Installation guide for DEMOS dApp

Note: This tutorial was created for Windows 10 however the same procedures apply for Linux and MacOS Operating Systems. DEMOS dApp was created using Linux Ubuntu 20.04

Necessary Software you need to install before starting the tutorial:

* Google Account with Google Drive
* Python 3 (<https://nodejs.org/en/download/>)
* Node.js (<https://www.python.org/downloads/>)
* Truffle (<https://trufflesuite.com/ganache/>)
* Metamask Extension for Google Chrome (<https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn>)
* A Kaggle account (this is needed if you want to use a Kaggle dataset. If you want to follow this tutorial until the end you need to create an account. You can create it using your Google Account) (<https://www.kaggle.com/>)

1. Clone or Download the files from https://gitlab.com/netmode/blockchain-ml.git . Main directory folder should look like this:

Text

Description automatically generated with medium confidence

CHAPTER 1: INITIAL TRAINING

We start this tutorial with the initial training of the models we will use. You don’t strictly have to use these datasets or even models, after all DEMOS was created to support multiple models and datasets. However, we advise you to follow this tutorial as it is before experimenting. In this tutorial we will create 3 instances of the same model using these datasets:

* <https://www.kaggle.com/datasets/yasserh/twitter-tweets-sentiment-dataset>
* [https://www.kaggle.com/datasets/yasserh/imdb-movie-ratings-sentiment-analysis](https://www.kaggle.com/datasets/yasserh/imdb-movie-ratings-sentiment-analysis?select=movie.csv)
* <https://www.kaggle.com/datasets/ishantjuyal/emotions-in-text>

We use this model <https://huggingface.co/bert-base-uncased> and the training of the Neural Network is based on this <https://towardsdatascience.com/sentiment-analysis-in-10-minutes-with-bert-and-hugging-face-294e8a04b671> tutorial.

1. Open Google Drive, create a folder named “ml-blockchain” and upload the contents of the GoogleColab-backend folder

Graphical user interface, text, application, email

Description automatically generated

1. Login to Kaggle <https://www.kaggle.com/> and navigate to <https://www.kaggle.com/me/account>. Scroll down to find API.

Graphical user interface, text, application, email

Description automatically generated

1. Click *Expire API Token* to remove previous tokens. Then click *Create New API Token* to download kaggle.json file



1. Upload kaggle.json to Google Colab

Graphical user interface, text, application, email

Description automatically generated

1. Double click Twitter-BERT-InitialTrainModel.ipynb to open it in Goolge Colab
2. Scroll to find the first code cell under “Download Kaggle dataset” and ensure that the path to kaggle.json file is correct

Graphical user interface, application

Description automatically generated

1. Now it’s time to set parameters for training. You are free to change these parameters however you like.

Note: There is no perfect universal configuration for all datasets, try multiple combinations to see what fits your dataset better. Before changing values check <https://www.tensorflow.org/api_docs/python/tf/keras/Model#compile>

Graphical user interface, text, application

Description automatically generated

1. Change runtime type to GPU to accelerate training

Graphical user interface, application

Description automatically generated

1. Create a new folder called savedModels and ensure that the paths in the following cell are pointing to this folder

Text

Description automatically generated with medium confidence

It’s time to start the initial training! After initial training is finished, savedModels folder will contain the model, the validation set and the test set

1. Select Runtime and Run all, a window will pop to enable connection to Google Drive. Accept it and wait…
2. If you are not satisfied with the accuracy, set new parameters and try again

Note: It is important to **save the value of accuracy** for the models as you will use it later



1. Follow the same procedure for IMDB-BERT-InitialTrainModel.ipynb and Emotions-BERT-InitialTrainModel.ipynb
2. When you finish with the initial training for the 3 models the savedModels folder should look like this

Table

Description automatically generated

CHAPTER 2: GANACHE SETUP

Pretraining is done! Time to create a local Blockchain

1. Start Ganache. When this window pops up select *NEW WORKSPACE ETHEREUM*

Timeline

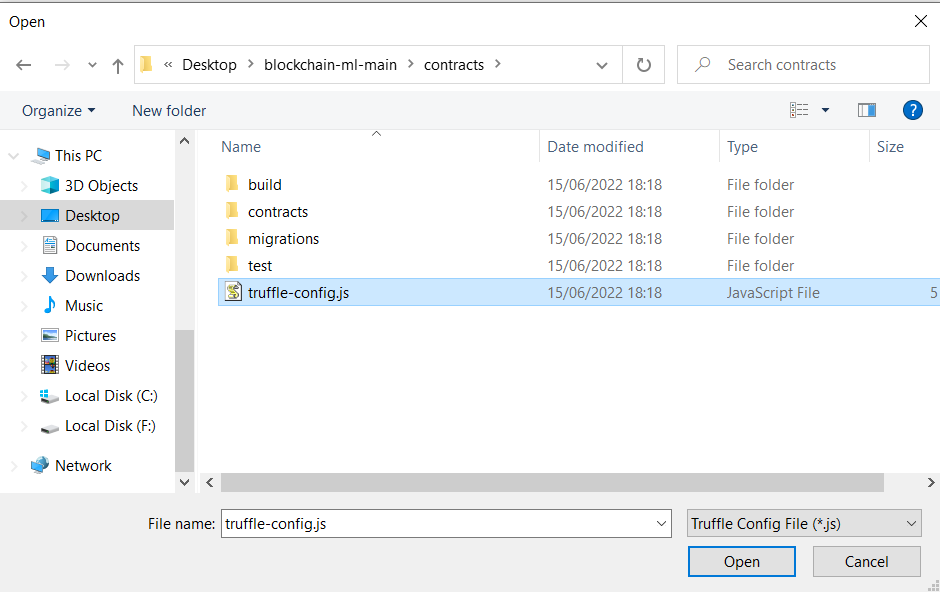
Description automatically generated

1. Change WORKSPACE-NAME to a name of your desire and then click *ADD PROJECT*

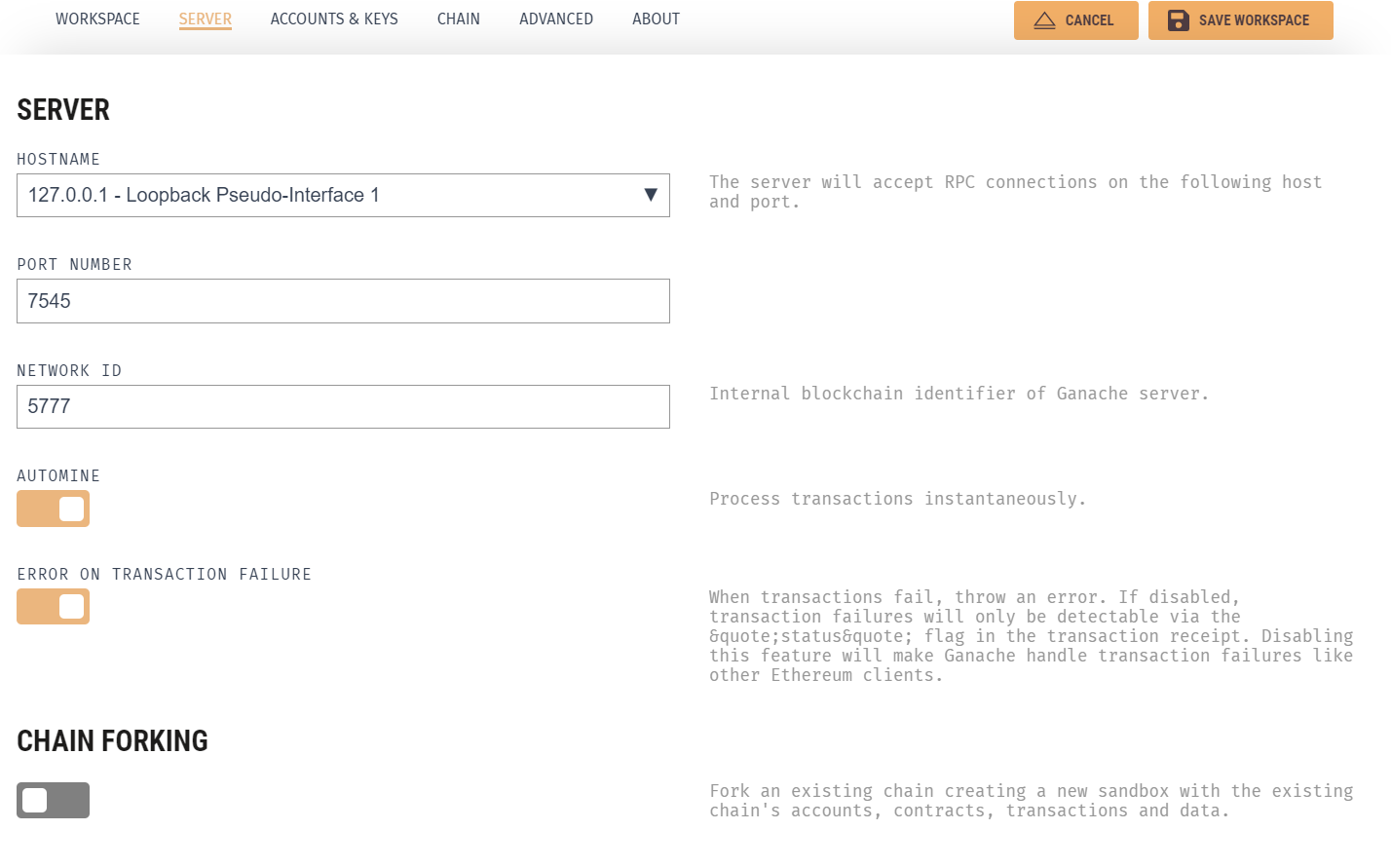
Graphical user interface, text, application, email

Description automatically generated

1. Select *truffle-config.js* file which is in *blockchain-ml-main/contracts* and click *Open*



1. Switch to *SERVER* tab and make sure the following configurations are set



1. Switch to ACCOUNTS & KEYS tab. Here you can change the starting balance and the number of accounts to be created. 10 Accounts starting with 100 ETH are sufficient for testing the DEMOS dApp, however, feel free to add more if you want.

Graphical user interface, text, application

Description automatically generated

Note 1: Remember that **these configurations cannot be changed** after the Blockchain is created. If you want to add more ETH or more accounts, you will need to create a new Blockchain

Note 2: Although not clearly stated, we advise you not to change the default Mnemonic if you don’t know what you are doing. This may cause problems with the connection to Metamask.

1. Let the rest of the configurations to default and click *SAVE WORKSPACE* to create the Blockchain. A new Ganache window will appear with a list of the newly created accounts

Table

Description automatically generated

Feel free to look around the options to view information about your newly created local Blockchain. If you click the CONTRACTS tab you can view the two Contracts which are not yet deployed and do not have an address.

Graphical user interface, application, website

Description automatically generated

Note: Always **Keep Ganache running** while running DEMOS

CHAPTER 3: FRONTEND SETUP

Local Blockchain is created! Let’s deploy the frontend

1. Start Visual Studio Code. Select *File > Open Folder…* and select the main directory folder as shown below. Then click *Select Folder*

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Select *Terminal > New Terminal* to start a new terminal

A screenshot of a computer

Description automatically generated with medium confidence

1. Download npm with command: *npm install -g npm*
2. Update npm with command: *npm install -g npm@latest*
3. Download Truffle with command: *npm install -g truffle*

Note: It seems like the latest version of Truffle has some glitches (16/06/2022). You can instead install a previous version using: *npm install -g* [*truffle@5.4.29*](mailto:truffle@5.4.29)

1. Ensure that Truffle was successfully downloaded with command: *truffle version*
2. Install truffle project dependencies with command: *truffle build* Text

   Description automatically generated
3. Deploy Contracts with command: truffle migrate –reset

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated



Note: You can now check Contracts tab in Ganache to see that state has changed to “DEPLOYED” and contracts have an address

Graphical user interface, text, application, email

Description automatically generated

1. Perform a basic test on CaptionReview contract’s function with command: *truffle test*

Text

Description automatically generated

1. Switch to CONTRACTS tab on Ganache, click CaptionReview and copy the address

Graphical user interface, text, application, email, website

Description automatically generated

1. Switch to Visual Studio Code and navigate to contract-address.js file located in .\blockchain-ml-main\client\constants\

Graphical user interface, text

Description automatically generated

1. Replace the old address with the one you copied from Ganache

Text

Description automatically generated

1. Navigate terminal to contracts folder with command: *cd..* and *cd client*



1. To install next.js project dependencies (aka create Node modules folder) run command: *npm install*

Note: This may take some time

Graphical user interface, application, chat or text message

Description automatically generated

1. To start the frontend run command: *npm run dev*

Text

Description automatically generated

This will start the frontend on localhost port 3007. If you ctrl + click on the given URL a browser window will pop up on this URL and you should be seeing the Login Page of DEMOS.

Graphical user interface, application, Teams

Description automatically generated

Note 1: Command *npm run dev* is an alias for *next dev -p 3007.* You can view and change the aliases in “scripts” located in package.json file which is in the client folder

Note 2: **Keep frontend running** when you are running DEMOS. You can terminate the deployment of the frontend by giving ctrl + C command in the terminal

CHAPTER 4: METAMASK SETUP

Frontend is up and running! Time to connect Metamask so users can login and interact with DEMOS

1. Switch to Google Chrome were DEMOS Login Page is open, select Extensions and then select Metamask

Graphical user interface, text, application

Description automatically generated

Note: You can pin Metamask extension to make your life a bit easier as you will use it regularly through the rest of the tutorial

1. A new window will open welcoming you to Metamask. Select *Get Started*Text, application

   Description automatically generated with medium confidence
2. Select Import Wallet

Graphical user interface, text, application

Description automatically generated

1. Switch to ACCOUNTS tab in Ganache and copy the MNEMONIC

Graphical user interface, website

Description automatically generated

1. Paste the Mnemonic, set a password, accept Terms of Use and click Import

Graphical user interface

Description automatically generated with medium confidence

1. Wallet is now created and you can see Account 1 which has the same address as the 1st account in Ganache

Graphical user interface, text

Description automatically generated

Graphical user interface, application, website

Description automatically generated

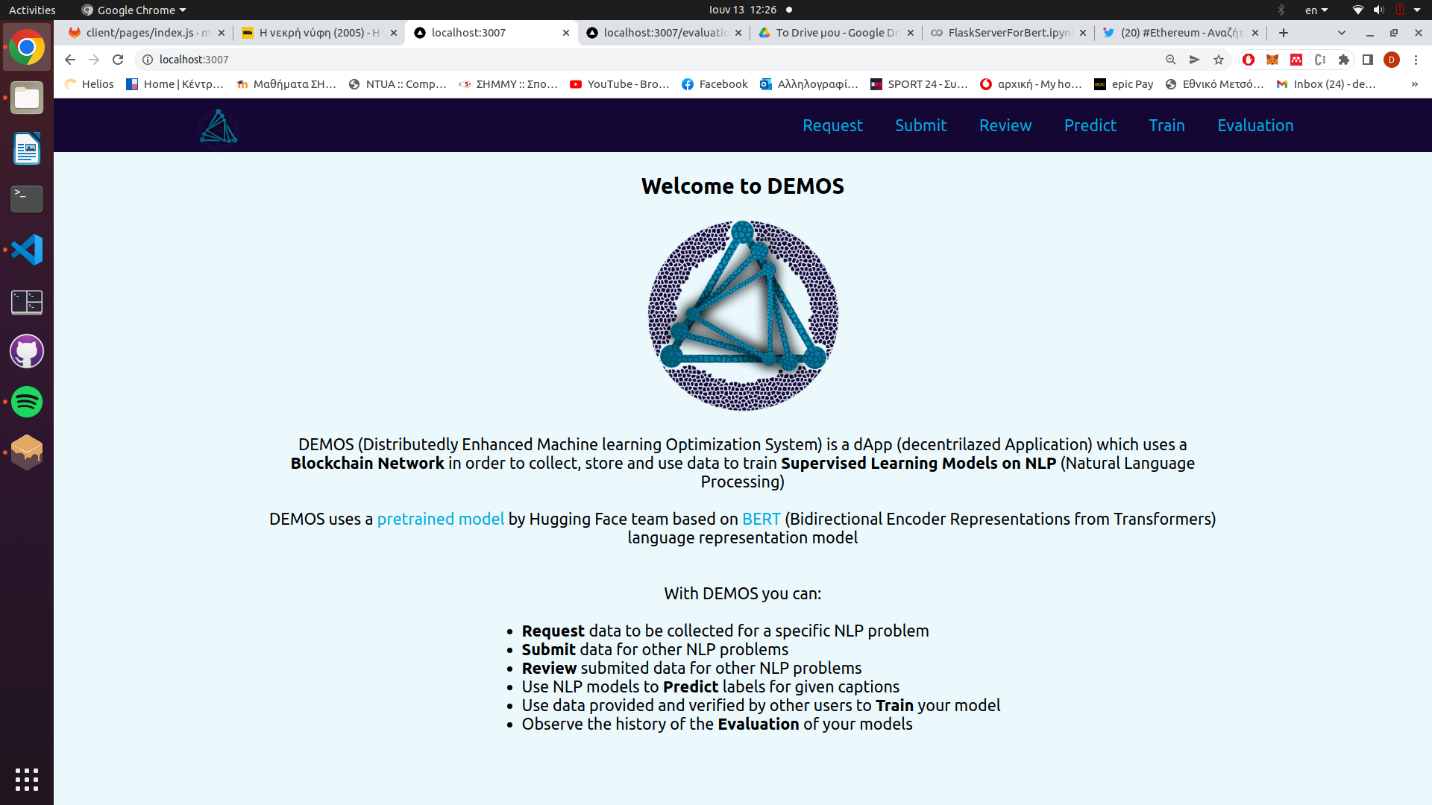
1. Switch to Google Chrome tab where DEMOS Login Page is and click Connect. A Metamask window will pop up. Click *next* and *Connect* to connect frontend with Account 1 with Metamask

Graphical user interface, text, application, chat or text message

Description automatically generated

Note: If Metamask window does not show up you may need to refresh or close and reopen the DEMOS Login Page

You can now see the DEMOS Main Page, but we are not done yet



1. Open Metamask extension

Graphical user interface, application

Description automatically generated

1. Select the Arrow next to Ethereum Mainnet and click Add Network

Graphical user interface, application

Description automatically generated

1. Fill the form with a name for the network (we use Localhost 7545). RPC URL must be set to <HTTP://127.0.0.1:7545> and Chain ID to 1337. You can choose your own Currency Symbol (we used ETH but it is up to you). Then click *Save*

Graphical user interface, text, application, chat or text message

Description automatically generated

1. The new network is successfully added to Metamask and you can see the balance of the account has changed and it is now the same as in Ganache

Graphical user interface, text, application, email

Description automatically generated

1. Switch to ACCOUNTS tab in Ganache and click *Show Keys* of the 2nd account

Table

Description automatically generated with medium confidence

1. Copy Private Key

Graphical user interface, text, application, email

Description automatically generated

1. Switch back to DEMOS page and open Metamask extension again. Click the picture next to Localhost 7545 to open the following menu. Then select *Import Account*Graphical user interface, application

   Description automatically generated
2. Set Private Key as Select Type and paste Private Key. Then click *Import*

Graphical user interface, text

Description automatically generated

1. The 2nd account is successfully added to Metamask

Graphical user interface, application

Description automatically generated

Note: If prompt to connect account to page click Connect

1. Repeat the same procedure to add more accounts to Metamask (we advise you to add at least 5 accounts). You can view the list of added accounts by clicking the image next to Localhost 7545

Graphical user interface, application

Description automatically generated

1. Connect all added accounts to the page by switching between accounts and clicking Not Connected and then Connect. Ensure that all accounts are connected

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, application

Description automatically generated

CHAPTER 5: FLASK SERVER SETUP ON GOOGLE COLAB

Blockchain accounts are connected to the frontend! Now it’s time to set up the Flask server.

If you want to run the Flask server locally continue with Chapter 6

If you want to run the Flask server on Google Colab follow the next steps

1. Switch to Google Drive and double click on FlaskServerForBert.ipynb to open it on Google Colab
2. Ensure that the paths to savedModels folder are correct



Graphical user interface, application, Word

Description automatically generated

1. Change runtime type to GPU and run all cells. Once the server is running you will see thisText

   Description automatically generated
2. Copy the Ngrok URL

Graphical user interface

Description automatically generated with low confidence

1. Switch to Visual Studio Code and navigate to \blockchain-ml-main\client\constants\parameters.js. Replace the URL on APIpath and save changes

Text

Description automatically generated

Note: **Keep Google Colab running** when using DEMOS. If you are on the free tier of Google Colab, use GPU runtime **only when you need it** as it is limited.

And that’s it! We are all set! You can now use the DEMOS dApp

CHAPTER 6: FLASK SERVER SETUP LOCALLY

If you want to run Flask server locally make sure that your computer has adequate infrastructure to support training. If you don’t have a powerful GPU or TPU we strongly advise you to run Flask server on Google Colab.

1. Download the contents of the savedModels folder and place them in \blockchain-ml-main\backend\NeuralNetwork
2. Create a new terminal in Visual Studio Code and navigate it to backend folder with command: cd backend
3. Install all necessary Python libraries

* python -m pip install --upgrade pip
* pip install transformers
* pip install flask
* pip install -U flask-cors
* pip install tensorflow
* pip install pandas
* pip install sklearn

1. Navigate to \blockchain-ml-main\client\constants\parameters.js. Replace the URL on APIpath with http://127.0.0.1:5000 and save changes

Text

Description automatically generated

1. Run local Flask server with command: flask run

Note: You can stop the Flask server with ctrl + C

Congratulations! Everything is set! You can now use the DEMOS dApp

CHAPTER 7: USING DEMOS

Welcome to DEMOS! First you need to Request data the models. We advise you to start with the 3 models created in INITIAL TRAINING before using your own models

A picture containing text

Description automatically generated

1. Go to the Request page and fill the form as shown below

Graphical user interface, text, application, email

Description automatically generated

Note: You can use any Name or Description you like to describe this model. We use 3 votes for validation but you can use a different number. Make sure you add Initial Accuracy of the model as it was in Initial Training using Google Colab. Finally, give labels without quotes or spaces in the same order as in Google Colab.

1. Make sure you are using Account 1 in Metamask and click *Request Model*. A Metamask window pops up to confirm the transaction, click *Confirm*

Graphical user interface, text, application

Description automatically generated

Note: Once you Confirm you will see a window confirming your transaction

Graphical user interface, application

Description automatically generated

1. Follow the same procedure to add the other two models

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Go to Submit page. Here you can see the three models that you previously requested. Change to Metamask Account 2. Give a caption for a model and choose the most appropriate label. Then Submit Caption and confirm transaction.

Graphical user interface, text, application, email

Description automatically generated

Note: An account can request data for a model and then submit such data, but it doesn’t make much sense. That’s why we switch to Account 2

1. After submitting a few captions change to Account 3 and go to Review page. Review a caption by selecting a label. Then click Review Caption and confirm transaction

Graphical user interface, application, Teams

Description automatically generated

Note: if you go to Review page with Account 2 you will see nothing! This is because the node (user) who submitted a caption cannot review it. Switch to Account 3 and refresh the page, now you can see the captions you submitted with Account 2

1. Review the captions using different Metamask accounts until you vote the same label 3 times using 3 different accounts

Note 1: A caption is verified when a specific label gets 3 votes from reviewers. That is because we set Number of Votes for verification to 3. Then the caption disappears from the Review page (if you refresh).

Note 2: Once a caption is verified, you can see that the balance of the accounts that vote for the verified label increases. If the verified caption matches the one that was provided by the node who submitted the caption, then his balance increases too.

1. Go to Predict page. Here, all the accounts can test the models using their own captions. Give it a try!

Graphical user interface, text, application, email

Description automatically generated

1. Switch back to Account 1 (who requested the models) and go to Train Page. Here you can see all the verified captions that can be used for training. Click *Train Model* to retrain the model using these captions. When training and evaluation are finished, you will need to confirm two transactions

Graphical user interface, application, table

Description automatically generated

Note 1: Only the node that requested data for a model can view the verified data and retrain the model

Note 2: Training and evaluation of the model takes time, you can review its progress from the Google Colab page

A picture containing text

Description automatically generated

Note 3: For each caption, if verified label matches proposed label its container is green, otherwise it’s orange.

1. Finally, go to Evaluation page to view the accuracy changes of the model

Note: The models are initially trained using tens of thousands of captions. If you just retrain the model by using a handful of captions it is expected that accuracy will not be greatly affected. It is also possible to have a small drop.

Chart, line chart

Description automatically generated

CHAPTER 8: RESTART DEMOS AFTER THE INITIAL SETUP

1. Start Ganache and select the Blockchain you created
2. Start Visual Studio Code. Open a terminal in client directory and run command: *npm run dev*
3. Start DEMOS frontend on browser to get the Login Page
4. Click Connect, a Metamask window will pop. Login to Metamask using the password set in Step 40

Note: You may need to switch to Ethereum Main Network to Login and then switch back to localhost 7545

1. Run Flask Server
   1. If you are running Flask server on Google Colab, run FlaskServerForBert.ipynb on Google Colab using GPU. You need to replace the APIpath with the new Ngrok URL as you did in steps 57-58 as it changes every time.
   2. If you are running Flask server locally, create a new terminal in backend folder and run command: *run flask*