# **YOUTUBE SENTIMENTAL ANALYSIS**
INITIAL DELIVERABLE

POL COSTA CASTELLS
DANIEL GONZÁLBEZ BIOSCA
POL VILAVELLA DILOY
EVA BRIGITTE FERNÁNDEZ ROMERO
GCED
Universitat Politècnica de Catalunya
`https://github.com/polvilavella/taed-youtube`

## INTRODUCTION

Our project is based on sentiment analysis of youtube comments. What it means sentiment analysis? Sentiment analysis is a Natural Language processing problem that identifies, extracts, quantifies, and study affective states and subjective information. So our project is based on extracting youtube comments and analyze them to classify each one in three different categories, a youtube comment can be positive, negative or neutral. An example of a positive comment would be: "Nice video, I love this type of content.", on the other hand a negative comment could be such as: "Listening to this song makes my ears bleed.". Probably the less found type of comments would be the neutral ones because mostly people shows their good opinions or their bad opinions, so to find and categorize neutral comments it's more difficult that the other two categories. An example for a neutral comment could be: "Where can I buy this device?".

So what is the goal of our project? We will use a model that will have to predict correctly the sentiment of the youtube comments, as the data that we have obtained has each comment tagged with it's sentiment it will be easy for us to obtain an accuracy, as we will have to count the number of correct predictions. We think that an 80% accuracy is acceptable for our project because we have the neutral sentiment that is difficult to categorize because it can be easily tagged as a positive comment.

In our dataset card you can find information about the dataset we are using for the problem. Some information you can find is the language which is written in the data, the structure that our data follows and the fields that it has. There is information about the social impact, personal and sensitive information, limitations and the creator of the dataset that people who want to use this dataset must know.

In addition we have also made a model card where you can find some details about the model that we are using to solve the problem that we have proposed. It's shown how to use the model, the training procedure that you have to follow for the model working properly, also it's shown the description of the model, limitations and intended uses. All this information must be followed if you want the model to solve the problem that you have with good accuracy.

In the first milestone firstly we choose the dataset for the ML problem, we decided first what type of problem we wanted to solve. As we found interesting sentiment analysis we started searching datasets for solving this problem, when we found the youtube statistics and comments dataset we thought it would fit properly into our idea and we started working with it. Then we defined the dataset card and model card of our dataset. To create these cards we relied on Hugging face examples and we followed it's structure. Our collaborative working space it consists on a GitHub repository and a discord server. GitHub is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git and other functionalities. Discord is an instant messaging social platform where users have the ability to communicate with voice calls, video calls, text messaging, media and files in private chats or as part of communities called "servers". We have created our own server where we have different text channels, in each channel the functionality is different, we have a general chat but also chats where we talk about the data or the model that we are deploying, so it's easier to find the messages when you have any trouble. Lastly on the first milestone we decided the cloud provider that we will use in the future to deploy our ML component, we decided to use the FIB virtual machine "Virtech".

Afterwards in the second milestone we were supposed to try the reproducibility of our problem. To do it we should create a project structure, we followed the cookiecutter data science template to create our project structure, as it can be seen in our GitHub repository which follows this structure. Then we are supposed to track our data and code versions, to do this we are using GitHub which it has the version control using Git so it works properly. Lastly to follow the experiment tracking we are using MLflow which it provides us the information of the previous experiments and we can see how the accuracy of our solution is upgrading on every change that we make on our code.

## METHODOLOGY

In out methodology we use different software engineering practice. A software engineering practice is is a broad array of concepts, principles, methods, and tools that you must consider as software is planned and developed. In our case we use GitHub, MLflow, cloud provider and Discord.

Github is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git and other functionalities. It's main function is the version control using Git, but what is Git? Git is free and open source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. So using GitHub we have the version control and also all the students from the group can access any time to the different versions of the code.

MLflow is a platform to streamline machine learning development, including tracking experiments, packaging code into reproducible runs, and sharing and deploying models. So how we use MLflow in our problem solving model? We want to keep track of our experiments on every step that we change on our model, so we use this tool for tracking the accuracy of each step we make.

Discord is an instant messaging social platform where users have the ability to communicate with voice calls, video calls, text messaging, media and files in private chats or as part of communities called "servers". So to contact between the different group members we used this tool because it provides us different ways of communication between us such as voice channels and different text channels which every different chat has it's own function, we have a chat to talk about the data stuff, one chat to discuss about the changes that we have made on our model, etc.

The cloud provider is used to do cloud computing which is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. This means that the computation of our model is not done locally at our host terminal so it's computed on a server found outside the local host.

## TECHNICAL DESCRIPTION

The aim of this section is to describe the architecture of the project, which is composed by many parts:

Virtech cloud provider:

The cloud provider we have decided to use in the Milestone 4 is virtech, which is a service provided by the university. We have chosen it because we have checked how it works and it is easy to use. We have made some tests trying to send files from our local machines to the virtual machine created in the cloud via ssh and it worked really well. The fact that is free is, of course, another relevant point that we have taken into account.

Github:

We have created a github repository (link on the cover of this document), where the model and dataset cards, as well as the current python script of our model can be found.

Github helps us to keep track of our work and analyse the changes we have made in our project. As we are in a collaborative project, we can review each other's code and suggest changes that could be beneficial to our project.

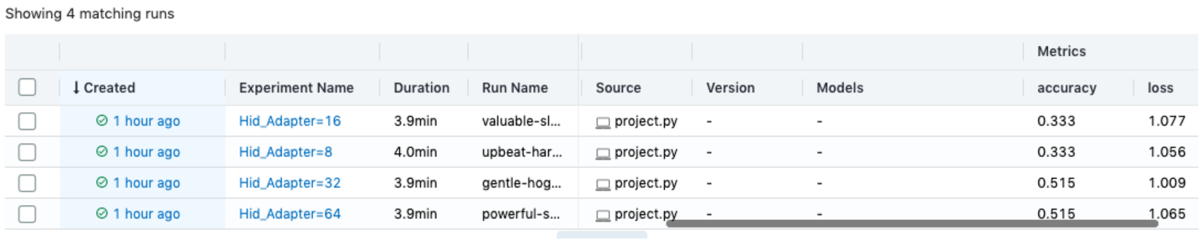
MlFlow:

In order to register the different experiments we do in our project, we have used MlFlow, which is a platform that helps to keep track of the desired metrics, as well as the value of the specified parameters. In our case, we have focused on the value of the validation loss (Cross Entropy Loss) during training and the validation accuracy, as we are solving a classification problem. At the moment we have done 4 experiments, changing the hidden size (8, 16, 32 and 64) of the adapter block of our model.

We have used very few data samples in order to have a first view of how the platform works and the experiments are registered. This is why the results are quite poor. We have registered, for each, experiment the value of these two metrics for each epoch of training.

The results are shown in the following way:

An overall view of the different experiments:



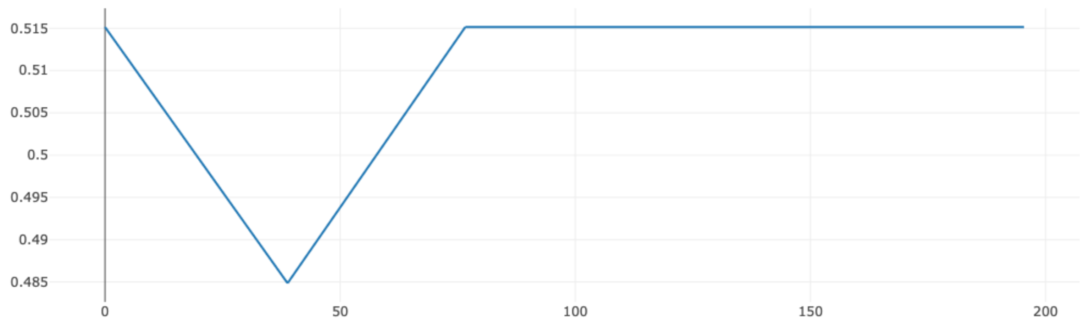
Showing 4 matching runs

| | | | | | | | | Metrics | |
|---|---|---|---|---|---|---|---|---|---|
| | ↓ Created | Experiment Name | Duration | Run Name | Source | Version | Models | accuracy | loss |
| ☐ | 1 hour ago | Hid_Adapter=16 | 3.9min | valuable-sl... | project.py | - | - | 0.333 | 1.077 |
| ☐ | 1 hour ago | Hid_Adapter=8 | 4.0min | upbeat-har... | project.py | - | - | 0.333 | 1.056 |
| ☐ | 1 hour ago | Hid_Adapter=32 | 3.9min | gentle-hog... | project.py | - | - | 0.515 | 1.009 |
| ☐ | 1 hour ago | Hid_Adapter=64 | 3.9min | powerful-s... | project.py | - | - | 0.515 | 1.065 |

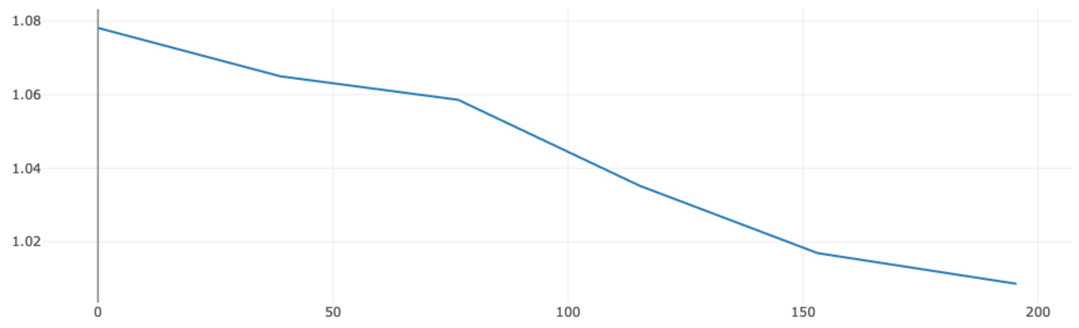
As we could expect, the results improve when the hidden size (and, because of that, the model complexity) increases, but it does not look very beneficial to increase it from 32 to 64. We will have to make further research, as we are training the model with very few data, but that could be a first idea to investigate with Mlflow.

Evolution through epochs of the experiment when the adapter has a hidden size of 32:

Evolution of the accuracy:



Evolution of the Cross Entropy Loss:



As we can see, the accuracy seems to converge, even though the cross entropy loss continues decreasing through the epochs.

## SELF-EVALUATION OF THE PROJECT

Starting this project has led us into facing many challenges. We have used tools that we had barely heard about them like experiment tracking with Mlflow.

One of the most important problems we have faced is the fact that we were used to execute our Machine Learning codes in a GPU offered by Kaggle. To track the experiments using Mlflow we have executed our code with our local CPU, which made the execution much slower. We would like to change this in order to execute the code in the virtual machine we have already created or in Kaggle, but we have had problems to connect it to the Mlflow server that tracks the experiments.

Another problem we will have to solve has to do with memory usage. We have used gradient accumulation so that, even though the size of our batches is 1, our effective batch size is much bigger and computationally cheaper. To do the experiments we have shown, we have used less data than what we really have, but the goal is to use as much data as possible, as the results will obviously be greater and the algorithm will generalize better.

In order to start this project, we have used the knowledge we had on NLP, as we knew of the existence of the BERT model and its distilled version and the capacity it has to understand text. We have had to make some research to find a version of the DistilBERT model that understood multiple languages.

In conclusion, we have faced multiple problems. Some of them have already been solved and in the future days we would like not only to continue progressing on our project, but also to address some of the issues we exposed in the previous lines that need to be solved.