```
In [28]: !pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.16.2-cp311-cp311-macosx_10_15_x86_64.whl.metadata (4.1 kB)
Collecting absl-py>=1.0.0 (from tensorflow)
  Downloading absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow)
  Downloading flatbuffers-25.2.10-py2.py3-none-any.whl.metadata (875 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow)
  Downloading gast-0.6.0-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting h5py>=3.10.0 (from tensorflow)
  Downloading h5py-3.13.0-cp311-cp311-macosx_10_9_x86_64.whl.metadata (2.5 kB)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-macosx_10_9_x86_64.whl.metadata (5.2 kB)
Collecting ml-dtypes~=0.3.1 (from tensorflow)
  Downloading ml_dtypes-0.3.2-cp311-cp311-macosx_10_9_universal2.whl.metadata (20 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow)
  Downloading opt_einsum-3.4.0-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: packaging in /Users/andrewfox/anaconda3/lib/python3.11/site-packages
(from tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0
dev,>=3.20.3 in /Users/andrewfox/anaconda3/lib/python3.11/site-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in /Users/andrewfox/anaconda3/lib/python3.11/sit
e-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /Users/andrewfox/anaconda3/lib/python3.11/site-package
s (from tensorflow) (68.2.2)
Requirement already satisfied: six>=1.12.0 in /Users/andrewfox/anaconda3/lib/python3.11/site-packag
es (from tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow)
  Downloading termcolor-2.5.0-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in /Users/andrewfox/anaconda3/lib/python3.1
1/site-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /Users/andrewfox/anaconda3/lib/python3.11/site-pack
ages (from tensorflow) (1.14.1)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow)
  Downloading grpcio-1.70.0-cp311-cp311-macosx_10_14_universal2.whl.metadata (3.9 kB)
Collecting tensorboard<2.17,>=2.16 (from tensorflow)
  Downloading tensorboard-2.16.2-py3-none-any.whl.metadata (1.6 kB)
Collecting keras>=3.0.0 (from tensorflow)
  Downloading keras-3.8.0-py3-none-any.whl.metadata (5.8 kB)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow)
  Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-macosx_10_14_x86_64.whl.metadata (14
kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /Users/andrewfox/anaconda3/lib/python3.11/si
te-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /Users/andrewfox/anaconda3/lib/python3.11/site
-packages (from astunparse>=1.6.0->tensorflow) (0.41.2)
Requirement already satisfied: rich in /Users/andrewfox/anaconda3/lib/python3.11/site-packages (fro
m keras>=3.0.0->tensorflow) (13.3.5)
Collecting namex (from keras>=3.0.0->tensorflow)
  Downloading namex-0.0.8-py3-none-any.whl.metadata (246 bytes)
Collecting optree (from keras>=3.0.0->tensorflow)
  Downloading optree-0.14.0-cp311-cp311-macosx_10_9_universal2.whl.metadata (47 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.7/47.7 kB 1.8 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/andrewfox/anaconda3/lib/python3.1
1/site-packages (from requests<3,>=2.21.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/andrewfox/anaconda3/lib/python3.11/site-packa
ges (from requests<3,>=2.21.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/andrewfox/anaconda3/lib/python3.11/site
-packages (from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /Users/andrewfox/anaconda3/lib/python3.11/site
-packages (from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in /Users/andrewfox/anaconda3/lib/python3.11/site-pa
ckages (from tensorboard<2.17,>=2.16->tensorflow) (3.4.1)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard<2.17,>=2.16->tensorflow)
  Downloading tensorboard_data_server-0.7.2-py3-none-macosx_10_9_x86_64.whl.metadata (1.1 kB)
Requirement already satisfied: werkzeug>=1.0.1 in /Users/andrewfox/anaconda3/lib/python3.11/site-pa
ckages (from tensorboard<2.17,>=2.16->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Users/andrewfox/anaconda3/lib/python3.11/site-
packages (from werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in /Users/andrewfox/anaconda3/lib/pytho
n3.11/site-packages (from rich->keras>=3.0.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /Users/andrewfox/anaconda3/lib/python3.1
1/site-packages (from rich->keras>=3.0.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in /Users/andrewfox/anaconda3/lib/python3.11/site-package
s (from markdown-it-py<3.0.0,>=2.2.0->rich->keras>=3.0.0->tensorflow) (0.1.0)
Downloading tensorflow-2.16.2-cp311-cp311-macosx_10_15_x86_64.whl (259.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 259.6/259.6 MB 3.5 MB/s eta 0:00:00:00:0100:01
Downloading absl_py-2.1.0-py3-none-any.whl (133 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.7/133.7 kB 4.2 MB/s eta 0:00:00
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-25.2.10-py2.py3-none-any.whl (30 kB)
Downloading gast-0.6.0-py3-none-any.whl (21 kB)
Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 3.0 MB/s eta 0:00:00
Downloading grpcio-1.70.0-cp311-cp311-macosx_10_14_universal2.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 11.6 MB/s eta 0:00:0000:0100:01
Downloading h5py-3.13.0-cp311-cp311-macosx_10_9_x86_64.whl (3.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.4/3.4 MB 11.2 MB/s eta 0:00:0000:0100:01
Downloading keras-3.8.0-py3-none-any.whl (1.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 10.4 MB/s eta 0:00:0000:0100:01
Downloading libclang-18.1.1-py2.py3-none-macosx_10_9_x86_64.whl (26.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.5/26.5 MB 12.1 MB/s eta 0:00:0000:0100:01
Downloading ml_dtypes-0.3.2-cp311-cp311-macosx_10_9_universal2.whl (389 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 389.8/389.8 kB 8.5 MB/s eta 0:00:00:00:01
Downloading opt_einsum-3.4.0-py3-none-any.whl (71 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.9/71.9 kB 2.4 MB/s eta 0:00:00
Downloading tensorboard-2.16.2-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 12.3 MB/s eta 0:00:0000:0100:01
Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-macosx_10_14_x86_64.whl (2.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 11.4 MB/s eta 0:00:00a 0:00:01
Downloading termcolor-2.5.0-py3-none-any.whl (7.8 kB)
Downloading tensorboard_data_server-0.7.2-py3-none-macosx_10_9_x86_64.whl (4.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.8/4.8 MB 11.1 MB/s eta 0:00:0000:010:01m
Downloading namex-0.0.8-py3-none-any.whl (5.8 kB)
Downloading optree-0.14.0-cp311-cp311-macosx_10_9_universal2.whl (619 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 619.8/619.8 kB 8.1 MB/s eta 0:00:00:00:0100:01
Installing collected packages: namex, libclang, flatbuffers, termcolor, tensorflow-io-gcs-filesyste
m, tensorboard-data-server, optree, opt-einsum, ml-dtypes, h5py, grpcio, google-pasta, gast, astunp
arse, absl-py, tensorboard, keras, tensorflow
  Attempting uninstall: h5py
    Found existing installation: h5py 3.9.0
    Uninstalling h5py-3.9.0:
      Successfully uninstalled h5py-3.9.0
Successfully installed absl-py-2.1.0 astunparse-1.6.3 flatbuffers-25.2.10 gast-0.6.0 google-pasta-
0.2.0 grpcio-1.70.0 h5py-3.13.0 keras-3.8.0 libclang-18.1.1 ml-dtypes-0.3.2 namex-0.0.8 opt-einsum-
3.4.0 optree-0.14.0 tensorboard-2.16.2 tensorboard-data-server-0.7.2 tensorflow-2.16.2 tensorflow-i
o-gcs-filesystem-0.37.1 termcolor-2.5.0
```

```
In [9]:
import pandas as pd
import numpy as np
# For text cleaning
import re
import nltk
from nltk.corpus import stopwords #for stopword removal
from nltk.stem import WordNetLemmatizer #for lemmatization
from nltk.corpus import wordnet
# For data splitting
from sklearn.model_selection import train_test_split
# For tokenizing and padding
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
# For model building
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense
# For model evaluation
from sklearn.metrics import classification_report
# For plotting
import matplotlib.pyplot as plt
```

```
2025-02-23 20:37:07.261341: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow bi
nary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the ap
propriate compiler flags.
```

```
In [11]: df = pd.read_csv("socialmedia-disaster-tweets-DFE.csv")
```

Dataset Preparation

We are only interested in the following columns:

- choose_one: target variable, whether tweet is about an actual disaster or not
- text: the tweet
- keyword: keyword from the tweet, potential train variable
- location: where tweet was sent form, potential train variable

```
In [15]:
# Filter for relevant columns
data = df[['choose_one', 'text', 'keyword', 'location']]

# Change column names
data.columns = ['disaster', 'tweet', 'keyword', 'location']

# Remove 'Can't Decide' rows
data = data[data['disaster'] != "Can't Decide"]

# Drop NA rows
data = data.dropna(subset=['disaster', 'tweet'])

# One-Hot encode 'disaster' column
data['disaster'] = data['disaster'].replace({"Relevant": 1, "Not Relevant": 0})

data.head()
```

Out[15]:

	disaster	tweet	keyword	location
0	1	Just happened a terrible car crash	NaN	NaN
1	1	Our Deeds are the Reason of this #earthquake M...	NaN	NaN
2	1	Heard about #earthquake is different cities, s...	NaN	NaN
3	1	there is a forest fire at spot pond, geese are...	NaN	NaN
4	1	Forest fire near La Ronge Sask. Canada	NaN	NaN

EDA

Data Preprocessing

Text Cleaning

```
In [20]:
# Get English stop words
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# Create lemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
```

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

```
In [22]:
def clean(text, stopwords=False, lemmatize=False):
    text = re.sub(r'https?://\S+|www\.\S+', '', text).strip() #remove links like https://
    text = text.lower()
    text = text.strip()
    text = re.sub(r'[^a-z\s]', ' ', text) #replace special characters with a whitespace
    text = re.sub(r'\s+', ' ', text).strip() #remove double white spaces

    if stopwords == True:
        text = ' '.join(word for word in text.split() if word.lower() not in stop_words) #remove st

    if lemmatize == True:
        text = ' '.join(lemmatizer.lemmatize(word) for word in text.split()) #lemmatize nouns if sp
        text = ' '.join(lemmatizer.lemmatize(word, pos=wordnet.VERB) for word in text.split()) #lem

    return text

def clean_test(num, stopwords=False, lemmatize=False):
    sentence = data['tweet'][num]
    print(sentence)
    print()
    print(clean(sentence, stopwords, lemmatize))

clean_test(100, False, False)
```

```
http://t.co/GKYe6gjTk5 Had a #personalinjury accident this summer? Read our advice &amp; see how a
#solicitor can help #OtleyHour

had a personalinjury accident this summer read our advice amp see how a solicitor can help otleyhou
r
```

Neural Networks

Simple Dense Models

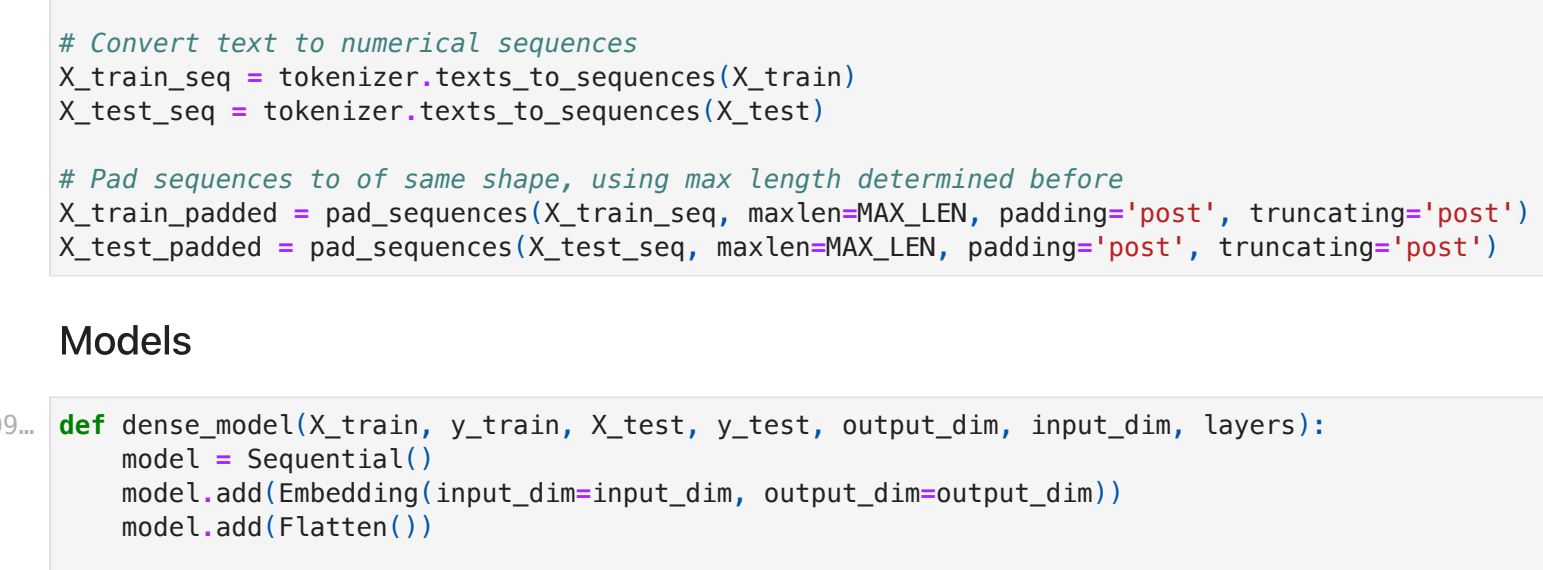
Split data

```
In [99]:
# Define features
X = data["tweet"].astype(str)

# Define target
y = data["disaster"]

# Clean data
X = X.apply(clean, stopwords=False, lemmatize=False)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Tokenize

Because we have to pad tweets so that they are all of the same size for the neural network, it is important to find a max length to which they will be filled to. Looking at the length of tweets in the dataset can help decide this value. We look at the 95th percentile of tweet lengths.

```
In [103…
# Find length of tweets
tweet_lengths = X.astype(str).apply(lambda x: len(x.split()))
MAX_LEN = int(np.percentile(tweet_lengths, 95))  # 95th percentile of tweet lengths

# Plot histogram
plt.figure(figsize=(6, 2))
plt.hist(tweet_lengths, bins=30, color="skyblue", edgecolor="black")
plt.axvline(np.percentile(tweet_lengths, 95), color='red', linestyle='dashed', label="95th percenti
plt.axvline(np.percentile(tweet_lengths, 99), color='green', linestyle='dashed', label="99th percer
plt.xlabel("Tweet Length (Number of Words)")
plt.ylabel("Frequency")
plt.title("Tweet Length Distribution")
plt.suptitle(f"Optimal max length: {MAX_LEN}", fontsize=10, color="gray", y=0.5, x=1.1)
plt.legend()
plt.show()
```



The maximum tweet length is not significantly longer than 25, ao a max length can be set at 50.

```
In [106…
# Tokenize our features
MAX_VOCAB = 10000  # Maximum unique words
MAX_LEN = 50
tokenizer = Tokenizer(num_words=MAX_VOCAB, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)  # Fit only on training data

# Convert text to numerical sequences
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

# Pad sequences to of same shape, using max length determined before
X_train_padded = pad_sequences(X_train_seq, maxlen=MAX_LEN, padding='post', truncating='post')
X_test_padded = pad_sequences(X_test_seq, maxlen=MAX_LEN, padding='post', truncating='post')
```

Models

```
In [109…
def dense_model(X_train, y_train, X_test, y_test, output_dim, input_dim, layers):
    model = Sequential()
    model.add(Embedding(input_dim=input_dim, output_dim=output_dim))
    model.add(Flatten())

    # Add layers
    for layers in layers:
        model.add(Dense(layers, activation='relu'))

    # Final layer, sigmoid for binary classification
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss="binary_crossentropy", optimizer="adam", metrics=['accuracy'])
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32, verbose

    # Evaluate
    test_loss, test_acc = model.evaluate(X_test, y_test)

    y_pred = (model.predict(X_test) > 0.5).astype("int32") # Convert probabilities to binary labels
    print(classification_report(y_test, y_pred))
```

```
In [111…
param_configs = [
    {"embedding_dim": 16, "layers": [16, 8]},
    {"embedding_dim": 32, "layers": [32, 16]},
    {"embedding_dim": 64, "layers": [64, 32, 16]}
]

cnt = 1
for config in param_configs:
    print("----------------------------------------------------------------------")
    print(f"Dense Model {cnt} with parameters embedding_dim={config['embedding_dim']} and layers={c
    print("----------------------------------------------------------------------")

    model = dense_model(X_train_padded, y_train, X_test_padded, y_test, 16, MAX_VOCAB, config["laye

    cnt += 1
```

```
----------------------------------------------------------------------
Dense Model 1 with parameters embedding_dim=16 and layers=[16, 8]
----------------------------------------------------------------------
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7573 - loss: 0.7517
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision    recall  f1-score   support

           0       0.78      0.83      0.80      1830
           1       0.76      0.70      0.73      1428

    accuracy                           0.77      3258
   macro avg       0.77      0.77      0.77      3258
weighted avg       0.77      0.77      0.77      3258

----------------------------------------------------------------------
Dense Model 2 with parameters embedding_dim=32 and layers=[32, 16]
----------------------------------------------------------------------
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7656 - loss: 0.7516
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision    recall  f1-score   support

           0       0.76      0.88      0.82      1830
           1       0.81      0.65      0.72      1428

    accuracy                           0.78      3258
   macro avg       0.79      0.76      0.77      3258
weighted avg       0.78      0.78      0.77      3258

----------------------------------------------------------------------
Dense Model 3 with parameters embedding_dim=64 and layers=[64, 32, 16]
----------------------------------------------------------------------
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7476 - loss: 1.1117
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision    recall  f1-score   support

           0       0.77      0.83      0.80      1830
           1       0.76      0.68      0.71      1428

    accuracy                           0.76      3258
   macro avg       0.76      0.75      0.76      3258
weighted avg       0.76      0.76      0.76      3258
```

```
In [159…
# BUILD MODEL
model = Sequential([
    Embedding(input_dim=MAX_VOCAB, output_dim=16),
    Flatten(),
    Dense(16, activation='relu'),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification uses sigmoid activation
])

# COMPILE MODEL
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# TRAIN MODEL
model.fit(X_train_padded, y_train, validation_data=(X_test_padded, y_test), epochs=10, batch_size=3

# EVALUATE MODEL
test_loss, test_acc = model.evaluate(X_test_padded, y_test)
print(f"Test Accuracy: {test_acc:.4f}")
```

```
Epoch 1/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.5837 - loss: 0.6614 - val_accuracy: 0.7845 -
val_loss: 0.4876
Epoch 2/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8523 - loss: 0.3689 - val_accuracy: 0.8026 -
val_loss: 0.4618
Epoch 3/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9275 - loss: 0.2068 - val_accuracy: 0.7931 -
val_loss: 0.5404
Epoch 4/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9650 - loss: 0.1210 - val_accuracy: 0.7839 -
val_loss: 0.5755
Epoch 5/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9741 - loss: 0.0829 - val_accuracy: 0.7707 -
val_loss: 0.6257
Epoch 6/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9803 - loss: 0.0658 - val_accuracy: 0.7747 -
val_loss: 0.6511
Epoch 7/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9806 - loss: 0.0592 - val_accuracy: 0.7821 -
val_loss: 0.6760
Epoch 8/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9785 - loss: 0.0558 - val_accuracy: 0.7855 -
val_loss: 0.6977
Epoch 9/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9793 - loss: 0.0563 - val_accuracy: 0.7796 -
val_loss: 0.6847
Epoch 10/10
238/238 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9819 - loss: 0.0486 - val_accuracy: 0.7775 -
val_loss: 0.6954
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7635 - loss: 0.7465
Test Accuracy: 0.7775
```

```
In [161…
y_pred = (model.predict(X_test_padded) > 0.5).astype("int32")  # Convert probabilities to binary la
print(classification_report(y_test, y_pred))
```

```
102/102 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
              precision    recall  f1-score   support

           0       0.78      0.83      0.81      1638
           1       0.77      0.71      0.74      1420

    accuracy                           0.78      3258
   macro avg       0.78      0.77      0.77      3258
weighted avg       0.78      0.78      0.78      3258
```

In [ ]: