032-02-Pretrained-models-solution

April 18, 2024

1 03-02 - Pretrained-models - Solution Notebook

- Written by Alexandre Gazagnes
- Last update: 2024-02-01

1.1 About

Context :

Let's Continue the Party!

Data :

You can find the dataset here.

1.2 Preliminaries

1.2.1 System

These commands will display the system information:

Uncomment theses lines if needed.

```
[ ]: # pwd
```

```
[ ]: # cd ..
```

```
[ ]: # ls
```

1.2.2 Import

```
[ ]: # import os, sys, warnings, secrets, datetime
     # import pickle

     from IPython.display import display
     import zipfile
```

```
[ ]: import pandas as pd

     # import numpy as np
```

[ ]:
```
import plotly.px as px
```

[ ]:
```
import tensorflow as tf
```

[ ]:
```
import transformers

from transformers import pipeline, set_seed

from transformers import T5Tokenizer, T5ForConditionalGeneration


from transformers import BertTokenizer, BertForTokenClassification
from transformers import BertTokenizer, BertForSequenceClassification

from transformers import DistilBertTokenizer,␣
 ↪DistilBertForSequenceClassification

from transformers import RobertaTokenizer, RobertaForSequenceClassification
from transformers import RobertaTokenizer, RobertaForMaskedLM

from transformers import BartTokenizer, BartForConditionalGeneration

from transformers import MarianMTModel, MarianTokenizer

from transformers import GPT2LMHeadModel, GPT2Tokenizer, pipeline
```

1.2.3 Third party tools

Set the seed :

[ ]:
```
set_seed(42)
```

Download the default classifier :

[ ]:
```
classifier = pipeline("sentiment-analysis")
```

Specifying a model :

[ ]:
```
roberta_sentiment = pipeline(
    "sentiment-analysis",
    model="cardiffnlp/twitter-roberta-base-sentiment-latest",
)
```

Question answering model :

[ ]:
```
question_answerer = pipeline("question-answering")
```

Text Generator :

[ ]:
```
gpt2_generator = pipeline("text-generation", model="gpt2")
```

```
[ ]: # bloom = pipeline("text-generation", model="bigscience/bloom-7b1")
```

1.2.4 Data

Download the dataset :

```
[ ]: !wget https://www.kaggle.com/datasets/shoumikdhar/
      ↪amazon-food-reviews-100k-datasets
```

Load .zip file

```
[ ]: with zipfile.ZipFile("archive.zip", "r") as zip_ref:
         zip_ref.extractall("archive")
         extracted_file = zip_ref.namelist()[0]
         df = pd.read_csv(f"archive/{extracted_file}")
```

1.3 Data Exploration

Head :

```
[ ]: df.head()
```

Tail :

```
[ ]: df.tail()
```

Sample :

```
[ ]: df.sample(10)
```

Split the text (but not with official tokenizer) :

```
[ ]: df["pseudo_token"] = df.Review.apply(lambda x: x.split())
     df
```

Describe :

```
[ ]: df["n_psuedo_token"] = df.pseudo_token.apply(len).describe().round(2)
     df.n_psuedo_token.describe()
```

Length of each doc :

```
[ ]: df["_len"] = df.Review.str.len().describe()
     df
```

Describe :

```
[ ]: df.Rating.describe().round(2)
```

1.4 High Level Implementation

1.4.1 Classification & Sentiment Analysis

Use a classifier :

```
[ ]: classifier("AI stuff is real hard to understand.")
```

```
[ ]: classifier("AI stuff is real hard to understand.", top_k=3)
```

```
[ ]: classifier("AI stuff is so fun")
```

```
[ ]: classifier("can you say me if AI is good or not...")
```

Apply on a column :

```
[ ]: results = df.Review.head().apply(classifier)
     results
```

Results :

```
[ ]: results.explode()
```

```
[ ]: results.apply(pd.Series)
     results
```

Join Both :

```
[ ]: df.head().join(results)
```

using another tool :

```
[ ]: roberta_sentiment("AI stuff is real hard to understand.")
```

```
[ ]: roberta_sentiment("AI stuff is so fun")
```

```
[ ]: roberta_sentiment("can you say me if AI is good or not...")
```

```
[ ]: results = df.Review.head().apply(roberta_sentiment).explode().apply(pd.Series)
     results
```

check this blog for more infomation: Getting Started with Sentiment Analysis using Python

1.4.2 Information Extraction & Questing Answering

```
[ ]: txt = "hello, i am a 40 years old guy liking in san francisco with my dog and␣
       ↪my guitar. I want to learn how to code, can you help me ?"

     out = question_answerer(question="are old am i ? ", context=txt, top_k=10)
```

[ ]:
```
out = pd.DataFrame(out)
out
```

[ ]:
```
out.score.sum()
```

[ ]:
```
out["_cumsum"] = out.score.cumsum()
out
```

[ ]:
```
threshold = 0.75

clean_out = out.loc[out._cumsum < threshold]
clean_out
```

[ ]:
```
answers = clean_out.answer.tolist()
answers
```

[ ]:
```
question_answerer(question="what is the product?", context=df.Review.values[4])
```

[ ]:
```
qa_model = pipeline("question-answering")
question = "Where do I live?"
context = "My name is Merve and I live in İstanbul."
qa_model(question=question, context=context, top_k=3)
## {'answer': 'İstanbul', 'end': 39, 'score': 0.953, 'start': 31}
```

1.4.3 Text Generation & Prompting

[ ]:
```
gpt2_generator("Hello, I'm an NLP student,", max_length=30,␣
  ↪num_return_sequences=5)
```

[ ]:
```
out = gpt2_generator(
    "Hello, I'm an computer science student,", max_length=30,␣
  ↪num_return_sequences=5
)
out
```

[ ]:
```
for dd in out:
    print(dd["generated_text"])
```

[ ]:
```
out = gpt2_generator(
    "Hello, I'm an computer science student,", max_length=100,␣
  ↪num_return_sequences=10
)
out
```

[ ]:
```
for dd in out:
    print(dd["generated_text"])
```

[ ]:
```
# with the open source Bloom model https://huggingface.co/bigscience/bloom
```

[ ]:
```
```

[ ]:
```
```

1.4.4 Translation

[ ]:
```
# UP TO YOU TO FIND IT
```

[ ]:
```
```

1.4.5 Summarization

[ ]:
```
# UP TO YOU TO FIND IT
```

[ ]:
```
```

[ ]:
```
```

1.5 Specific Implementation

1.5.1 Sentiment

Load pre-trained model and tokenizer

[ ]:
```
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertForSequenceClassification.from_pretrained("bert-base-uncased")
```

Sentiment analysis pipeline

[ ]:
```
nlp = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)
```

Example text :

[ ]:
```
result = nlp("I love learning about data science with Transformers!")
print(result)
```

1.5.2 NER

Load pre-trained model and tokenizer

[ ]:
```
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertForTokenClassification.from_pretrained("bert-base-uncased")
```

NER pipeline

[ ]:
```
nlp = pipeline("ner", model=model, tokenizer=tokenizer)
```

Example text

```
[ ]: result = nlp("Hugging Face is a technology company based in New York")
     print(result)
```

1.5.3 Text-generation

```
[ ]: tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
     model = GPT2LMHeadModel.from_pretrained("gpt2")
```

Text generation pipeline

```
[ ]: text_generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

Generate text

```
[ ]: print(text_generator("Artificial intelligence is", max_length=50))
```

```
[ ]: tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
     model = GPT2LMHeadModel.from_pretrained("gpt2")
```

```
[ ]: text_generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

```
[ ]: # Generate text
     print(text_generator("Artificial intelligence is", max_length=50))
```

Assuming the same model and tokenizer loaded from the previous example Simulate a chatbot response

```
[ ]: chat_input = "Hello, how can I assist you today?"
     chat_response = text_generator(chat_input, max_length=50)
```

```
[ ]: print(chat_response)
```

1.5.4 Filled Masked

Load tokenizer and model

```
[ ]: tokenizer = RobertaTokenizer.from_pretrained("")
     model = RobertaForMaskedLM.from_pretrained("roberta-base")
```

Fill-mask pipeline

```
[ ]: fill_mask = pipeline("fill-mask", model="roberta-base")
```

Example

```
[ ]: print(fill_mask("The weather today is <mask>."))
```

1.5.5 ...

```
[ ]:
```

```
[ ]:
```