```
# Install the necessary libraries if they weren't already installed
!pip install pandas transformers scipy

# Install necessary libraries:
import pandas as pd
import numpy as np
from transformers import pipeline, DistilBertTokenizerFast

!python --version

# For Sentiment Analysis:
# prefix the command with ! to run it as a shell command in Jupyter Notebook
#!pip install transformers
#!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117
#!pip install tensorflow

# Install PyTorch:
#!pip install torch torchvision torchaudio transformers pandas --no-cache-dir
#!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117
#!pip install torch torchvision torchaudio --no-cache-dir
#!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu --no-cache-dir
!pip install torch torchvision torchaudio --no-cache-dir
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.47.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (1.13.1)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.27.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.24.0->transformers) (2024.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.24.0->transforme
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.12.14)
Python 3.10.12
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.20.1+cu121)
Requirement already satisfied: torchaudio in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (11.0.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
```

What is the sentiment of the reviews, sorted by language?

We selected the Amazon Reviews Multilingual Dataset as our dataset: https://www.kaggle.com/datasets/mexwell/amazon-reviews-multi/data.
It depicts a comprehensive collection of multilingual product reviews in CSV format. This dataset contains about 1.3 million samples in 6 languages (DE = German, EN = English, ES = Spanish, FR = French, JA = Japanese, ZH = Chinese) with the following features:

review_id: A string identifier of the review.

product_id: A string identifier of the product being reviewed.

reviewer_id: A string identifier of the reviewer.

stars: An int between 1-5 indicating the number of stars.

review_body: The text body of the review.

review_title: The text title of the review.

language: The string identifier of the review language.

product_category: String representation of the product's category.

The data will be directly downloaded and processed from the CSV files provided by the dataset. During preprocessing we will focus on removing stop words as well as choosing the correct embedding for the reviews.

Splitting:

The data was split into sentiment labels based on a star rating, where:

1 (Positive): stars > 3

0 (Negative): stars <= 3

4) Prepare Files

Split the big csv-file into smaller files according to the language:

zh (Chinese) == Simplified Chinese / Mandarin Chinese

```
# # filter df for Chinese reviews
# chinese_reviews = df[df['language'] == 'zh']

# # save them to a new csv file
# output_file = 'reviews_zh.csv'
# chinese_reviews.to_csv(output_file, index=False)

# print(f"Saved Chinese reviews to {output_file}")
```

5) Sentiment Analysis

BERT can only read comments up to 512 chars so longer comments need to be tokenized:

zh (Chinese)

```
from transformers import AutoTokenizer, pipeline
import pandas as pd
from sklearn.metrics import accuracy_score  # For accuracy calculation

# load csv file
file_path = 'reviews_zh.csv'
df = pd.read_csv(file_path)
chinese_reviews = df[df['language'] == 'zh'] # filter for Chinese

# define sentiment label function based on stars
def sentiment_label(stars):
    return 1 if stars > 3 else 0

# apply sentiment label function to create the true_label column
chinese_reviews['true_label'] = chinese_reviews['stars'].apply(sentiment_label)

# init tokenizer
#tokenizer = AutoTokenizer.from_pretrained("bert-base-chinese")  # A lightweight Chinese model
#tokenizer = AutoTokenizer.from_pretrained("uer/roberta-base-finetuned-dianping-chinese")

# init sentiment analysis pipeline & tokenizer:

# tokenizer = AutoTokenizer.from_pretrained("uer/roberta-base-finetuned-chinanews-chinese")
#sentiment_pipeline = pipeline("sentiment-analysis", model="uer/roberta-base-finetuned-chinanews-chinese", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("bert-base-chinese")
# sentiment_pipeline = pipeline("sentiment-analysis", model="bert-base-chinese", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("uer/roberta-base-finetuned-dianping-chinese")
# sentiment_pipeline = pipeline("sentiment-analysis", model="uer/roberta-base-finetuned-dianping-chinese", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("uer/chinese-roberta-wwm-ext")
# sentiment_pipeline = pipeline("sentiment-analysis", model="uer/chinese-roberta-wwm-ext", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("hfl/chinese-bert-wwm")
# sentiment_pipeline = pipeline("sentiment-analysis", model="hfl/chinese-bert-wwm", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("brightmart/albert_zh")
# sentiment_pipeline = pipeline("sentiment-analysis", model="brightmart/albert_zh", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("uer/roberta-large-finetuned-dianping-chinese")
# sentiment_pipeline = pipeline("sentiment-analysis", model="uer/roberta-large-finetuned-dianping-chinese", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("hfl/chinese-roberta-wwm-ext-large")
# sentiment_pipeline = pipeline("sentiment-analysis", model="hfl/chinese-roberta-wwm-ext-large", framework="pt")

# tokenizer = AutoTokenizer.from_pretrained("uer/chinese-roberta-base")
# sentiment_pipeline = pipeline("sentiment-analysis", model="uer/chinese-roberta-base", framework="pt")

tokenizer = AutoTokenizer.from_pretrained("IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment")
sentiment_pipeline = pipeline("sentiment-analysis", model="IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment", framework="pt")

# function to truncate reviews to a maximum length of 512 tokens using the tokenizer (BERT limitation)
def truncate_review(review, max_length=512):
    if not isinstance(review, str):
```

```
    if not isinstance(review, str):
        return ""
    tokens = tokenizer(review, truncation=True, max_length=max_length, return_tensors='pt')['input_ids'][0]
    return tokenizer.decode(tokens, skip_special_tokens=True)

# apply truncation to review body column
chinese_reviews['review_body'] = chinese_reviews['review_body'].apply(truncate_review)

# sentiment analysis of review column
sentiments = sentiment_pipeline(chinese_reviews['review_body'].tolist(), batch_size=32)  # Use batching for speed

# convert predicted labels to binary format (0 = negative, 1 = positive)
chinese_reviews['predicted_label'] = [1 if s['label'].lower() == 'positive' else 0 for s in sentiments]

# accuracy
accuracy = accuracy_score(chinese_reviews['true_label'], chinese_reviews['predicted_label'])
print(f"Accuracy: {accuracy * 100:.2f}%")

# Save results
chinese_reviews.to_csv('chinese_reviews_with_sentiments_and_accuracy.csv', index=False)

print("Sentiment analysis complete and saved to 'chinese_reviews_with_sentiments_and_accuracy.csv'.")
```

```
config.json: 100%  785/785 [00:00<00:00, 28.3kB/s]
vocab.txt: 100%  110k/110k [00:00<00:00, 1.60MB/s]
pytorch_model.bin: 100%  409M/409M [00:02<00:00, 171MB/s]
Device set to use cuda:0
Accuracy: 82.72%
Sentiment analysis complete and saved to 'chinese_reviews_with_sentiments_and_accuracy.csv'.
```

6) Accuracy

```
# Load the CSV
result_file_path = 'chinese_reviews_with_sentiments_and_accuracy.csv'
result_df = pd.read_csv(result_file_path)

# recalculate accuracy to ensure correctness
overall_accuracy = accuracy_score(result_df['true_label'], result_df['predicted_label'])

# print average accuracy
print(f"Overall Average Accuracy: {overall_accuracy * 100:.2f}%")
```

```
Overall Average Accuracy: 82.72%
```