0.90	40
33	0.65	0.85	0.74	40
34	0.72	0.78	0.75	40
35	0.76	0.65	0.70	40
36	0.66	0.78	0.71	40
37	0.66	0.68	0.67	40
38	0.40	0.95	0.56	40
39	0.76	0.85	0.80	40
40	0.88	0.93	0.90	40
41	0.68	0.68	0.68	40
42	0.93	0.93	0.93	40
43	0.69	0.68	0.68	40
44	0.81	0.65	0.72	40
45	0.79	0.78	0.78	40
46	0.71	0.80	0.75	40
47	0.71	0.60	0.65	40
48	0.65	0.60	0.62	40
49	0.94	0.72	0.82	40
50	0.82	0.68	0.74	40

```
          51       0.79      0.82      0.80        40
          52       0.81      0.75      0.78        40
          53       0.72      0.78      0.75        40
          54       0.71      0.60      0.65        40
          55       0.90      0.93      0.91        40
          56       0.95      0.53      0.68        40
          57       0.93      0.93      0.93        40
          58       0.93      0.70      0.80        40
          59       0.83      0.72      0.77        40
          60       0.95      0.90      0.92        40
          61       0.60      0.70      0.64        40
          62       0.72      0.72      0.73        40
          63       0.70      0.88      0.78        40
          64       0.82      0.82      0.82        40
          65       0.68      0.75      0.71        40
          66       0.80      0.70      0.75        40
          67       0.70      0.70      0.70        40
          68       0.93      0.35      0.51        40
          69       0.49      0.75      0.59        40
          70       0.89      1.00      0.94        40
          71       0.91      1.00      0.95        40
          72       0.79      0.82      0.80        40
          73       0.97      0.88      0.92        40
          74       0.65      0.38      0.48        40
          75       0.89      0.78      0.83        40
          76       0.93      0.62      0.75        40

    accuracy                           0.78      3080
   macro avg       0.80      0.78      0.78      3080
weighted avg       0.80      0.78      0.78      3080

The number of misclassifications: 682
Proportion of misclassifications: 22.14%
Input Text: locate card
Actual Label: 11
Predicted Label: 41

Input Text: ordered card arrived help please
Actual Label: 11
Predicted Label: 12

Input Text: way know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: get card
Actual Label: 11
Predicted Label: 43
```

```
Input Text: received card
Actual Label: 11
Predicted Label: 43

Input Text: track card
Actual Label: 11
Predicted Label: 12

Input Text: long card delivery take
Actual Label: 11
Predicted Label: 12

Input Text: waiting longer expected bank card could provide information arrive
Actual Label: 11
Predicted Label: 12

Input Text: get card yet lost
Actual Label: 11
Predicted Label: 41

Input Text: status card ordered
Actual Label: 11
Predicted Label: 22

Input Text: long new card take arrive
Actual Label: 11
Predicted Label: 12

Input Text: know card arrive
Actual Label: 11
Predicted Label: 12

Input Text: tracking info available
Actual Label: 11
Predicted Label: 74

Input Text: received new card dont see app anywhere
Actual Label: 13
Predicted Label: 12

Input Text: add card account
Actual Label: 13
Predicted Label: 39

Input Text: put old card back system found
Actual Label: 13
Predicted Label: 41
```

```
Input Text: way make old card usable app
Actual Label: 13
Predicted Label: 11

Input Text: found lost stolen card way link card account app
Actual Label: 13
Predicted Label: 42

Input Text: good time exchange
Actual Label: 32
Predicted Label: 33

Input Text: much get exchange rate
Actual Label: 32
Predicted Label: 17

Input Text: im trying figure current exchange rate
Actual Label: 32
Predicted Label: 17

Input Text: kind foreign exchange rate get exchange money
Actual Label: 32
Predicted Label: 17

Input Text: rate get determined
Actual Label: 32
Predicted Label: 17

Input Text: made currency exchange think charged
Actual Label: 17
Predicted Label: 31

Input Text: charged
Actual Label: 17
Predicted Label: 63

Input Text: conversion value card payments incorrect
Actual Label: 17
Predicted Label: 15

Input Text: explain random charge
Actual Label: 34
Predicted Label: 63

Input Text: remember purchasing anything £ statement please tell
Actual Label: 34
Predicted Label: 53
```

```
Input Text: transaction credited
Actual Label: 34
Predicted Label: 62

Input Text: many fees statement
Actual Label: 34
Predicted Label: 57
```

1.7 Tokenize the text in the dataset

1.7.1 For DistilBERT Model

ver. 1) Use the preprocessed dataset

[ ]:
```
# Change the format as dataframe
banking77_preprocessed.reset_format()
```

[ ]:
```
# Perform train-test split to make a validation set
# Export only data set to split (training 80%, validation 20% from the training␣
↪set)
dataset_dict = banking77_preprocessed['train'].train_test_split(test_size=0.2)
```

[ ]:
```
# Change the name 'test' to 'validation'
dataset_dict['validation'] = dataset_dict.pop('test')
```

[ ]:
```
# Check the dataset dictionary
dataset_dict
```

[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 8002
    })
    validation: Dataset({
        features: ['text', 'label'],
        num_rows: 2001
    })
})
```

[ ]:
```
# Define training, validation and test sets
trainset = dataset_dict['train']
valset = dataset_dict['validation']
testset = banking77_preprocessed['test']
```

[ ]:
```
# Change the format as dataframe to save the test set
banking77_preprocessed.set_format(type='pandas')
```

[ ]:
```python
# Define a dataframe for the test set
test_df = banking77_preprocessed['test'][:]
```

[ ]:
```python
# Specify the test set file path
csv_file_path = "/content/drive/MyDrive/1. NLP CW/DistilBERT/test set/testset.
↪csv"

# Save the DataFrame as CSV
test_df.to_csv(csv_file_path, index=False)
```

[ ]:
```python
# Check the number of data points
pprint(len(trainset))
pprint(len(valset))
pprint(len(testset))
```

8002
2001
3080

[ ]:
```python
# This code is derived from lab tutorial 8
# Import libraries
from transformers import DistilBertTokenizer

# Tokenization
tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132:
FutureWarning: `resume_download` is deprecated and will be removed in version
1.0.0. Downloads always resume when possible. If you want to force a new
download, use `force_download=True`.
  warnings.warn(
```

[ ]:
```python
# This code is derived from lab tutorial 8
# Tokenize the data
def tokenize(batch):
    return tokenizer(batch['text'], padding='max_length', truncation=True,␣
↪max_length=29) # Define the maximum length as 29

train_set = dataset_dict['train'].map(tokenize, batched=True)
val_set = dataset_dict['validation'].map(tokenize, batched=True)
test_set = banking77_preprocessed['test'].map(tokenize, batched=True)
```

```
Map:   0%|          | 0/8002 [00:00<?, ? examples/s]
Map:   0%|          | 0/2001 [00:00<?, ? examples/s]
Map:   0%|          | 0/3080 [00:00<?, ? examples/s]
```

[ ]:
```python
# Check the inside of test set
pprint(test_set[:1], sort_dicts=False)
```

```
{'text': ['locate card'],
 'label': [11],
 'input_ids': [[101,
                12453,
                4003,
                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]],
 'attention_mask': [[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]]}
```

[ ]:
```
# This code is derived from lab tutorial 8
# Set the data format
train_set.set_format('pt', columns=['input_ids', 'attention_mask', 'label'])
val_set.set_format('pt', columns=['input_ids', 'attention_mask', 'label'])
test_set.set_format('pt', columns=['input_ids', 'attention_mask', 'label'])
```

[ ]:
```
# Check the datatype of the training set and test set
pprint(train_set[:1])
pprint(val_set[:1])
pprint(test_set[:1])
```

```
{'attention_mask': tensor([[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]]),
 'input_ids': tensor([[  101,  5356,  2134,  2102,  2131,  3065, 10439,   102,
0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0]]),
 'label': tensor([20])}
{'attention_mask': tensor([[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]]),
 'input_ids': tensor([[  101,  3841, 12879, 24108,  2854, 23439,   102,     0,
0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0]]),
 'label': tensor([7])}
{'attention_mask': tensor([[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]]),
 'input_ids': tensor([[  101, 12453,  4003,   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]]),
 'label': tensor([11])}
```

DistilBERT

[ ]:
```python
# Import libraries
import torch
import torch.nn as nn

# Check the run time
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda:0
```

[ ]:
```python
# This code is derived from lab tutorial 8
from transformers import DistilBertModel

model = DistilBertModel.from_pretrained('distilbert-base-uncased')
```

[ ]:
```python
# This code is derived from lab tutorial 8
# Define the model architecture
class DistilBERT(nn.Module):
    def __init__(self, model):
        super(DistilBERT, self).__init__()
        self.model = model
        self.linear = nn.Linear(768, 77) # 77 classes

    def forward(self, input_ids, attention_mask):
        outputs = self.model(input_ids=input_ids, attention_mask=attention_mask)
        last_hidden_state = outputs.last_hidden_state[:, 0, :]
        logits = self.linear(last_hidden_state)
        return logits

# Define the model
model = DistilBERT(model)
model.to(device)
```

[ ]:
```
DistilBERT(
  (model): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
```

```
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): MultiHeadSelfAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
        )
      )
    )
  )
  (linear): Linear(in_features=768, out_features=77, bias=True)
)
```

[ ]:
```
# This code is derived from lab tutorial 8
# Set up the optimizer and loss function
optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)
loss_fn = nn.CrossEntropyLoss()
```

[ ]:
```
# Set up the data loader for each dataset
train_loader = torch.utils.data.DataLoader(train_set, batch_size=32,␣
 ↪shuffle=True)
val_loader = torch.utils.data.DataLoader(val_set, batch_size=32, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=32,␣
 ↪shuffle=False)
```

[ ]:
```
# Import the library to save the test set for the distilBERT model
import os

# Define the save function for test loader
def save_test_data(test_loader, save_dir):
    for i, batch in enumerate(test_loader):
        input_ids = batch['input_ids']
        attention_mask = batch['attention_mask']
        labels = batch['label']
```

```
        # Save each batch of data
        torch.save((input_ids, attention_mask, labels), os.path.join(save_dir,␣
↪f"test_batch_{i}.pt"))

# Specify the directory to save test data
save_dir = "/content/drive/MyDrive/1. NLP CW/DistilBERT"
save_test_data(test_loader, save_dir)
```

[ ]:
```
# Define the folder path to save the state dictionary
folder_path = '/content/drive/MyDrive/1. NLP CW/DistilBERT/'

# Define the dictionary file path for the model checkpoint
model_save_path = folder_path + 'distilBERT_model.pth'
```

[ ]:
```
#Free up GPU memory
torch.cuda.empty_cache()
```

[ ]:
```
# Define the train function
# Import time to measure the training time
import time

def train_and_evaluate(model, train_loader, val_loader, optimizer, loss_fn,␣
↪device, model_save_path):
    train_losses, val_losses = [], [] # Empty lists to store losses
    train_accuracies, val_accuracies = [], [] # Empty lists to store accuracies

    # Measure the total training time
    total_start_time = time.time()

    for epoch in range(5):
        # Training
        start_time = time.time() # Measure each training time
        model.train()

        epoch_train_loss = 0.0
        correct_train, total_train = 0, 0

        for batch in train_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['label'].to(device)

            optimizer.zero_grad()
            outputs = model(input_ids, attention_mask)
            loss = loss_fn(outputs, labels)
            loss.backward()
            optimizer.step()
```

```
            epoch_train_loss += loss.item() * input_ids.size(0) # Check the␣
↪train loss per epoch

            predictions_train = torch.round(torch.softmax(outputs, dim=1))
            predicted_train = torch.argmax(predictions_train, dim=1)
            total_train += labels.size(0)
            correct_train += (predicted_train == labels).sum().item()

        train_loss = epoch_train_loss / len(train_loader.dataset)
        train_accuracy = correct_train / total_train
        train_losses.append(train_loss) # Total train loss
        train_accuracies.append(train_accuracy)

        # Validation
        model.eval()
        correct_val, total_val = 0, 0
        epoch_val_loss = 0.0

        with torch.no_grad():
            for batch in val_loader:
                input_ids = batch['input_ids'].to(device)
                attention_mask = batch['attention_mask'].to(device)
                labels = batch['label'].to(device)

                outputs = model(input_ids, attention_mask)
                predictions_val = torch.round(torch.softmax(outputs, dim=1))
                predicted_val = torch.argmax(predictions_val, dim=1)

                loss_val = loss_fn(outputs, labels)
                epoch_val_loss += loss_val.item() * input_ids.size(0) # Check␣
↪the validation loss per epoch

                total_val += labels.size(0)
                correct_val += (predicted_val == labels).sum().item()

        val_loss = epoch_val_loss / len(val_loader.dataset)
        val_accuracy = correct_val / total_val
        val_losses.append(val_loss) # Total validation loss
        val_accuracies.append(val_accuracy)

        end_time = time.time()
        each_train_duration = end_time - start_time

        # Add 1 to epoch as it starts from 0
```

```
        print(f'Epoch {epoch+1} - Training Time: {each_train_duration:.3f}␣
↪seconds, Train Loss: {train_loss:.4f}, Train Accuracy: {train_accuracy:.4f},␣
↪Validation Loss: {val_loss:.4f}, Validation Accuracy: {val_accuracy:.4f}')

    total_end_time = time.time()
    total_train_duration = end_time - start_time
    print(f'Total training time: {total_train_duration:.3f} seconds')

    # Save the state dictionary
    torch.save(model.state_dict(), model_save_path)
    # Define the file name and path to save the model itself
    # This saving model code is derived from the tutorial of Huggingface's␣
↪DistilBERT (A notebook on how to finetune DistilBERT for multiclass␣
↪classification with PyTorch)
    # (https://huggingface.co/docs/transformers/en/model_doc/
↪distilbert#transformers.DistilBertConfig)
    output_model_file = '/content/drive/MyDrive/1. NLP CW/DistilBERT/
↪processed_distilbert.bin'
    model_to_save = model
    # Save the model itself
    torch.save(model_to_save, output_model_file)
    print('Model and state dictionary have been saved')

    return train_losses, val_losses, train_accuracies, val_accuracies
```

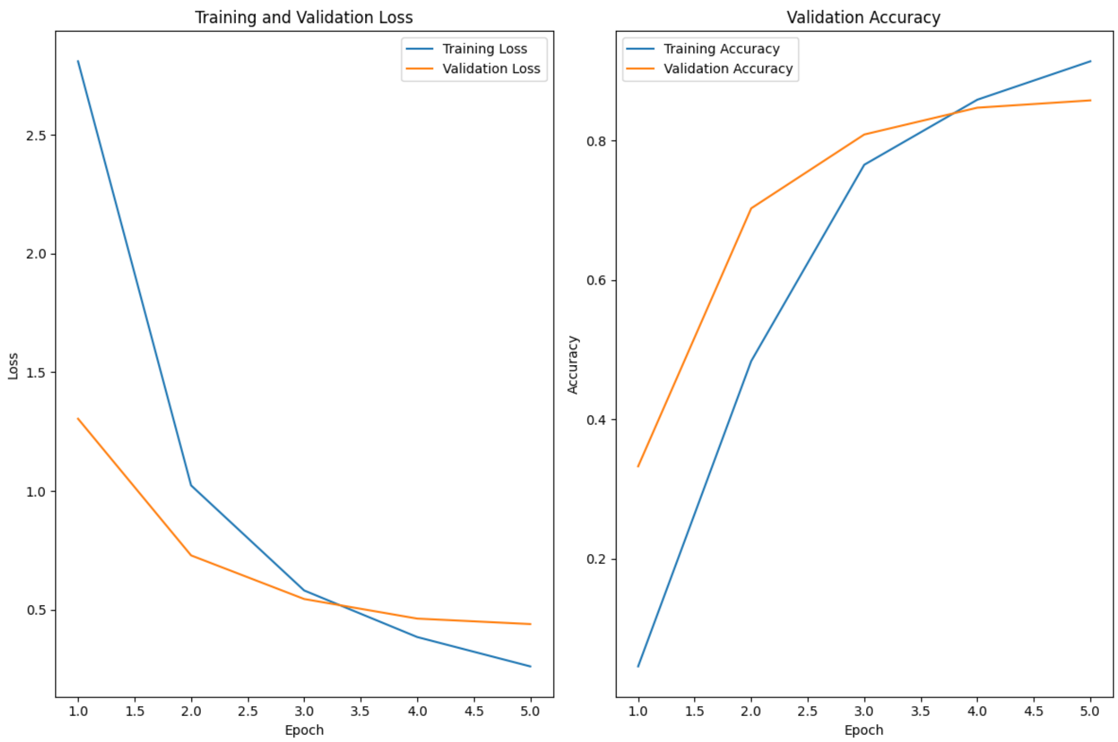
[ ]:
```
# Train the model
train_losses, val_losses, train_accuracies, val_accuracies =␣
↪train_and_evaluate(model, train_loader, val_loader, optimizer, loss_fn,␣
↪device, model_save_path)
```

```
Epoch 1 - Training Time: 24.662 seconds, Train Loss: 2.8097, Train Accuracy:
0.0452, Validation Loss: 1.3046, Validation Accuracy: 0.3323
Epoch 2 - Training Time: 23.877 seconds, Train Loss: 1.0237, Train Accuracy:
0.4834, Validation Loss: 0.7290, Validation Accuracy: 0.7026
Epoch 3 - Training Time: 24.640 seconds, Train Loss: 0.5813, Train Accuracy:
0.7652, Validation Loss: 0.5454, Validation Accuracy: 0.8086
Epoch 4 - Training Time: 24.036 seconds, Train Loss: 0.3852, Train Accuracy:
0.8585, Validation Loss: 0.4629, Validation Accuracy: 0.8471
Epoch 5 - Training Time: 24.423 seconds, Train Loss: 0.2616, Train Accuracy:
0.9136, Validation Loss: 0.4399, Validation Accuracy: 0.8576
Total training time: 24.423 seconds
Model and state dictionary have been saved
```

[ ]:
```
# Plot the loss and accuracy
# Define the plot fuction
def plot_training_curve(train_losses, val_losses, train_accuracies,␣
↪val_accuracies):
```

```
    epochs = range(1, len(train_losses) + 1) # Add 1 to the length of the list␣
 ↪as the epoch starts from 0

    plt.figure(figsize=(12, 8))

    # Plot training and validation losses
    plt.subplot(1, 2, 1)
    plt.plot(epochs, train_losses, label='Training Loss')
    plt.plot(epochs, val_losses, label='Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()

    # Plot validation accuracy
    plt.subplot(1, 2, 2)
    plt.plot(epochs, train_accuracies, label='Training Accuracy')
    plt.plot(epochs, val_accuracies, label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Validation Accuracy')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

[ ]:
```
# Plot the loss and accuracy of train and validation
plot_training_curve(train_losses, val_losses, train_accuracies, val_accuracies)
```



[ ]:
```
# Load the model to test
test_model = torch.load('/content/drive/MyDrive/1. NLP CW/DistilBERT/
↪processed_distilbert.bin', map_location=torch.device('cpu'))
# Match the state dictionary to the loaded model
state_dict = torch.load(model_save_path, map_location=torch.device('cpu'))
test_model.load_state_dict(state_dict)
```

[ ]:
```
<All keys matched successfully>
```

[ ]:
```
# Define the test function
def evaluate(model, test_loader):
    model.eval()
    correct = 0
    total = 0
    predictions_list = []
    labels_list = []
    with torch.no_grad():
        for batch in test_loader:
            input_ids = batch['input_ids']
            attention_mask = batch['attention_mask']
            labels = batch['label']

            outputs = model(input_ids, attention_mask)
```

```
            predictions = torch.argmax(outputs, dim=1)
            correct += (predictions == labels).sum().item()
            total += labels.size(0)

            predictions_list.extend(predictions.cpu().numpy()) # Make sure it␣
↪will run in CPU
            labels_list.extend(labels.cpu().numpy()) # Make sure it will run in␣
↪CPU

    accuracy = correct / total
    precision = precision_score(labels_list, predictions_list,␣
↪average='weighted')
    recall = recall_score(labels_list, predictions_list, average='weighted')
    f1 = f1_score(labels_list, predictions_list, average='weighted')

    return accuracy, precision, recall, f1
```

[ ]:
```
# Test the model
test_model.eval()

# Get the test accuracy
test_accuracy, test_precision, test_recall, test_f1 = evaluate(test_model,␣
↪test_loader)

print(f'Test Accuracy: {round((test_accuracy*100), 2)}')
print(f'Test Precision: {round((test_precision*100), 2)}')
print(f'Test Recall: {round((test_recall*100), 2)}')
print(f'Test F1 Score: {round((test_f1*100), 2)}')
```

```
Test Accuracy: 89.55
Test Precision: 90.08
Test Recall: 89.55
Test F1 Score: 89.56
```

[ ]:
```
# Make predictions on the test dataset
predictions = model.predict(test_set)

# Get the predicted labels
predicted_labels = [np.argmax(pred) for pred in predictions]

# Get the ground truth labels
true_labels = test_set['label']

# Initialize lists to store misclassified instances
misclassified_texts = []
misclassified_predicted_labels = []
misclassified_true_labels = []
```

```python
# Compare predictions with ground truth labels
for i in range(len(true_labels)):
    if predicted_labels[i] != true_labels[i]:
        # Add misclassified instance to lists
        misclassified_texts.append(test_set['text'][i])
        misclassified_predicted_labels.append(predicted_labels[i])
        misclassified_true_labels.append(true_labels[i])

# Print some misclassified instances for analysis
for i in range(min(10, len(misclassified_texts))):
    print("Text:", misclassified_texts[i])
    print("Predicted Label:", misclassified_predicted_labels[i])
    print("True Label:", misclassified_true_labels[i])
    print()
```

ver. 2) Use the unprocessed dataset

[ ]:
```python
# Perform train-test split to make a validation set
# Export only data set to split (training 80%, validation 20% from the training␣
↪set)
dataset_dict2 = banking77['train'].train_test_split(test_size=0.2)
```

[ ]:
```python
# Change the name 'test' to 'validation'
dataset_dict2['validation'] = dataset_dict2.pop('test')
```

[ ]:
```python
# Check the dataset dictionary
dataset_dict2
```

[ ]:
```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 8002
    })
    validation: Dataset({
        features: ['text', 'label'],
        num_rows: 2001
    })
})
```

[ ]:
```python
# Define a dataframe for the test set
test_df2 = banking77['test'][:]
```

[ ]:
```python
# Check the dataframe of the test set
test_df2
```

[ ]:
```
                                                     text  label
0                                    How do I locate my card?     11
1      I still have not received my new card, I order…     11
2      I ordered a card but it has not arrived. Help …     11
3       Is there a way to know when my card will arrive?     11
4                               My card has not arrived yet.     11
…                                                    …    …
3075      If i'm not in the UK, can I still get a card?     24
3076                  How many countries do you support?     24
3077               What countries do you do business in?     24
3078              What are the countries you operate in.     24
3079         Can the card be mailed and used in Europe?     24

[3080 rows x 2 columns]
```

[ ]:
```
# Specify the test set file path
csv_file_path = "/content/drive/MyDrive/1. NLP CW/DistilBERT/test set/testset2.
 ↪csv"

# Save the DataFrame as CSV
test_df2.to_csv(csv_file_path, index=False)
```

[ ]:
```
# Define training, validation and test sets
trainset2 = dataset_dict2['train']
valset2 = dataset_dict2['validation']
testset2 = banking77['test']
```

[ ]:
```
# This code is derived from lab tutorial 8
# Import libraries
from transformers import DistilBertTokenizer

# Tokenization
tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132:
FutureWarning: `resume_download` is deprecated and will be removed in version
1.0.0. Downloads always resume when possible. If you want to force a new
download, use `force_download=True`.
  warnings.warn(
```

[ ]:
```
# This code is derived from lab tutorial 8
# Tokenize the data
def tokenize(batch):
    return tokenizer(batch['text'], padding='max_length', truncation=True,␣
 ↪max_length=29) # Define the maximum length as 29

train_set2 = dataset_dict2['train'].map(tokenize, batched=True)
```

```
val_set2 = dataset_dict2['validation'].map(tokenize, batched=True)
test_set2 = banking77['test'].map(tokenize, batched=True)
```

```
Map:   0%|          | 0/8002 [00:00<?, ? examples/s]

Map:   0%|          | 0/2001 [00:00<?, ? examples/s]
```

[ ]:
```
# This code is derived from lab tutorial 8
# Set the data format
train_set2.set_format('pt', columns=['input_ids', 'attention_mask', 'label'])
val_set2.set_format('pt', columns=['input_ids', 'attention_mask', 'label'])
test_set2.set_format('pt', columns=['input_ids', 'attention_mask', 'label'])
```

DistilBERT

[ ]:
```
# Import libraries
import torch
import torch.nn as nn

# Check the run time
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda:0
```

[ ]:
```
# This code is derived from lab tutorial 8
from transformers import DistilBertModel

model = DistilBertModel.from_pretrained('distilbert-base-uncased')
```

[ ]:
```
# This code is derived from lab tutorial 8
# Define the model architecture
class DistilBERT(nn.Module):
    def __init__(self, model):
        super(DistilBERT, self).__init__()
        self.model = model
        self.linear = nn.Linear(768, 77) # 77 classes

    def forward(self, input_ids, attention_mask):
        outputs = self.model(input_ids=input_ids, attention_mask=attention_mask)
        last_hidden_state = outputs.last_hidden_state[:, 0, :]
        logits = self.linear(last_hidden_state)
        return logits

# Define the model
model = DistilBERT(model)
model.to(device)
```

[ ]:
```
DistilBERT(
  (model): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): MultiHeadSelfAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
        )
      )
    )
  )
  (linear): Linear(in_features=768, out_features=77, bias=True)
)
```

[ ]:
```
# This code is derived from lab tutorial 8
# Set up the optimizer and loss function
optimizer = torch.optim.Adam(model.parameters(), lr=2e-5)
loss_fn = nn.CrossEntropyLoss()
```

[ ]:
```
# Set up the data loader for each dataset
train_loader2 = torch.utils.data.DataLoader(train_set2, batch_size=32,
 ↪shuffle=True)
val_loader2 = torch.utils.data.DataLoader(val_set2, batch_size=32, shuffle=True)
test_loader2 = torch.utils.data.DataLoader(test_set2, batch_size=32,
 ↪shuffle=False)
```

[ ]:
```python
# Define the folder path to save the state dictionary
folder_path = '/content/drive/MyDrive/1. NLP CW/DistilBERT/'

# Define the dictionary file path for the model checkpoint
model_save_path = folder_path + 'unprocessed_distilBERT_model.pth'
```

[ ]:
```python
#Free up GPU memory
torch.cuda.empty_cache()
```

[ ]:
```python
# Define the train function
# Import time to measure the training time
import time

def train_and_evaluate(model, train_loader, val_loader, optimizer, loss_fn,␣
↪device, model_save_path):
    train_losses, val_losses = [], [] # Empty lists to store losses
    train_accuracies, val_accuracies = [], [] # Empty lists to store accuracies

    # Measure the total training time
    total_start_time = time.time()

    for epoch in range(5):
        # Training
        start_time = time.time() # Measure each training time
        model.train()

        epoch_train_loss = 0.0
        correct_train, total_train = 0, 0

        for batch in train_loader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['label'].to(device)

            optimizer.zero_grad()
            outputs = model(input_ids, attention_mask)
            loss = loss_fn(outputs, labels)
            loss.backward()
            optimizer.step()

            epoch_train_loss += loss.item() * input_ids.size(0)

            predictions_train = torch.round(torch.softmax(outputs, dim=1))
            predicted_train = torch.argmax(predictions_train, dim=1)
            total_train += labels.size(0)
            correct_train += (predicted_train == labels).sum().item()
```

```python
        train_loss = epoch_train_loss / len(train_loader.dataset)
        train_accuracy = correct_train / total_train
        train_losses.append(train_loss)
        train_accuracies.append(train_accuracy)

        # Validation
        model.eval()
        correct_val, total_val = 0, 0
        epoch_val_loss = 0.0

        with torch.no_grad():
            for batch in val_loader:
                input_ids = batch['input_ids'].to(device)
                attention_mask = batch['attention_mask'].to(device)
                labels = batch['label'].to(device)

                outputs = model(input_ids, attention_mask)
                predictions_val = torch.round(torch.softmax(outputs, dim=1))
                predicted_val = torch.argmax(predictions_val, dim=1)

                loss_val = loss_fn(outputs, labels)
                epoch_val_loss += loss_val.item() * input_ids.size(0)

                total_val += labels.size(0)
                correct_val += (predicted_val == labels).sum().item()

        val_loss = epoch_val_loss / len(val_loader.dataset)
        val_accuracy = correct_val / total_val
        val_losses.append(val_loss)
        val_accuracies.append(val_accuracy)

        end_time = time.time()
        each_train_duration = end_time - start_time

        # Add 1 to epoch as it starts from 0
        print(f'Epoch {epoch+1} - Training Time: {each_train_duration:.3f}␣
↪seconds, Train Loss: {train_loss:.4f}, Train Accuracy: {train_accuracy:.4f},␣
↪Validation Loss: {val_loss:.4f}, Validation Accuracy: {val_accuracy:.4f}')

    total_end_time = time.time()
    total_train_duration = end_time - start_time
    print(f'Total training time: {total_train_duration:.3f} seconds')

    # Save the state dictionary
    torch.save(model.state_dict(), model_save_path)
    # Define the file name and path to save the model itself
```

```
    # This saving model code is derived from the tutorial of Huggingface's␣
↪DistilBERT (A notebook on how to finetune DistilBERT for multiclass␣
↪classification with PyTorch)
    # (https://huggingface.co/docs/transformers/en/model_doc/
↪distilbert#transformers.DistilBertConfig)
    output_model_file = '/content/drive/MyDrive/1. NLP CW/DistilBERT/
↪unprocessed_distilbert.bin'
    model_to_save = model
    # Save the model itself
    torch.save(model_to_save, output_model_file)
    print('Model and state dictionary have been saved')

    return train_losses, val_losses, train_accuracies, val_accuracies
```

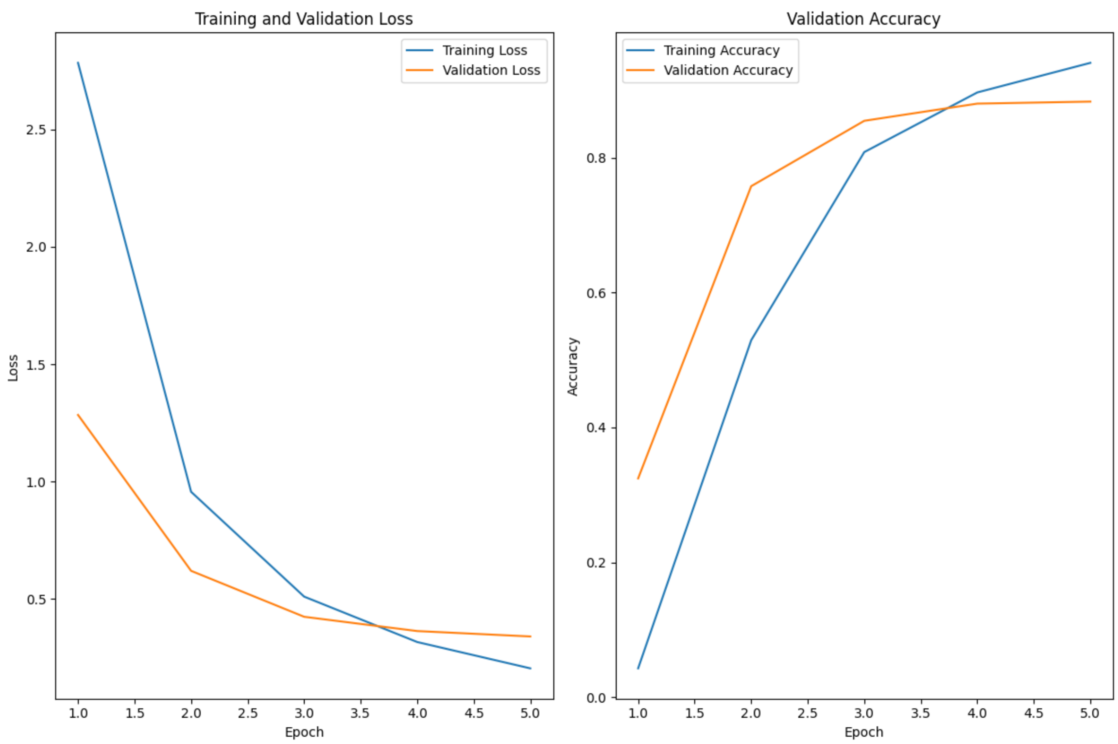
[ ]:
```
# Train the model
train_losses, val_losses, train_accuracies, val_accuracies =␣
↪train_and_evaluate(model, train_loader2, val_loader2, optimizer, loss_fn,␣
↪device, model_save_path)
```

```
Epoch 1 - Training Time: 24.963 seconds, Train Loss: 2.7830, Train Accuracy:
0.0427, Validation Loss: 1.2840, Validation Accuracy: 0.3243
Epoch 2 - Training Time: 25.533 seconds, Train Loss: 0.9567, Train Accuracy:
0.5292, Validation Loss: 0.6202, Validation Accuracy: 0.7576
Epoch 3 - Training Time: 23.989 seconds, Train Loss: 0.5104, Train Accuracy:
0.8082, Validation Loss: 0.4246, Validation Accuracy: 0.8546
Epoch 4 - Training Time: 24.265 seconds, Train Loss: 0.3171, Train Accuracy:
0.8967, Validation Loss: 0.3638, Validation Accuracy: 0.8801
Epoch 5 - Training Time: 24.323 seconds, Train Loss: 0.2051, Train Accuracy:
0.9405, Validation Loss: 0.3408, Validation Accuracy: 0.8831
Total training time: 24.323 seconds
Model and state dictionary have been saved
```

[ ]:
```
# Plot the loss and accuracy
# Define the plot fuction
def plot_training_curve(train_losses, val_losses, train_accuracies,␣
↪val_accuracies):
    epochs = range(1, len(train_losses) + 1) # Add 1 to the length of the list␣
↪as the epoch starts from 0

    plt.figure(figsize=(12, 8))

    # Plot training and validation losses
    plt.subplot(1, 2, 1)
    plt.plot(epochs, train_losses, label='Training Loss')
    plt.plot(epochs, val_losses, label='Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
```

```
    plt.title('Training and Validation Loss')
    plt.legend()

    # Plot validation accuracy
    plt.subplot(1, 2, 2)
    plt.plot(epochs, train_accuracies, label='Training Accuracy')
    plt.plot(epochs, val_accuracies, label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.title('Validation Accuracy')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

[ ]:
```
# Plot the loss and accuracy of train and validation
plot_training_curve(train_losses, val_losses, train_accuracies, val_accuracies)
```



[ ]:
```
# Load the model to test
test_model = torch.load('/content/drive/MyDrive/1. NLP CW/DistilBERT/
 ↪unprocessed_distilbert.bin', map_location=torch.device('cpu'))
# Match the state dictionary to the loaded model
state_dict = torch.load(model_save_path, map_location=torch.device('cpu'))
```

```
test_model.load_state_dict(state_dict)
```

[ ]: <All keys matched successfully>

```
[ ]: # Define the test function
def evaluate(model, test_loader):
    model.eval()
    correct = 0
    total = 0
    predictions_list = []
    labels_list = []
    with torch.no_grad():
        for batch in test_loader:
            input_ids = batch['input_ids']
            attention_mask = batch['attention_mask']
            labels = batch['label']

            outputs = model(input_ids, attention_mask)
            predictions = torch.argmax(outputs, dim=1)
            correct += (predictions == labels).sum().item()
            total += labels.size(0)

            predictions_list.extend(predictions.cpu().numpy()) # Make sure it␣
 ↪will run in CPU
            labels_list.extend(labels.cpu().numpy()) # Make sure it will run in␣
 ↪CPU

    accuracy = correct / total
    precision = precision_score(labels_list, predictions_list,␣
 ↪average='weighted')
    recall = recall_score(labels_list, predictions_list, average='weighted')
    f1 = f1_score(labels_list, predictions_list, average='weighted')

    return accuracy, precision, recall, f1
```

```
[ ]: # Test the model
test_model.eval()

# Get the test accuracy
test_accuracy, test_precision, test_recall, test_f1 = evaluate(test_model,␣
 ↪test_loader)

print(f'Test Accuracy: {round((test_accuracy*100), 2)}')
print(f'Test Precision: {round((test_precision*100), 2)}')
print(f'Test Recall: {round((test_recall*100), 2)}')
print(f'Test F1 Score: {round((test_f1*100), 2)}')
```

```
Test Accuracy: 83.28
Test Precision: 85.21
Test Recall: 83.28
Test F1 Score: 83.44
```

1.8 Intermediate results

The model below does not learn properly when applying the true maximum length.

[ ]:
```python
# Define the maximum length as 303
true_max_length_train_text = 303
true_max_length_train_text
```

[ ]:
```
303
```

[ ]:
```python
# Import the library for padding
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Padding and truncation will be added to post-texts
X_train_true_padded = pad_sequences(X_train_sequences,␣
 ↪maxlen=true_max_length_train_text, padding="post", truncating="post")
X_val_true_padded = pad_sequences(X_val_sequences,␣
 ↪maxlen=true_max_length_train_text, padding="post", truncating="post")
X_test_true_padded = pad_sequences(X_test_sequences,␣
 ↪maxlen=true_max_length_train_text, padding="post", truncating="post")
```

[ ]:
```python
# Check the dimension of the variables
print(X_train_true_padded.shape)
print(X_val_true_padded.shape)
print(X_test_true_padded.shape)
```

```
(7346, 303)
(1837, 303)
(3080, 303)
```

[ ]:
```python
# Check the first 3 elements of all X train variables
print(X_train_array[3])
print(X_train_sequences[3])
print(X_train_true_padded[3])
```

```
something wrong account balance didnt change transferred money
[50, 31, 2, 93, 23, 68, 127, 3]
[ 50  31   2  93  23  68 127   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   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   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    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    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    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    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    0    0    0    0    0    0    0    0    0]
```

[ ]:
```
# Define the output dimension for the embedding layer and hidden units
embedding_output_dim = 100
hidden_unit = 30
nlabel = 77

model = keras.models.Sequential()
model.add(layers.Embedding(voca_size, embedding_output_dim))
model.add(layers.LSTM(hidden_unit))
model.add(layers.Dense(nlabel, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy']) #, run_eagerly=True

# Summary the model
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 100)         206900

 lstm (LSTM)                 (None, 30)                15720

 dense (Dense)               (None, 77)                2387

=================================================================
Total params: 225007 (878.93 KB)
Trainable params: 225007 (878.93 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
# Define the folder path to save the model
folder_path = '/content/drive/MyDrive/1. NLP CW/'
```

```
# Define the file path for the model checkpoint
model_checkpoint_path = folder_path + 'LSTM1.keras'

# Define the model checkpoint
mc = tf.keras.callbacks.ModelCheckpoint(
    filepath=model_checkpoint_path,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
```

[ ]:
```
# Define early stopping
es =  tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3) # Random␣
↪number of patience
```

[ ]:
```
# Import time to measure the elapsed time
import time

# Measure time before training
start_time = time.time()

# Fit the model
history = model.fit(
    X_train_true_padded, y_train,
    epochs = 100,
    validation_data = (X_val_true_padded, y_val),
    callbacks = [mc, es],
    batch_size = 32)

# End the training time
end_time = time.time()

# Measure the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/100
230/230 [==============================] - 54s 199ms/step - loss: 4.3260 -
accuracy: 0.0174 - val_loss: 4.3061 - val_accuracy: 0.0196
Epoch 2/100
230/230 [==============================] - 34s 147ms/step - loss: 4.3104 -
accuracy: 0.0166 - val_loss: 4.3029 - val_accuracy: 0.0196
Epoch 3/100
230/230 [==============================] - 35s 152ms/step - loss: 4.3080 -
accuracy: 0.0186 - val_loss: 4.3034 - val_accuracy: 0.0196
Epoch 4/100
230/230 [==============================] - 33s 145ms/step - loss: 4.3072 -
```

```
accuracy: 0.0180 - val_loss: 4.3038 - val_accuracy: 0.0191
Epoch 5/100
230/230 [==============================] - 34s 147ms/step - loss: 4.3070 -
accuracy: 0.0181 - val_loss: 4.3025 - val_accuracy: 0.0196
Epoch 6/100
230/230 [==============================] - 32s 141ms/step - loss: 4.3070 -
accuracy: 0.0200 - val_loss: 4.3025 - val_accuracy: 0.0196
Epoch 7/100
230/230 [==============================] - 34s 147ms/step - loss: 4.3062 -
accuracy: 0.0189 - val_loss: 4.3024 - val_accuracy: 0.0191
Epoch 8/100
230/230 [==============================] - 33s 143ms/step - loss: 4.3063 -
accuracy: 0.0188 - val_loss: 4.3028 - val_accuracy: 0.0196
Epoch 9/100
230/230 [==============================] - 36s 154ms/step - loss: 4.3059 -
accuracy: 0.0180 - val_loss: 4.3022 - val_accuracy: 0.0196
Epoch 10/100
230/230 [==============================] - 32s 139ms/step - loss: 4.3059 -
accuracy: 0.0197 - val_loss: 4.3021 - val_accuracy: 0.0191
Epoch 11/100
230/230 [==============================] - 33s 143ms/step - loss: 4.3056 -
accuracy: 0.0166 - val_loss: 4.3023 - val_accuracy: 0.0196
Epoch 12/100
230/230 [==============================] - 34s 147ms/step - loss: 4.3058 -
accuracy: 0.0193 - val_loss: 4.3020 - val_accuracy: 0.0196
Epoch 13/100
230/230 [==============================] - 32s 139ms/step - loss: 4.3056 -
accuracy: 0.0199 - val_loss: 4.3023 - val_accuracy: 0.0196
Epoch 14/100
230/230 [==============================] - 33s 142ms/step - loss: 4.3055 -
accuracy: 0.0186 - val_loss: 4.3020 - val_accuracy: 0.0196
Epoch 15/100
230/230 [==============================] - 33s 145ms/step - loss: 4.3052 -
accuracy: 0.0193 - val_loss: 4.3021 - val_accuracy: 0.0191
Epoch 16/100
230/230 [==============================] - 33s 143ms/step - loss: 4.3054 -
accuracy: 0.0180 - val_loss: 4.3021 - val_accuracy: 0.0196
Epoch 17/100
230/230 [==============================] - 36s 156ms/step - loss: 4.3054 -
accuracy: 0.0174 - val_loss: 4.3021 - val_accuracy: 0.0196
Training time: 589.6596763134003 seconds
```

[ ]: