LSTM And T5 Model

+ Code + Text

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re

from tensorflow.keras.models import Sequential
```

Loading and Preprocessing of the Dataset

The dataset used for the project is then loaded from the csv file into a pandas dataframe.

```
counseldf = pd.read_csv("/content/counselchat-data.csv")
```

```
counseldf.head()
```

	questionID	questionTitle	questionText	questionUrl	topics	therapistName	
0	5566fab2a64752d71ec3ca69	Escalating disagreements between mother and wife	My wife and mother are having tense disagreeme...	https://counselchat.com/questions/escalating-d...	Family Conflict	Kristi King-Morgan, LMSW	htt
1	5566f94fa64752d71ec3ca64	I'm addicted to smoking. How can I stop?	I'm planning to have baby, so I have to quit s...	https://counselchat.com/questions/i-m-addicted...	Substance Abuse,Addiction	Rebecca Duellman	https:/
2	5567d26887a1cc0c3f3d8f46	Keeping secrets from my family	I have secrets in my mind, and I don't know wh...	https://counselchat.com/questions/keeping-secr...	Family Conflict	Jeevna Bajaj	https
3	556bed15c969ba5861709df5	The Underlying Causes of Being Possessive	I am extremely possessive in my relationships ...	https://counselchat.com/questions/the-underlyi...	Behavioral Change,Social Relationships	Rebecca Duellman	https:/
4	556ba115c969ba5861709de6	Can I control anxiety without medication?	I had a head injury a few years ago and my min...	https://counselchat.com/questions/can-i-contro...	Anxiety	Rebecca Duellman	https:/

```
counseldf.drop(['questionID', 'questionTitle', 'questionUrl', 'therapistName', 'therapistUrl','answerText', 'upvotes'], axis = 1, inpla
```

```
counseldf.head()
```

	questionText	topics
0	My wife and mother are having tense disagreeme...	Family Conflict
1	I'm planning to have baby, so I have to quit s...	Substance Abuse,Addiction
2	I have secrets in my mind, and I don't know wh...	Family Conflict
3	I am extremely possessive in my relationships ...	Behavioral Change,Social Relationships
4	I had a head injury a few years ago and my min...	Anxiety

```
counseldf.shape
```

```
(1658, 2)
```

```
counseldf.isnull().sum()
```

```
questionText    272
topics          185
dtype: int64
```

```
counseldf = counseldf.dropna(axis=0)
```

```
counseldf.isnull().sum()
```

```
questionText    0
topics          0
dtype: int64
```

```
counseldf.shape
```

```
(1377, 2)
```

Finally, we have the clean dataset which can now be processed with NLP techniques. After removing rows with null values, the size of the dataset remains at 1376 samples.

The 'topics' column is the target categories for the samples. As seen in the first few lines, some of the samples contain multiple labels. An LSTM network was tried to train using multiple labels but as the model failed to converge, it was decided that only the first label would be kept as the target label of the sample if it has multiple labels. The below code does just that.

```
counseldf['topics'] = counseldf['topics'].str.split(',')
counseldf['topics'] = counseldf['topics'].apply(lambda x: x[0])
```

```
counseldf.head(1376)
```

	questionText	topics
0	My wife and mother are having tense disagreeme...	Family Conflict
1	I'm planning to have baby, so I have to quit s...	Substance Abuse
2	I have secrets in my mind, and I don't know wh...	Family Conflict
3	I am extremely possessive in my relationships ...	Behavioral Change
4	I had a head injury a few years ago and my min...	Anxiety
...	...	...
1652	My ex-wife married and used me to have a child...	Parenting
1653	My grandson's step-mother sends him to school ...	Parenting
1654	My boyfriend is in recovery from drug addictio...	Relationships
1655	The birth mother attempted suicide several tim...	Family Conflict
1656	I think adult life is making him depressed and...	Relationships

1376 rows × 2 columns

```
counseldf['topics'].value_counts()
```

```
Relationships
246
Anxiety
178
Family Conflict
113
Depression
97
Marriage
89
Self-esteem
64
Parenting
59
Trauma
51
Human Sexuality
45
Behavioral Change
43
Relationship Dissolution
42
Intimacy
42
Counseling Fundamentals
39
Social Relationships
38
Anger Management
31
Professional Ethics
28
LGBTQ
24
```

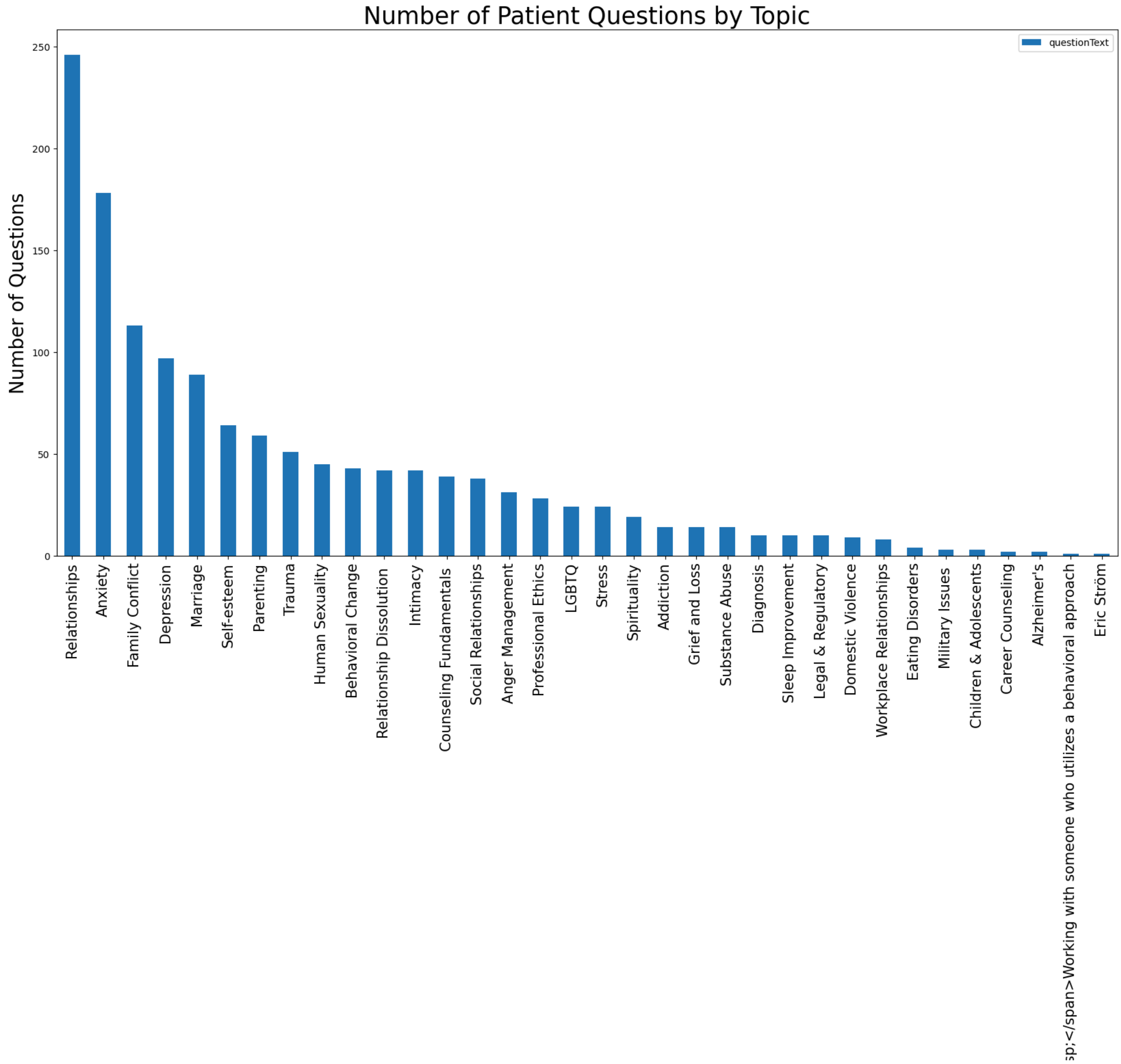
```
Stress
24
Spirituality
19
Substance Abuse
14
Grief and Loss
14
Addiction
14
Diagnosis
10
Legal & Regulatory
10
Sleep Improvement
10
Domestic Violence
9
Workplace Relationships
8
Eating Disorders
4
Military Issues
3
```

```
counseldf['topics'].count()
```

```
1377
```

```
fig, ax = plt.subplots(figsize=(20, 10))
counseldf.groupby('topics').agg('count').sort_values('questionText', ascending=False).plot.bar(ax=ax)
ax.set_title("Number of Patient Questions by Topic", fontsize=25)
ax.set_ylabel("Number of Questions", fontsize=20)
ax.set_xlabel("Topic", fontsize=20)
ax.set_xticklabels(ax.get_xticklabels(), fontsize=15)
plt.tight_layout()
plt.show()
```

```
<ipython-input-60-f832b81ee22c>:7: UserWarning: Tight layout not applied. The bottom and top margins cannot be made large enough to
  plt.tight_layout()
```




```
targetdf = pd.get_dummies(counseldf['topics'])
targetdf.head()
```

	trauma history or having a cold the time it occurred. Working with someone who utilizes a behavioral approach	Addiction	Alzheimer's	Anger Management	Anxiety	Behavioral Change	Career Counseling	Children & Adolescents	Counseling Fundamentals	Depressi
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	1	0	0	0	
4	0	0	0	0	1	0	0	0	0	

5 rows × 34 columns

```
# Preprocess function


import nltk, re
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from nltk.corpus import wordnet
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from collections import Counter

stop_words = stopwords.words('english')
normalizer = WordNetLemmatizer()

def get_part_of_speech(word):
    probable_part_of_speech = wordnet.synsets(word)
    pos_counts = Counter()
    pos_counts["n"] = len(  [ item for item in probable_part_of_speech if item.pos()=="n"]  )
    pos_counts["v"] = len(  [ item for item in probable_part_of_speech if item.pos()=="v"]  )
    pos_counts["a"] = len(  [ item for item in probable_part_of_speech if item.pos()=="a"]  )
    pos_counts["r"] = len(  [ item for item in probable_part_of_speech if item.pos()=="r"]  )
    most_likely_part_of_speech = pos_counts.most_common(1)[0][0]
    return most_likely_part_of_speech

def preprocess_text(text):
    cleaned = re.sub(r'\W+', ' ', text).lower()
    tokenized = word_tokenize(cleaned)
    normalized = [normalizer.lemmatize(token, get_part_of_speech(token)) for token in tokenized]
    return normalized
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
counseldf['questionText'][0]
```

```
'My wife and mother are having tense disagreements. In the past, they’ve had minor differences. For example, my wife would complain
to me my mother is too overbearing; my mother would complain my wife is lazy.\r\nHowever, it’s intensified lately. I think the caus
e is my wife talked back to her once. Now, any little disagreement is magnified, leading to major disagreements. What can I do?'
```

```
text = counseldf['questionText'][0]
cleaned = re.sub(r'\W+', ' ', text).lower()
print(cleaned)
```

```
my wife and mother are having tense disagreements in the past they ve had minor differences for example my wife would complain to me
```

Tokenizer

```
tokenized = word_tokenize(cleaned)
print(tokenized)
```

```
['my', 'wife', 'and', 'mother', 'are', 'having', 'tense', 'disagreements', 'in', 'the', 'past', 'they', 've', 'had', 'minor', 'diffe
```

Normalizer

```
normalized = [normalizer.lemmatize(token, get_part_of_speech(token)) for token in tokenized]
print(normalized)
```

```
['my', 'wife', 'and', 'mother', 'be', 'have', 'tense', 'disagreement', 'in', 'the', 'past', 'they', 've', 'have', 'minor', 'differen
```

Stop words Removal

```
processed_questionText = counseldf['questionText'].apply(lambda x: preprocess_text(x))
```

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

questionText_nostops = []
for title in processed_questionText:
    text_no_stops = [word for word in title if word not in stop_words]
    questionText_nostops.append(text_no_stops)


print(questionText_nostops[0])
```

```
['wife', 'mother', 'tense', 'disagreement', 'past', 'minor', 'difference', 'example', 'wife', 'would', 'complain', 'mother', 'overbe
```

```
import tensorflow as tf
from tensorflow.keras import preprocessing
from tensorflow.keras.models import Model
tf.random.set_seed(4)
tf.__version__
```

```
'2.15.0'
```

```
tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts( questionText_nostops )
tokenized_questions = tokenizer.texts_to_sequences( questionText_nostops )
print('Sample tokenized: {}'.format(tokenized_questions[0]))
print('==========================================================================================================\n')

length_list = list()
for token_seq in tokenized_questions:
    length_list.append( len( token_seq ))
max_input_length = np.array( length_list ).max()
print( 'Questions max length is {} words'.format( max_input_length ))
print('==========================================================================================================\n')

padded_questions = preprocessing.sequence.pad_sequences( tokenized_questions , maxlen=max_input_length , padding='post' )
input_data = np.array( padded_questions )
print( 'Input data shape -> {}\n'.format( input_data.shape ))
print('Input data sample->\n {}'.format(input_data[0]))
print('==========================================================================================================\n')

question_word_dict = tokenizer.word_index
num_question_tokens = len( question_word_dict )+1
print( 'Number of Question tokens = {}\n'.format( num_question_tokens))
print('Dictionary: {}'.format(question_word_dict))
```

```
Sample tokenized: [68, 101, 1759, 971, 49, 857, 972, 684, 68, 52, 476, 101, 1760, 101, 52, 476, 68, 858, 165, 1386, 150, 8, 216, 68,
==========================================================================================================

Questions max length is 220 words
==========================================================================================================

Input data shape -> (1377, 220)

Input data sample->
 [  68  101 1759  971   49  857  972  684   68   52  476  101 1760  101
   52  476   68  858  165 1386  150    8  216   68   18   43  166  971
 1761  352  626  971    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0]
==========================================================================================================

Number of Question tokens = 2420

Dictionary: {'feel': 1, 'get': 2, 'want': 3, 'like': 4, 'know': 5, 'time': 6, 'go': 7, 'think': 8, 'year': 9, 'say': 10, 'make': 11,
```

```
from sklearn.model_selection import train_test_split


x_train, x_test, y_train, y_test = train_test_split(input_data, targetdf, test_size = 0.3, random_state=2)
```

```
print(x_train.shape, x_test.shape)
print(y_train.shape, y_test.shape)
```

```
(963, 220) (414, 220)
(963, 34) (414, 34)
```

Apply SMOTE algo

```
pip install -U imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.12.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.25.2)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.3.0)
```

```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.datasets import make_classification
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

```
X, y = make_classification(n_classes=2, class_sep=2, weights=[0.1,  0.9], n_informative=3, n_redundant=1, flip_y=0, n_features=20, n_cl
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

```
model = LogisticRegression(random_state=42)
model.fit(X_train_resampled, y_train_resampled)
```

```
▾        LogisticRegression
LogisticRegression(random_state=42)
```

```
print("Test set score:", model.score(X_test, y_test))
```

```
Test set score: 1.0
```

```
print(x_train.shape, x_test.shape)
print(y_train.shape, y_test.shape)
```

```
(963, 220) (414, 220)
(963,) (414,)
```

```
# Convert numpy arrays to lists
X_train_list = X_train_resampled.tolist()
y_train_list = y_train_resampled.tolist()

# Print the lists
print("X_train_resampled:")
for row in X_train_list:
    print(row)

print("\ny_train_resampled:")
print(y_train_list)
```

```
X_train_resampled:
[1.0730087796146113, -3.1926620435719792, -0.2525095885708062, -0.4972073211294648, 0.3275018792101215, 0.43124749065184353, 1.8
[-0.7806670947846979, -1.2671772025324108, 0.5136248344916076, -0.2983033904056267, -1.2546646590394417, -0.46632108240172193, 0
[0.41862785593669566, -0.5340805053469384, 0.14130758772629332, 1.0146506787471257, 0.08948536216355198, 0.3926417524225065, -0.
[1.381448000298201, 0.011770280345170104, -2.674127193980635, -0.7186066026567806, -0.638076624721393, -0.363387937294465, 1.618
[0.14103636009841478, 0.737150833485878, 1.1302324392549863, -2.0419491449103386, 0.22877106236300113, 0.23621130114731917, 0.06
[0.9642190070830898, -0.3947169607797443, -0.049593950864100796, 0.23325627006292815, -1.2161818972699687, -1.492657382157762, 0
[1.0723827900883542, 0.6029647314120573, -0.19872581336611392, 0.3961541628349539, 1.7060822637975852, -0.8042358509598833, -0.7
[-1.1252043145182535, -0.4648385964618851, 0.1795562547259763, -0.1947182844216411, -0.15430325676795897, 0.1753810875907576, 0.
[0.28394523080729317, 0.5873989685022405, 0.11385024150127392, -0.23211701466719922, -1.0965104553540381, 1.68886262623733, -1.
[-0.8760535210053186, 0.8459522907766495, -1.7706037217862776, -0.2656014172835938, 0.2874866496963868, 1.9231196456469626, -1.2
[-0.6277372768520053, 0.10931635328946339, -0.7491887921537363, 0.3770157140577358, -0.3845552460937213, -0.4689329083299948, -0
```

```
[0.7371695820372347, 0.8080187925458685, -0.09509573900533472, -0.7596184034910803, 0.9322624035622205, -1.3996468302366252, -1.
[0.13753353495393958, -0.11049975417569206, 0.47501745063071593, -1.1292863093870136, -0.34231364871354497, 0.6775368593127222,
[-0.9266269200088034, -0.05298403450245096, -0.1884165413443542, -1.0365751236117267, 0.37274507766249754, 1.2048528724427472, 0
[0.9268800068381982, 0.6475676925701569, 0.040349208273140244, 0.596952460842647, 0.7071910971397228, 1.5359333555601418, 0.5195
[1.110072001074394, -0.029391051220279973, -0.5599837583084882, -1.8051096255468315, -0.5292194251340807, 0.942318084884891, -0.
[0.23674830934787133, -2.740500179331377, 1.1769700585556258, 1.7360410270704254, 0.9204430141330783, 1.5870502834429576, 0.2257
[0.35347194110941926, 0.47719346255199574, -0.2563724795341493, -0.3037057850304475, -0.11382862571241169, -0.6783475779270258,
[0.3861698878212157, -0.5775952514710012, 1.636892785676963, 0.750538030297768, -1.5382551416500136, 2.5210620191948006, -1.1329
[-0.08011324057369668, -1.2752079964321026, -0.807595776259671, -0.7897741218074396, -2.323719932661396, 1.3491009148145954, 0.4
[0.9057593083406789, 0.7296664067293706, -1.762634383515962, -0.10360740519221814, 1.04425335868, -0.37401752890737394, 1.122401
[-0.2854507025698915, 1.3772127645283943, 0.027452612231626428, -0.7628128291020341, 1.3779265469878332, 0.7763565439275977, 1.9
[0.4191272137309378, 0.12398039076408081, -2.0304808255870905, 0.8671436957705146, 0.07082480185812867, 0.14590991763658467, -0.
[0.8091698047476438, 0.47615636925189253, 0.3173641337953706, 0.5962207727442014, 0.4773614289774876, 0.07642315066921182, 2.603
[-0.3416648393289148, -0.32516383819484723, -0.11466971551925677, 0.07116066476597317, -0.5199295536831527, 0.3055255506555342,
[-0.40704083737523694, -0.5319046245146986, -1.7349927945677373, 1.2283074978159543, -0.3492233957337249, 1.5501476541405064, 1.
[0.4046516606550104, 0.7942750919412066, 0.1464389019733903, 1.041501504285878, 0.7945363173540928, 0.041989572819037056, 0.0094
[-0.06902377646662715, 0.2429445065298347, 3.231978965219223, -0.2058888614759287, -0.8863792876394272, -0.8273796622893304, 0.2
[-0.12505997432455754, 1.1608253826027215, 0.945681507631384, -0.46563151852045787, 1.7248067412753991, -0.21539077482172128, 0.
[0.07438181421174833, -3.3156063933141624, -0.5212372772700709, 1.4003646895996937, 0.2540854465062237, 0.7345995799925312, 1.40
[1.7132854190738354, -0.6622438518592986, 0.7185750596306807, -1.304836853425113, -1.6731726926900086, -1.5532802453162151, -0.3
[0.20069768201646931, 0.4268222849654089, -1.2049495586958234, 0.8654415474594342, 0.5195782751211648, -0.3322255126966573 4, -0.
[1.711915081853535, -1.953598165642685, 0.5197575154562364, 0.7230953713089183, 0.004948278380791308, -2.210805646410082, 0.0137
[-0.940220491850413, 0.5828373104993178, -0.28101315439322466, -1.2416481736723883, 1.0889429469355039, 2.1292270818695953, 0.12
[-0.467089603620586, -0.5348552861981358, -0.3174557594346386, -1.7282815091180703, -1.1672538061433384, -1.5072956303461982, -0
[0.14880419313175278, -0.28826578874656894, -0.31904904510555904, -0.18781941444482653, -0.9545856168023412, 1.04193023478068, 0
[-0.7182026298726497, 0.2380557682065338, -1.2802961658517418, 0.35890157561849656, 0.13324048200651795, 0.1262562372203581, -0.
[0.5885823256504555, -0.46765232683640146, -0.16341152854136748, 2.284916789489503, -1.108151124334631, -0.808300256936232, 0.11
[-0.9463213640892358, -1.9554204607883618, 1.9702678786675936, 0.7291528604789875, 0.7520166627686281, 0.4914174590255336, -1.13
[0.848114552064508, 0.23019942189180365, -0.388746516869197, 0.6554666940586921, 0.0613219879912778, -2.049121849863545, 0.47743
[0.47741993374501546, 1.2567057954865444, -0.32211789585033335, 0.3745683363287327, 1.125038123417162, -0.48780297473254186, 1.2
[-0.03966618429511068, -0.2571081210375041, -0.27763140868717967, 1.8629069593078629, 0.19675650055289512, -0.5408849096710548,
[0.3045350599237466, 0.5521288675513127, -0.966332769242731, -0.20942254869697396, 0.047809684626923005, -0.09056045489348392, 0
[-0.08142387176339733, 1.3813600695505168, -0.16061028260503263, -0.6232401158158315, 0.9216362115280307, -1.1139775682737068, -
[0.025822575427128975, 1.3008890505123885, -0.7934226605688783, -0.7643639272389554, 1.3562592890183176, 0.2585649221811676, 0.0
[0.7840437334536828, 0.4568146818210352, 0.2782943749591145, -1.3249267687072517, 0.9995126218744463, 1.404458845616503, 1.01887
[1.3738162377634489, 0.2610934125122378, -0.9500900038567293, 0.8716464494774195, -0.6635985565916019, 0.2152517362312304, 1.320
[-0.04421300327948067, -0.889893359776568, 0.9513379502395346, -0.21989120888536104, -1.5180442861056473, 0.7389531616680859, -0
[0.6789988183427019, -0.24332966575227713, 0.8124203581921899, 0.04141480789116345, 0.3061133893488673, 0.23870019769847153, 0.1
[-0.2592196732127245, 1.0900320335815858, -1.0238782858529254, 0.9498768758253179, 1.7093643083661887, -0.08338865000403453, -0.
[-0.23102335834076532, 0.4019596426323694, -0.3185935147944248, 1.5178844644429783, 1.0214998386412317, -0.3103947075495551, -0.
[0.32434104711858386, 0.7743876417341664, -1.4405197173431061, -0.8215507528617115, 1.0048342017312075, -0.14085668698488615, 0.
[0.7442690853230194, -0.2816296832219601, -1.9071931360663699, -0.8283907172988457, -0.9736118106128655, -0.8676393469952367, -0
[-0.895606736173639, 0.3954470111625961, -0.50685305542042, -0.5696502394467109, -0.6990727129006714, 0.7990258994975803, -0.579
[-0.9315456201705308, 0.19919813163604538, 0.6265093794906961, 1.34116841266675, -0.12686186174746084, -0.1102625536741896, -0.0
[1.5930726278176808, 0.7234165974171177, 0.2702618980200686, 0.1611364998382381 3, 0.37908963572870163, -1.4672549419843985, 0.17
```

```python
from imblearn.over_sampling import SMOTE
import numpy as np

# Assuming you have your original dataset X_train and y_train
# X_train and y_train should be numpy arrays or pandas DataFrames

# Instantiate the SMOTE algorithm
smote = SMOTE()

# Apply SMOTE to generate synthetic samples
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Print the shape of the resampled dataset
print("Shape of X_train_resampled:", X_train_resampled.shape)
print("Shape of y_train_resampled:", y_train_resampled.shape)

# Print the class distribution before and after SMOTE
print("Class distribution of y_train before SMOTE:")
print(np.unique(y_train, return_counts=True))

print("\nClass distribution of y_train_resampled after SMOTE:")
print(np.unique(y_train_resampled, return_counts=True))
```

```
Shape of X_train_resampled: (1730, 20)
Shape of y_train_resampled: (1730,)
Class distribution of y_train before SMOTE:
(array([0, 1]), array([ 98, 865]))

Class distribution of y_train_resampled after SMOTE:
(array([0, 1]), array([865, 865]))
```

```python
y_train_resampled.shape
```

```
(1730,)
```

```python
X_train_resampled.shape
```

```
(1730, 20)
```

Build LSTM Model Architecture

```
import numpy as np
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
import tensorflow as tf

# Instantiate the SMOTE algorithm
smote = SMOTE()

# Apply SMOTE to generate synthetic samples
X_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train)

# Convert y_train_resampled to one-hot encoded format
num_classes = 32 # Adjust this based on the number of classes in your dataset
y_train_resampled = tf.keras.utils.to_categorical(y_train_resampled, num_classes)

# Ensure y_test is also one-hot encoded
y_test = tf.keras.utils.to_categorical(y_test, num_classes)

# Define the LSTM model with additional dropout layers
LSTM_model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(None,)),
    tf.keras.layers.Embedding(num_question_tokens, 300, mask_zero=True),
    tf.keras.layers.Dropout(0.2),  # Dropout layer after the embedding layer
    tf.keras.layers.LSTM(128, return_sequences=False),
    tf.keras.layers.Dropout(0.2),  # Dropout layer after the LSTM layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),  # Dropout layer after the dense layer
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.3),  # Additional dropout layer after another dense layer
    tf.keras.layers.Dense(num_classes, activation='softmax')  # Output layer
])

# Compile the model
LSTM_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', tf.keras.metrics.Precision(), tf.keras.metric

# Train the model with increased epochs and batch size
history = LSTM_model.fit(X_train_resampled, y_train_resampled, epochs=50, batch_size=128, validation_data=(x_test, y_test))
```

```
Epoch 1/50
14/14 [==============================] - 15s 729ms/step - loss: 2.8627 - accuracy: 0.4936 - precision_2: 0.5843 - recall_2: 0.03
Epoch 2/50
14/14 [==============================] - 9s 636ms/step - loss: 0.8690 - accuracy: 0.5214 - precision_2: 0.5248 - recall_2: 0.495
Epoch 3/50
14/14 [==============================] - 8s 566ms/step - loss: 0.7943 - accuracy: 0.5214 - precision_2: 0.5210 - recall_2: 0.517
Epoch 4/50
14/14 [==============================] - 9s 626ms/step - loss: 0.7166 - accuracy: 0.5312 - precision_2: 0.5325 - recall_2: 0.530
Epoch 5/50
14/14 [==============================] - 9s 645ms/step - loss: 0.6890 - accuracy: 0.5578 - precision_2: 0.5562 - recall_2: 0.552
Epoch 6/50
14/14 [==============================] - 8s 547ms/step - loss: 0.6233 - accuracy: 0.6613 - precision_2: 0.6599 - recall_2: 0.656
Epoch 7/50
14/14 [==============================] - 9s 643ms/step - loss: 0.5289 - accuracy: 0.7572 - precision_2: 0.7597 - recall_2: 0.754
Epoch 8/50
14/14 [==============================] - 8s 596ms/step - loss: 0.4505 - accuracy: 0.8098 - precision_2: 0.8099 - recall_2: 0.807
Epoch 9/50
14/14 [==============================] - 7s 484ms/step - loss: 0.3888 - accuracy: 0.8370 - precision_2: 0.8372 - recall_2: 0.835
Epoch 10/50
14/14 [==============================] - 7s 543ms/step - loss: 0.3258 - accuracy: 0.8682 - precision_2: 0.8681 - recall_2: 0.867
Epoch 11/50
14/14 [==============================] - 9s 657ms/step - loss: 0.3050 - accuracy: 0.8717 - precision_2: 0.8716 - recall_2: 0.871
Epoch 12/50
14/14 [==============================] - 7s 524ms/step - loss: 0.2672 - accuracy: 0.9040 - precision_2: 0.9046 - recall_2: 0.904
Epoch 13/50
14/14 [==============================] - 10s 702ms/step - loss: 0.2423 - accuracy: 0.9139 - precision_2: 0.9144 - recall_2: 0.91
Epoch 14/50
14/14 [==============================] - 8s 606ms/step - loss: 0.2086 - accuracy: 0.9260 - precision_2: 0.9265 - recall_2: 0.926
Epoch 15/50
14/14 [==============================] - 7s 503ms/step - loss: 0.1851 - accuracy: 0.9301 - precision_2: 0.9306 - recall_2: 0.929
Epoch 16/50
14/14 [==============================] - 9s 582ms/step - loss: 0.1616 - accuracy: 0.9399 - precision_2: 0.9404 - recall_2: 0.939
Epoch 17/50
14/14 [==============================] - 8s 565ms/step - loss: 0.1405 - accuracy: 0.9514 - precision_2: 0.9520 - recall_2: 0.951
Epoch 18/50
14/14 [==============================] - 7s 486ms/step - loss: 0.1304 - accuracy: 0.9543 - precision_2: 0.9543 - recall_2: 0.953
Epoch 19/50
14/14 [==============================] - 8s 563ms/step - loss: 0.1514 - accuracy: 0.9451 - precision_2: 0.9451 - recall_2: 0.945
Epoch 20/50
```

```
14/14 [==============================] - 8s 540ms/step - loss: 0.1353 - accuracy: 0.9480 - precision_2: 0.9480 - recall_2: 0.948
Epoch 21/50
14/14 [==============================] - 7s 446ms/step - loss: 0.1325 - accuracy: 0.9520 - precision_2: 0.9520 - recall_2: 0.952
Epoch 22/50
14/14 [==============================] - 8s 543ms/step - loss: 0.1224 - accuracy: 0.9549 - precision_2: 0.9554 - recall_2: 0.954
Epoch 23/50
14/14 [==============================] - 6s 444ms/step - loss: 0.1113 - accuracy: 0.9584 - precision_2: 0.9584 - recall_2: 0.958
Epoch 24/50
14/14 [==============================] - 8s 587ms/step - loss: 0.1107 - accuracy: 0.9601 - precision_2: 0.9607 - recall_2: 0.960
Epoch 25/50
14/14 [==============================] - 7s 497ms/step - loss: 0.0968 - accuracy: 0.9590 - precision_2: 0.9590 - recall_2: 0.959
Epoch 26/50
14/14 [==============================] - 7s 485ms/step - loss: 0.1015 - accuracy: 0.9520 - precision_2: 0.9520 - recall_2: 0.952
Epoch 27/50
14/14 [==============================] - 8s 564ms/step - loss: 0.1043 - accuracy: 0.9578 - precision_2: 0.9578 - recall_2: 0.957
Epoch 28/50
14/14 [==============================] - 6s 467ms/step - loss: 0.1009 - accuracy: 0.9549 - precision_2: 0.9554 - recall_2: 0.954
```

```
LSTM_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_2 (Embedding)     (None, None, 300)         726000    
                                                                 
 lstm_2 (LSTM)               (None, 128)               219648    
                                                                 
 dense_4 (Dense)             (None, 512)               66048     
                                                                 
 dropout_2 (Dropout)         (None, 512)               0         
                                                                 
 dense_5 (Dense)             (None, 32)                16416     
                                                                 
=================================================================
Total params: 1028112 (3.92 MB)
Trainable params: 1028112 (3.92 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
LSTM_model.save('lstm_model_project', save_format='tf')
```

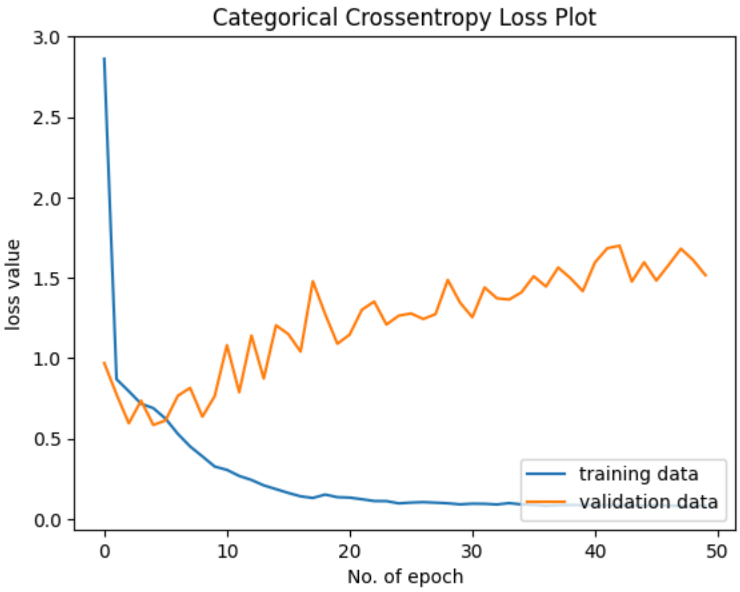
LSTM Model Performance

```
history_df = pd.DataFrame(history.history)
history_df['f1_score'] = (2 * history_df['precision_2']*history_df['recall_2'])/(history_df['precision_2']+history_df['recall_2'])
history_df['val_f1_score'] = (2 * history_df['val_precision_2']*history_df['val_recall_2'])/(history_df['val_precision_2']+history_df['v
```

```
history_df
```

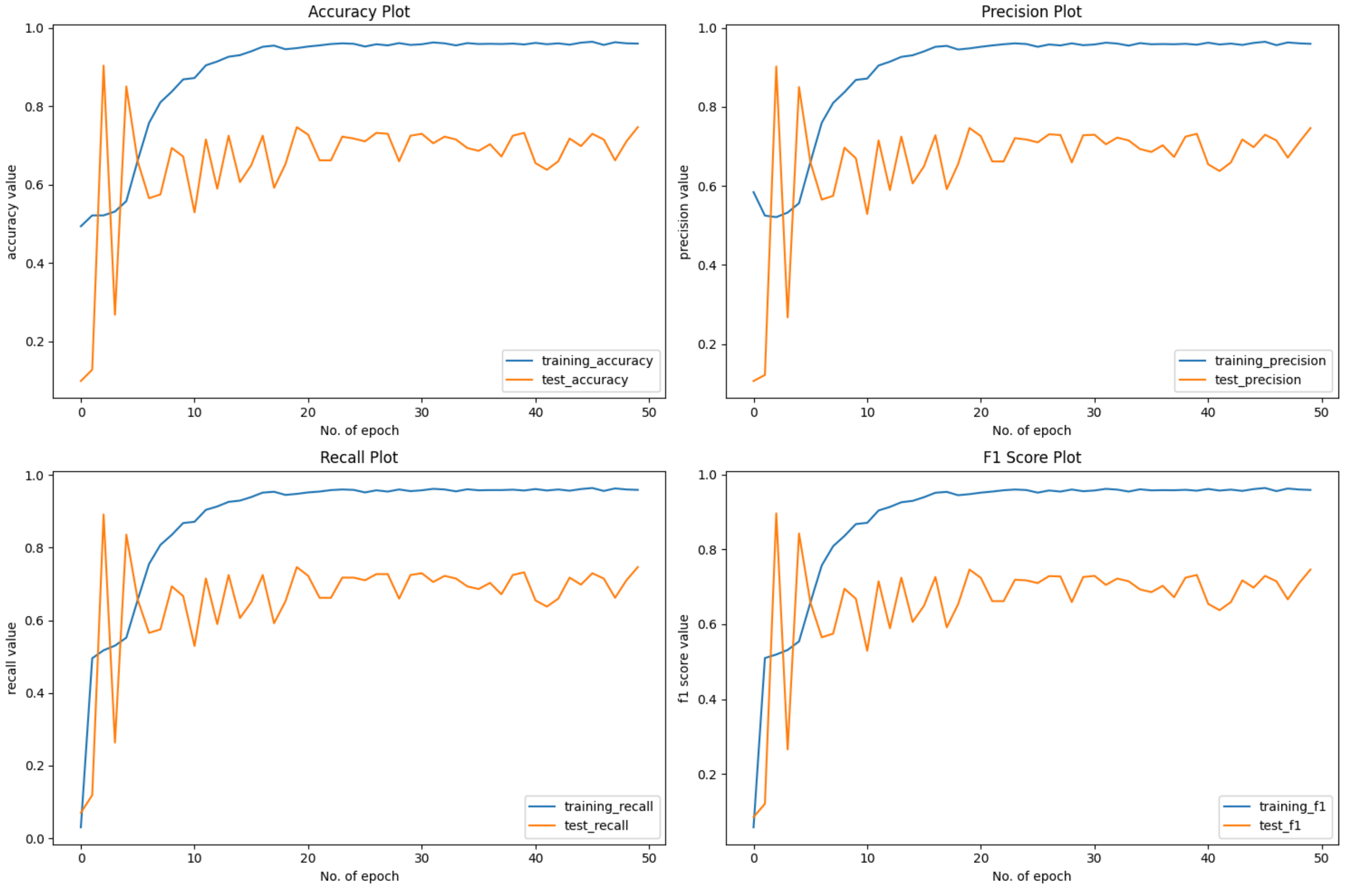
	loss	accuracy	precision_2	recall_2	val_loss	val_accuracy	val_precision_2	val_recall_2	f1_score	val_f1_score
0	2.862701	0.493642	0.584270	0.030058	0.969546	0.099034	0.106227	0.070048	0.057174	0.084425
1	0.869050	0.521387	0.524801	0.495376	0.774813	0.128019	0.121588	0.118357	0.509664	0.119951
2	0.794323	0.521387	0.520955	0.517341	0.594846	0.903382	0.902200	0.891304	0.519142	0.896719
3	0.716553	0.531214	0.532520	0.530058	0.735604	0.268116	0.267157	0.263285	0.531286	0.265207
4	0.689019	0.557803	0.556203	0.552023	0.584240	0.850242	0.850123	0.835749	0.554105	0.842875
5	0.623345	0.661272	0.659884	0.656069	0.613050	0.659420	0.660194	0.657005	0.657971	0.658596
6	0.528907	0.757225	0.759744	0.754913	0.766241	0.565217	0.565217	0.565217	0.757321	0.565217
7	0.450549	0.809827	0.809855	0.807514	0.815105	0.574879	0.574879	0.574879	0.808683	0.574879
8	0.388825	0.836994	0.837196	0.835260	0.636046	0.693237	0.696602	0.693237	0.836227	0.694915
9	0.325784	0.868208	0.868132	0.867630	0.763962	0.671498	0.669903	0.666667	0.867881	0.668281
10	0.305003	0.871676	0.871602	0.871098	1.080602	0.528986	0.528986	0.528986	0.871350	0.528986
11	0.267238	0.904046	0.904569	0.904046	0.787994	0.714976	0.714976	0.714976	0.904308	0.714976
12	0.242342	0.913873	0.914352	0.913295	1.140908	0.589372	0.589372	0.589372	0.913823	0.589372
13	0.208568	0.926012	0.926547	0.926012	0.873300	0.724638	0.724638	0.724638	0.926279	0.724638
14	0.185104	0.930058	0.930556	0.929480	1.204469	0.606280	0.606280	0.606280	0.930017	0.606280
15	0.161622	0.939884	0.940394	0.939306	1.149942	0.649758	0.649758	0.649758	0.939850	0.649758
16	0.140500	0.951445	0.951995	0.951445	1.040878	0.724638	0.728155	0.724638	0.951720	0.726392
17	0.130351	0.954335	0.954309	0.953757	1.478866	0.591787	0.591787	0.591787	0.954033	0.591787
18	0.151409	0.945087	0.945087	0.945087	1.274918	0.652174	0.655340	0.652174	0.945087	0.653753
19	0.135261	0.947977	0.947977	0.947977	1.089806	0.746377	0.746377	0.746377	0.947977	0.746377
20	0.132501	0.952023	0.952023	0.952023	1.146346	0.727053	0.725728	0.722222	0.952023	0.723971
21	0.122423	0.954913	0.955440	0.954335	1.301408	0.661836	0.661836	0.661836	0.954887	0.661836
22	0.111322	0.958381	0.958381	0.958381	1.352328	0.661836	0.661836	0.661836	0.958381	0.661836
23	0.110678	0.960116	0.960671	0.960116	1.209092	0.722222	0.720874	0.717391	0.960393	0.719128
24	0.096766	0.958960	0.958960	0.958960	1.264247	0.717391	0.717391	0.717391	0.958960	0.717391
25	0.101493	0.952023	0.952023	0.952023	1.278076	0.710145	0.710145	0.710145	0.952023	0.710145
26	0.104291	0.957803	0.957803	0.957803	1.243805	0.731884	0.730583	0.727053	0.957803	0.728814
27	0.100924	0.954913	0.955440	0.954335	1.274171	0.729469	0.728814	0.727053	0.954887	0.727932
28	0.097371	0.960694	0.960671	0.960116	1.487226	0.659420	0.659420	0.659420	0.960393	0.659420
29	0.090515	0.956069	0.956044	0.955491	1.345182	0.724638	0.728155	0.724638	0.955768	0.726392
30	0.094794	0.957803	0.957803	0.957803	1.254694	0.729469	0.729469	0.729469	0.957803	0.729469

```
plt.plot(history.history['loss'], label='training data')
plt.plot(history.history['val_loss'], label='validation data')
plt.title('Categorical Crossentropy Loss Plot')
plt.ylabel('loss value')
plt.xlabel('No. of epoch')
plt.legend(loc="lower right")
plt.show()
```




```
plt.subplots(figsize=(15,10))
ax = plt.subplot(2,2,1)
plt.plot(history.history['accuracy'], label='training_accuracy')
plt.plot(history.history['val_accuracy'], label='test_accuracy')
plt.title('Accuracy Plot')
plt.ylabel('accuracy value')
plt.xlabel('No. of epoch')
plt.legend(loc="lower right")

ax = plt.subplot(2,2,2)
plt.plot(history.history['precision_2'], label='training_precision')
plt.plot(history.history['val_precision_2'], label='test_precision')
plt.title('Precision Plot')
plt.ylabel('precision value')
plt.xlabel('No. of epoch')
plt.legend(loc="lower right")

ax = plt.subplot(2,2,3)
plt.plot(history.history['recall_2'], label='training_recall')
plt.plot(history.history['val_recall_2'], label='test_recall')
plt.title('Recall Plot')
plt.ylabel('recall value')
plt.xlabel('No. of epoch')
plt.legend(loc="lower right")

ax = plt.subplot(2,2,4)
plt.plot(history_df['f1_score'], label='training_f1')
plt.plot(history_df['val_f1_score'], label='test_f1')
plt.title('F1 Score Plot')
plt.ylabel('f1 score value')
plt.xlabel('No. of epoch')
plt.legend(loc="lower right")

plt.tight_layout()
plt.show()
```

```
<ipython-input-98-50af9b6b51d5>:2: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will b
  ax = plt.subplot(2,2,1)
```



Prediction

```
import tensorflow.keras as keras
import numpy as np


def predict(text):
    model = keras.models.load_model("lstm_model_project")
    processed = preprocess_text(text)
    text_no_stops = [word for word in processed if word not in stop_words]
    tokenized = tokenizer.texts_to_sequences(text_no_stops)
    padded = preprocessing.sequence.pad_sequences(tokenized, maxlen=max_input_length, padding='post')
    input_data = np.array(padded)
    output = model.predict(input_data)
    index = np.argmax(output[0])
    predicted_disorder = targetdf.columns[index]


    disorder_suggestions = {
        'Relationships': "If you're facing challenges in relationships, open communication and seeking therapy can help.",
        'Anxiety': "If you're experiencing anxiety, practicing relaxation techniques and seeking professional help may be beneficial.",
        'Family Conflict': "Dealing with family conflict can be challenging. Seeking family therapy or mediation may help.",
        'Depression': "Depression is a serious condition. Please consider seeking help from a mental health professional.",
        'Marriage': "Marriage counseling can provide support and guidance during challenging times in your relationship.",
        'Self-esteem': "Building self-esteem takes time and effort. Consider therapy or self-help resources.",
        'Parenting': "Parenting can be challenging. Seek support from other parents or consider parenting classes.",
        'Trauma': "Dealing with trauma requires patience and support. Consider therapy with a trauma-informed therapist.",
        'Human Sexuality': "Understanding human sexuality is complex. Seek guidance from a qualified therapist or counselor.",
        'Behavioral Change': "Changing behavior can be difficult. Consider therapy or counseling to explore underlying issues.",
        'Intimacy': "Building intimacy in relationships takes time and effort. Open communication is key.",
        'Relationship Dissolution': "Going through a relationship breakup is tough. Seek support from friends and family.",
        'Counseling Fundamentals': "Counseling can provide valuable support and guidance during challenging times.",
        'Social Relationships': "Building and maintaining social relationships is important for mental well-being.",
        'Anger Management': "Managing anger requires self-awareness and coping skills. Consider therapy or anger management classes.",
        'Professional Ethics': "Maintaining professional ethics is important in all fields. Consult with mentors or supervisors for guic
        'LGBTQ': "If you identify as LGBTQ+, seek out supportive communities and consider therapy with a LGBTQ+ affirming therapist.",
        'Stress': "Stress management techniques such as mindfulness and relaxation can help alleviate stress.",
        'Spirituality': "Exploring spirituality can provide comfort and meaning. Consider meditation or spiritual practices.",
        'Grief and Loss': "Grieving is a natural process. Seek support from loved ones or consider grief counseling.",
        'Substance Abuse': "If you're struggling with substance abuse, consider seeking help from a substance abuse counselor or support
        'Addiction': "Addiction is a complex issue. Seek help from addiction specialists or support groups.",
        'Legal & Regulatory': "Understanding legal and regulatory issues is important. Consult with legal experts or regulators for guic
        'Sleep Improvement': "Improving sleep hygiene can promote better sleep. Consider consulting with a sleep specialist.",
        'Diagnosis': "Receiving a diagnosis can be overwhelming. Seek support from healthcare professionals and loved ones.",
        'Domestic Violence': "If you're experiencing domestic violence, seek help from domestic violence hotlines or shelters.",
        'Workplace Relationships': "Navigating workplace relationships can be challenging. Seek guidance from HR or a workplace counselc
        'Eating Disorders': "If you suspect you have an eating disorder, seek help from eating disorder specialists or therapists.",
        'Military Issues': "Military service comes with unique challenges. Seek support from military organizations or mental health prc
        'Children & Adolescents': "Supporting children and adolescents' mental health is crucial. Consult with child psychologists or cc
        'Career Counseling': "Career counseling can provide clarity and guidance in career decisions. Seek out career counselors for sup
        'Alzheimer\'s': "Alzheimer's disease requires specialized care and support. Consult with healthcare professionals for guidance.'
    }

    disorder_medicines = {
        'Relationships': "Therapy sessions and counseling can be beneficial.",
        'Anxiety': "Medications like SSRIs or benzodiazepines may be prescribed by a healthcare professional.",
        'Family Conflict': "Family therapy sessions or individual counseling may be recommended.",
        'Depression': "Antidepressants or therapy are common treatments for depression.",
        'Marriage': "Couples therapy or individual counseling may be recommended to address marital issues.",
        'Self-esteem': "Therapy or self-help resources can help improve self-esteem.",
        'Parenting': "Parenting classes or therapy can provide support and guidance.",
        'Trauma': "Therapy techniques such as EMDR or cognitive-behavioral therapy can help address trauma.",
        'Human Sexuality': "Therapy with a sex therapist or counselor may help navigate issues related to human sexuality.",
        'Behavioral Change': "Therapy or counseling can help address underlying issues contributing to behavioral change.",
        'Intimacy': "Couples therapy or individual counseling may help improve intimacy in relationships.",
        'Relationship Dissolution': "Support from friends and family, as well as therapy, can help cope with a breakup.",
        'Counseling Fundamentals': "Counseling sessions with a qualified therapist can provide support and guidance.",
        'Social Relationships': "Engaging in social activities and seeking support from friends can improve social relationships.",
        'Anger Management': "Therapy, anger management classes, or mindfulness techniques can help manage anger.",
        'Professional Ethics': "Consulting with mentors or supervisors and seeking ethical guidance can help maintain professional ethic
        'LGBTQ': "Seeking support from LGBTQ+ affirming therapists or support groups can provide valuable support.",
        'Stress': "Stress management techniques, therapy, or mindfulness practices can help reduce stress levels.",
        'Spirituality': "Exploring spirituality through meditation, prayer, or connecting with a spiritual community can provide comfort
        'Grief and Loss': "Grief counseling, support groups, or therapy can provide support during the grieving process.",
        'Substance Abuse': "Seeking help from substance abuse counselors, support groups, or rehab programs can aid recovery.",
        'Addiction': "Addiction treatment programs, therapy, and support groups can provide support in recovery.",
        'Legal & Regulatory': "Consulting with legal experts or seeking advice from regulators can help navigate legal and regulatory is
        'Sleep Improvement': "Improving sleep hygiene, therapy, or medications prescribed by a sleep specialist can help improve sleep c
        'Diagnosis': "Seeking support from healthcare professionals, therapy, and support groups can help cope with a diagnosis.",
        'Domestic Violence': "Seeking help from domestic violence hotlines, shelters, or therapy can provide support and safety.",
        'Workplace Relationships': "HR support, workplace counseling, or therapy can help navigate workplace relationship challenges.",
        'Eating Disorders': "Treatment for eating disorders may include therapy, nutritional counseling, and medical monitoring.",
        'Military Issues': "Military support organizations, therapy, or counseling can provide support for military-related challenges.'
        'Children & Adolescents': "Child psychologists, school counselors, or therapy can provide support for children and adolescents.'
```

```
        'Career Counseling': "Career counselors, vocational rehabilitation, or therapy can provide guidance in career decisions.",
        'Alzheimer\'s': "Consulting with healthcare professionals, support groups, and specialized care can help manage Alzheimer's dise
    }


    suggestion = disorder_suggestions.get(predicted_disorder, "No suggestion available for this disorder")
    medicine = disorder_medicines.get(predicted_disorder, "No medicine available for this disorder")

    return predicted_disorder, suggestion, medicine


predicted_disorder, suggestion, medicine = predict("Text to predict the disorder")
print("Predicted Disorder:", predicted_disorder)
print("Suggestion:", suggestion)
print("Medicine:", medicine)
```

```
1/1 [==============================] - 1s 1s/step
Predicted Disorder: Addiction
Suggestion: Addiction is a complex issue. Seek help from addiction specialists or support groups.
Medicine: Addiction treatment programs, therapy, and support groups can provide support in recovery.
```

```
predict("My love life is not good")
```

```
1/1 [==============================] - 1s 1s/step
(' trauma history or having a cold the time it occurred.</span><span style=""line-height: 1.42857;"">  </span>Working
with someone who utilizes a behavioral approach',
 'No suggestion available for this disorder',
 'No medicine available for this disorder')
```

```
predict("I am going through a divorce from a narcissistic sociopath who left me for another woman after mentally and emotionally abusing
```

```
2/2 [==============================] - 4s 16ms/step
(' trauma history or having a cold the time it occurred.</span><span style=""line-height: 1.42857;"">  </span>Working
with someone who utilizes a behavioral approach',
 'No suggestion available for this disorder',
 'No medicine available for this disorder')
```

Build T5 Model Architecture

```
!pip install transformers[torch] tokenizers datasets evaluate rouge_score sentencepiece huggingface_hub --upgrade
```

```
Requirement already satisfied: transformers[torch] in /usr/local/lib/python3.10/dist-packages (4.38.2)
Requirement already satisfied: tokenizers in /usr/local/lib/python3.10/dist-packages (0.15.2)
Collecting datasets
  Downloading datasets-2.18.0-py3-none-any.whl (510 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 510.5/510.5 kB 3.5 MB/s eta 0:00:00
Collecting evaluate
  Downloading evaluate-0.4.1-py3-none-any.whl (84 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.1/84.1 kB 7.9 MB/s eta 0:00:00
Collecting rouge_score
  Downloading rouge_score-0.1.2.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/dist-packages (0.1.99)
Collecting sentencepiece
  Downloading sentencepiece-0.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 21.8 MB/s eta 0:00:00
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.10/dist-packages (0.20.3)
Collecting huggingface_hub
  Downloading huggingface_hub-0.21.4-py3-none-any.whl (346 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 346.4/346.4 kB 39.7 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (3.13.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2023.12.
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.31.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (0.4.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (4.66.2)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.2.1+cu121)
Collecting accelerate>=0.21.0 (from transformers[torch])
  Downloading accelerate-0.28.0-py3-none-any.whl (290 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 290.1/290.1 kB 26.0 MB/s eta 0:00:00
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 13.0 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (1.5.3)
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 11.1 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 15.8 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<=2024.2.0,>=2023.1.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (202
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.9.3)
Collecting responses<0.19 (from evaluate)
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from rouge_score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from rouge_score) (3.8.1)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from rouge_score) (1.16.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface_hub) (4.1
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate>=0.21.0->transformers[torch])
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)
```

```python
import nltk
from datasets import load_dataset
import evaluate
import numpy as np
from transformers import T5Tokenizer, DataCollatorForSeq2Seq
from transformers import T5ForConditionalGeneration, Seq2SeqTrainingArguments, Seq2SeqTrainer
```

```python
from datasets import load_dataset

dataset = load_dataset("Vedant64/counsel_chat")
dataset = dataset["train"].train_test_split(test_size=0.2)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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 s
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 data: 100% 3.02M/3.02M [00:00<00:00, 14.9MB/s]

Generating train split: 1658/0 [00:00<00:00, 14030.35 examples/s]

```python
tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-base")
model = T5ForConditionalGeneration.from_pretrained("google/flan-t5-base")
data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)
```

tokenizer_config.json: 100% 2.54k/2.54k [00:00<00:00, 150kB/s]

spiece.model: 100% 792k/792k [00:00<00:00, 12.5MB/s]

special_tokens_map.json: 100% 2.20k/2.20k [00:00<00:00, 132kB/s]

tokenizer.json: 100% 2.42M/2.42M [00:00<00:00, 49.3MB/s]

```
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, ar
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
```

config.json: 100% 1.40k/1.40k [00:00<00:00, 99.8kB/s]

model.safetensors: 100% 990M/990M [00:09<00:00, 62.0MB/s]

generation_config.json: 100% 147/147 [00:00<00:00, 11.0kB/s]

```python
dataset
```

```
DatasetDict({
    train: Dataset({
        features: ['questionID', 'questionTitle', 'questionText', 'questionUrl', 'topics', 'therapistName', 'therapistUrl', 
'answerText', 'upvotes'],
        num_rows: 1326
    })
    test: Dataset({
        features: ['questionID', 'questionTitle', 'questionText', 'questionUrl', 'topics', 'therapistName', 'therapistUrl', 
'answerText', 'upvotes'],
        num_rows: 332
    })
})
```

```python
dataset = dataset.filter(lambda example: all(value is not None for value in example.values()))
```

Filter: 100% 1326/1326 [00:00<00:00, 16219.68 examples/s]

Filter: 100% 332/332 [00:00<00:00, 6483.63 examples/s]

```
prefix = "Please answer this question: "

# Define the preprocessing function

def preprocess_function(examples):
   """Add prefix to the sentences, tokenize the text, and set the labels"""
   # The "inputs" are the tokenized answer:
   inputs = [prefix + doc for doc in examples["questionText"]]
   model_inputs = tokenizer(inputs, max_length=128, truncation=True)

   # The "labels" are the tokenized outputs:
   labels = tokenizer(text_target=examples["topics"],
                      max_length=512,
                      truncation=True)

   model_inputs["labels"] = labels["input_ids"]
   return model_inputs


tokenized_dataset = dataset.map(preprocess_function, batched=True)
```

```
Map: 100%    1093/1093 [00:00<00:00, 2426.52 examples/s]
Map: 100%    279/279 [00:00<00:00, 1951.05 examples/s]
```

```
nltk.download("punkt", quiet=True)
metric = evaluate.load("rouge")

def compute_metrics(eval_preds):
    preds, labels = eval_preds

    # decode preds and labels
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # rougeLSum expects newline after each sentence
    decoded_preds = ["\n".join(nltk.sent_tokenize(pred.strip())) for pred in decoded_preds]
    decoded_labels = ["\n".join(nltk.sent_tokenize(label.strip())) for label in decoded_labels]

    result = metric.compute(predictions=decoded_preds, references=decoded_labels, use_stemmer=True)
    return result
```

```
Downloading builder script: 100%    6.27k/6.27k [00:00<00:00, 402kB/s]
```

```
training_args = Seq2SeqTrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=3e-4,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=4,
    weight_decay=0.01,
    save_total_limit=3,
    num_train_epochs=50,
    predict_with_generate=True,
    push_to_hub=False
)

# Set up trainer
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
```

```
/usr/local/lib/python3.10/dist-packages/accelerate/accelerator.py:432: FutureWarning: Passing the following arguments to `Accelerat
dataloader_config = DataLoaderConfiguration(dispatch_batches=None, split_batches=False, even_batches=True, use_seedable_sampler=True
  warnings.warn(
```

```
trainer.train()
```

[3450/3450 54:26, Epoch 50/50]

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
1	No log	0.534262	0.595873	0.247909	0.575416	0.576717
2	No log	0.423535	0.675094	0.351109	0.666786	0.668096
3	No log	0.376831	0.761100	0.438880	0.752157	0.751822
4	No log	0.370361	0.790000	0.470242	0.787814	0.786903
5	No log	0.390049	0.826282	0.521889	0.812714	0.812647
6	No log	0.411387	0.810983	0.482855	0.805325	0.805146
7	No log	0.516917	0.825932	0.526818	0.815495	0.815745
8	0.355200	0.510018	0.820057	0.515156	0.813732	0.813772
9	0.355200	0.579280	0.818408	0.523750	0.809623	0.809561
10	0.355200	0.614593	0.822107	0.533393	0.814311	0.815112
11	0.355200	0.624442	0.820157	0.533094	0.814577	0.813742
12	0.355200	0.702807	0.824100	0.522111	0.815672	0.815600
13	0.355200	0.701660	0.847687	0.545861	0.839529	0.840092
14	0.355200	0.719518	0.817273	0.531422	0.813253	0.813228
15	0.040000	0.685036	0.826984	0.536013	0.818591	0.818979
16	0.040000	0.733718	0.819526	0.522760	0.812511	0.812148
17	0.040000	0.751757	0.829796	0.532378	0.821109	0.820959
18	0.040000	0.798812	0.818488	0.522680	0.811634	0.810999
19	0.040000	0.783629	0.819210	0.522222	0.811973	0.811602
20	0.040000	0.777112	0.821121	0.530944	0.815446	0.815310