DOCUMENT: SENTIMENT ANALYSIS FOR MARKETING INNOVATION

Fine Tuning Pre-trained Model for Sentiment Analysis

Introduction

In this tutorial, I will be fine-tuning a Roberta model for the **Sentiment Analysis** problem.

The flow of the notebook

The notebook will be divided into separate sections to provide an organized walk-through of the process used. This process can be modified for individual use cases. The sections are:

1. Importing Python Libraries and preparing the environment
2. Importing and Pre-Processing the domain data
3. Preparing the Dataset and Data loader
4. fine-tuning
5. Fine Tuning the Model
6. Validating the Model Performance
7. Saving the model and artifacts for Inference in the Future

Technical Details

This script leverages multiple tools designed by other teams. Details of the tools used are below. Please ensure that these elements are present in your setup to successfully implement this script.

- Data: I will be using the dataset available at the Kaggle Competition
- I will be referring only to the first CSV file from the data dump: train's
- Language Model Used:
- The Roberta model was proposed in Roberta: A Robustly Optimized BERT Pretraining Approach by Yinan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Dangi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. It is based on Google's BERT model released in 2018.
- Blog-Post
- Research Paper
- Documentation for python
- Hardware Requirements:
- Python 3.6 and above
- Pytorch, Transformers, and All the stock Python ML Libraries
- GPU-enabled setup

Sentiment Analysis Application with Chat GPT and Druid

Like many in the developer community, I have tried Chat GPT from Open AI. Over my IT career, I have worked in many positions including as a data scientist/data engineer. So, as I did my assessment of Chat GPT, I thought of ways to use the technology in practice. I have done sentiment analysis before using custom algorithms that I wrote, specific NLP (Natural Language Processing) libraries, and low-code platforms like Weka, Rapid Miner, and Data Robot. Why not do something similar using Chat GPT and combine it with a real-time analytics database like Apache Druid?

There are many benefits to combining a trained, NLP model with Apache Druid for sentiment analysis. Modern models such as GPT-3 and GPT-4 are highly effective in understanding and

processing natural language. They can better identify nuances and context, resulting in more accurate results. Sentiment analysis often requires processing large volumes of data, such as social media posts, reviews, or customer feedback. And then, aggregating and analysing those sentiments at scale can reveal even more insights and identify patterns and trends – all crucial for businesses that need to react quickly to changes in customer sentiment.
As this blog will show, the integration is relatively easy using technologies that are publicly accessible.

How to Train a ChatGPT Model for Sentiment Analysis

Let's learn how to train a sentiment analysis model with ChatGPT! ChatGPT is a great choice for this task because it can understand and interpret human language accurately. Traditional sentiment analysis methods rely on keyword matching or rule-based systems, which can lead to incomplete or inaccurate results. But ChatGPT uses deep learning algorithms to analyze text at a deeper level, considering not only individual words but also the context in which they are used. So, let's follow these six steps to train ChatGPT with a sentiment analysis model.

Step 1: Data Collection: First, you need a large dataset of text data containing sentiment. This could be customer reviews, social media posts, or any other type of text that expresses sentiment. You can collect data from popular review sites like Yelp, Amazon, or TripAdvisor.

Step 2: Data Preprocessing: Once you have your dataset, it's time to preprocess it. This means cleaning and formatting the data so that it can be used to train the ChatGPT model. For instance, you can remove stop words, punctuation, and numbers and convert the text into a numerical format using one-hot encoding or word embeddings.

Step 3: Labeling Data: To train a supervised machine learning model, you need to label the data based on the sentiment it expresses. You can assign a binary label to each piece of text data, with positive sentiment labeled as 1 and negative sentiment labeled as 0. For example, you can label a positive review of a restaurant as 1 and a negative review as 0.

Step 4: Training the Model: There are 2 approaches to train LLMs to answer questions –

1. Model fine-tuning, and
2. Context injection.