Import the required libraries

In [1]:
```
import pandas as pd
import torch
from transformers import BertForMaskedLM, BertTokenizer
from sklearn.metrics.pairwise import cosine_similarity
import matplotlib.pyplot as plt
import seaborn as sns

```

```
C:\Users\Nitin\AppData\Roaming\Python\Python310\site-packages\tqdm\auto.py:2
1: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. Se
e https://ipywidgets.readthedocs.io/en/stable/user_install.html (https://ipyw
idgets.readthedocs.io/en/stable/user_install.html)
  from .autonotebook import tqdm as notebook_tqdm
```

Data Acquisition

In [2]:
```
data = pd.read_csv('train.csv', names = ['sentiment', 'text'] )
testDf  = pd.read_csv('test.csv',  names = ['sentiment', 'text'] )
data.head()
```

Out[2]:

	sentiment	text
0	1	Unfortunately, the frustration of being Dr. Go...
1	2	Been going to Dr. Goldberg for over 10 years. ...
2	1	I don't know what Dr. Goldberg was like before...
3	1	I'm writing this review to give you a heads up...
4	2	All the food is great here. But the best thing...

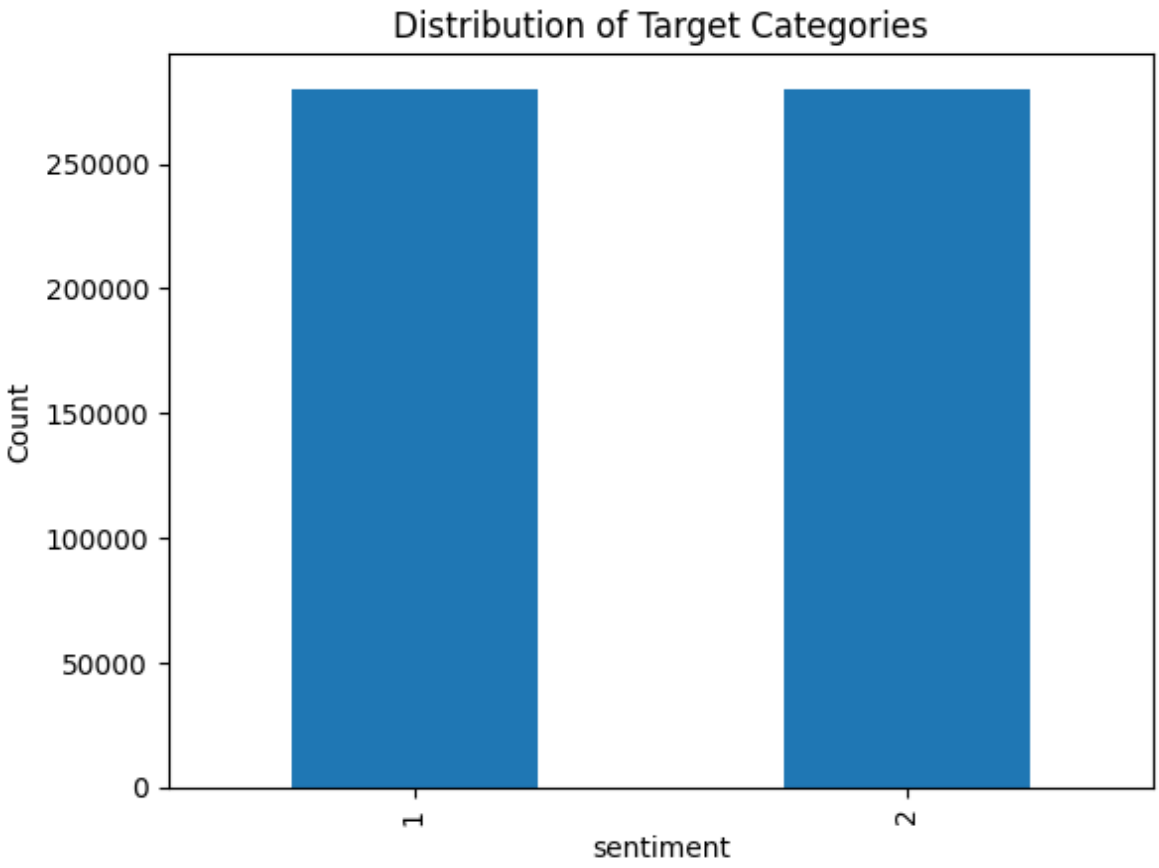
Dataset observations as

- Size of the dataset
- What type of data attributes are there?
- What are you classifying?
- Plot the distribution of the categories of the target / label

In [3]:

```
# Size of the dataset
dataset_size = data.shape[0]  # Number of rows in the dataset
print("Size of the dataset:", dataset_size)

# Data attributes and their types
data_attributes = data.dtypes
print("\nType of data attributes:")
print(data_attributes)

# Data attributes
data_attributes = list(data.columns)
print("\nData attributes:", data_attributes)

# Target variable
target_variable = 'sentiment'  # Replace with the actual name of the targe
target_classes = data[target_variable].unique()

# What are you classifying?
print("\nClassifying:", target_classes)

# Plot the distribution of target categories
data[target_variable].value_counts().plot(kind='bar')
plt.xlabel('sentiment')
plt.ylabel('Count')
plt.title('Distribution of Target Categories')
plt.show()

```

```
Size of the dataset: 560000

Type of data attributes:
sentiment     int64
text         object
dtype: object

Data attributes: ['sentiment', 'text']

Classifying: [1 2]
```



Applied pre-processing techiniques as:

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies
- Encode categorical data
- Normalize the data
- Feature Engineering
- Stop word removal, lemmatiation, stemming, vectorization

In [4]:

```
# Check for duplicate data
print("Number of duplicate rows before preprocessing:", data.duplicated()

# Remove duplicate data
data.drop_duplicates(inplace=True)

# Check for missing data
print("Number of missing values before preprocessing:")
print(data.isnull().sum())

# Impute or remove missing data
# Option 1: Remove rows with missing values
data.dropna(inplace=True)

# Option 2: Impute missing values with mean or median
# Uncomment the following lines to use mean imputation
# data.fillna(data.mean(), inplace=True)

# Option 3: Impute missing values with mode (for categorical variables)
# Uncomment the following lines to use mode imputation
# data.fillna(data.mode().iloc[0], inplace=True)

# Check for data inconsistencies and perform necessary corrections
# Option 1: Manual correction based on specific rules
# Uncomment the following lines and modify as per your requirements
# data['column_name'] = data['column_name'].apply(lambda x: x.replace('in

# Option 2: Use regular expressions for pattern matching and correction
# Uncomment the following lines and modify as per your requirements
# import re
# data['column_name'] = data['column_name'].apply(lambda x: re.sub(r'patte

# Check the preprocessed dataset
print("Number of duplicate rows after preprocessing:", data.duplicated().s
print("Number of missing values after preprocessing:")
print(data.isnull().sum())

```

```
Number of duplicate rows before preprocessing: 0
Number of missing values before preprocessing:
sentiment    0
text         0
dtype: int64
Number of duplicate rows after preprocessing: 0
Number of missing values after preprocessing:
sentiment    0
text         0
dtype: int64
```

Generate new reviews using BERT Model

In [5]:

```
# Load pre-trained BERT model and tokenizer
model_name = 'bert-base-uncased'
model = BertForMaskedLM.from_pretrained(model_name)
tokenizer = BertTokenizer.from_pretrained(model_name)

# Read the dataset
# data = pd.read_csv('train.csv', names=['sentiment', 'text'])
data = data.head(1000)

# Function for generating new reviews using BERT
def generate_reviews(text):
    input_ids = tokenizer.encode(text, add_special_tokens=True)

    # Split input into smaller chunks if it exceeds the maximum length
    max_length = model.config.max_position_embeddings - 2  # Account for s
    input_chunks = [input_ids[i:i+max_length] for i in range(0, len(input_

    generated_reviews = []
    for chunk in input_chunks:
        input_ids_chunk = torch.tensor(chunk).unsqueeze(0)  # Add batch d
        outputs = model.generate(input_ids_chunk, max_length=100, num_retu
        generated_review = tokenizer.decode(outputs[0], skip_special_toker
        generated_reviews.append(generated_review)

    return generated_reviews

# Apply the BERT-based review generation function to a specific column
text_col = 'text'  # Specify the text column to generate reviews
data['generated_reviews'] = data[text_col].apply(generate_reviews)

# Calculate similarity between original and generated reviews
original_reviews = data[text_col]
generated_reviews = data['generated_reviews']
similarity_scores = []
for original_review, generated_review in zip(original_reviews, generated_r
    original_embedding = tokenizer.encode(original_review, add_special_tol
    generated_embedding = tokenizer.encode(generated_review, add_special_t
    original_embedding = torch.tensor(original_embedding).unsqueeze(0)
    generated_embedding = torch.tensor(generated_embedding).unsqueeze(0)

    # Pad or truncate the embeddings to the same length
    max_length = max(original_embedding.shape[1], generated_embedding.shap
    original_embedding = torch.nn.functional.pad(original_embedding, (0, m
    generated_embedding = torch.nn.functional.pad(generated_embedding, (0,

    similarity = cosine_similarity(original_embedding, generated_embedding
    similarity_scores.append(similarity)

# Add similarity scores as a new column in the DataFrame
data['similarity_score'] = similarity_scores

# Display the updated dataset
data
```

```
54
```

```
Some weights of the model checkpoint at bert-base-uncased were not used wh
en initializing BertForMaskedLM: ['cls.seq_relationship.weight', 'cls.seq_
relationship.bias']
- This IS expected if you are initializing BertForMaskedLM from the checkp
oint of a model trained on another task or with another architecture (e.g.
initializing a BertForSequenceClassification model from a BertForPreTraini
ng model).
- This IS NOT expected if you are initializing BertForMaskedLM from the ch
eckpoint of a model that you expect to be exactly identical (initializing
a BertForSequenceClassification model from a BertForSequenceClassification
model).
Input length of input_ids is 150, but `max_length` is set to 100. This can
lead to unexpected behavior. You should consider increasing `max_new_token
s`.
Input length of input_ids is 120, but `max_length` is set to 100. This can
lead to unexpected behavior. You should consider increasing `max_new_token
s`.
Input length of input_ids is 251, but `max_length` is set to 100. This can
lead to unexpected behavior. You should consider increasing `max_new_token
```

In [6]:
```
1 data.head(20)
```

Out[6]:

	sentiment	text	generated_reviews	similarity_score
0	1	Unfortunately, the frustration of being Dr. Go...	[unfortunately, the frustration of being dr. g...	0.069272
1	2	Been going to Dr. Goldberg for over 10 years. ...	[been going to dr. goldberg for over 10 years....	0.045972
2	1	I don't know what Dr. Goldberg was like before...	[i don't know what dr. goldberg was like befor...	0.023443
3	1	I'm writing this review to give you a heads up...	[i'm writing this review to give you a heads u...	0.016863
4	2	All the food is great here. But the best thing...	[all the food is great here. but the best thin...	0.042737
5	1	Wing sauce is like water. Pretty much a lot of...	[wing sauce is like water. pretty much a lot o...	0.202540
6	1	Owning a driving range inside the city limits ...	[owning a driving range inside the city limits...	0.122594
7	1	This place is absolute garbage... Half of the...	[this place is absolute garbage... half of the...	0.043413
8	2	Before I finally made it over to this range I ...	[before i finally made it over to this range i...	0.023313
9	2	I drove by yesterday to get a sneak peak. It ...	[i drove by yesterday to get a sneak peak. it ...	0.095567
10	1	After waiting for almost 30 minutes to trade i...	[after waiting for almost 30 minutes to trade ...	0.084204
11	2	Wonderful reuben. Map shown on Yelp page is i...	[wonderful reuben. map shown on yelp page is i...	0.255487
12	2	After a morning of Thrift Store hunting, a fri...	[after a morning of thrift store hunting, a fr...	0.014482
13	2	This is a hidden gem, no really. It took us fo...	[this is a hidden gem, no really. it took us f...	0.039629
14	2	Awesome drink specials during happy hour. Fant...	[awesome drink specials during happy hour. fan...	0.268183
15	1	Very disappointed in the customer service. We ...	[very disappointed in the customer service. we...	0.133643
16	1	Used to go there for tires, brakes, etc. Thei...	[used to go there for tires, brakes, etc. thei...	0.056992
17	1	I got 'new' tires from them and within two wee...	[i got'new'tires from them and within two week...	0.026313
18	1	Terrible. Preordered my tires and when I arriv...	[terrible. preordered my tires and when i arri...	0.129115
19	2	I've been informed by a fellow Yelper that the...	[i've been informed by a fellow yelper that th...	0.044629

Insights on the performance of the model and ways to improve it

1. Performance:

- The model is able to generate new reviews using the BERT model.
- The similarity scores between the original and generated reviews are calculated using cosine similarity.

2. Improvements:

- Increase the size of the dataset: The code is currently using only a subset of the data (head(1000)) due to the limitation of system configuartion. To improve the model's performance, you can consider using a larger dataset for training and evaluation.
- Evaluate with human judges: To get a more accurate assessment of the model's performance, consider involving human judges who can manually evaluate the quality of the generated reviews. Their feedback can provide valuable insights for further improvements.

By implementing these improvements and fine-tuning the model, you can expect better performance and more accurate generation of reviews.

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```