NLP 2025-26 -- Exam: LLM Project

Laurea Magistrale in Artificial Intelligence, University of Verona
Course: Natural Language Processing (topic: Large Language Models)
Alberto Castellini

This document contains the definition of the project about the LLM module of the NLP course.

1. Objective

The goal of this project is to develop three types of agent based on Large Language Models (LLMs) capable of automatically routing incoming emails/tickets to the appropriate department/office of a company. Routing must be based on the message content.

Each email must be assigned to one of the following departments:

- Technical Support
- Customer Service
- Billing and Payments
- Sales and Pre-Sales
- General Inquiry

The three agents must be implemented in Python (using PyTorch for LLM models) in three different ways:

1. **Routing with prompting using GPT-2 (frozen model)**
   - Use a pretrained GPT-2 model (e.g., the Hugging Face **GPT2** "gpt2" or **DistilGPT2 "**distilgpt2") without updating weights.
   - The model receives an instruction-style **prompt** containing the **task** description and the **email text**, and must output only the **department name**.

2. **Routing with fine-tuning (e.g., LoRA or other Parameter-Efficient Fine-Tuning method) on GPT-2**
   - Reuse Hugging Face **GPT2** "gpt2" or **DistilGPT2 "**distilgpt2", but this time perform **parameter-efficient fine-tuning** (e.g., **LoRA**).
   - The model is trained on a **dataset** of **labeled emails of the company** (see dataset section below).
   - The fine-tuned model is evaluated as the pretrained one (see point 1), namely using the same prompt, but the weights of the model are fine-tuned on the dataset of the company.

3. **Routing with discriminative classifier using a BERT-like model**
   - Use an **encoder** model (e.g., the **DistilBERT base model (uncased)**, distilbert-base-uncased) **fine-tuned** for **text classification**.
   - The model receives the subject and body of the email and it returns probability distribution over the department labels. The label with the highest probability is selected as the predicted class.

2. LLM Models (hints)

Several LLM can be used in this project (see for instance the Hugging Face repository https://github.com/huggingface/transformers and https://github.com/huggingface/transformers/tree/main/src/transformers/models). However, to keep the computation manageable in terms of (memory and processing) resources, the following LLMs are suggested.

- Hugging Face **GPT-2 Small** (**124M parameters**): https://github.com/huggingface/transformers/tree/main/src/transformers/models/gpt2
- Hugging Face **Distilgpt2** (**80M parameters**): https://github.com/huggingface/transformers-research-projects/tree/main/distillation
- Hugging Face **DistilBERT base model (uncased)** (**66M parameters**): https://huggingface.co/distilbert/distilbert-base-uncased https://github.com/huggingface/transformers/tree/main/src/transformers/models/distilbert

Using these LLMs you should be able to generate all the three types of agent using only CPUs. The usage of small GPUs can speedup the training of agents based on fine-tuned LLMs. See section *Resources* for more information about the resources that you can use to develop this project.

3. Dataset

You will use a public dataset of **customer support ticket emails**, called

Tobi-Bueck/customer-support-tickets (Hugging Face Datasets)

Link: https://huggingface.co/datasets/Tobi-Bueck/customer-support-tickets

It contains, among others, the following fields:

- ***subject***: email subject
- ***body***: email/ticket content
- ***queue***: department assigned to the ticket (e.g., Technical Support, Customer Service, Billing and Payments, Sales and Pre-Sales, General Inquiry, etc.). This can be used as a label
- ***language***: language of the ticket (we will use only English)

For this project consider only english texts by filtering emails where language is equal to *"en"*. Furthermore, restrict the set of departments to the five listed above, namely, *Technical Support*, *Customer Service*, *Billing and Payments*, *Sales and Pre-Sales*, *General Inquiry.*

Files used: Data loading and preprocessing must be performed by function *load_and_prepare_data* in file ***datapreparation.py*** (see section 8)**.** The function:

- loads the dataset,
- selects only messages in English language,
- selects only messages for departments "Technical Support", "Customer Service", "Billing and Payments", "Sales and Pre-Sales", "General Inquiry",

- shuffles the messages
- prepares training set (train_ds), validation set (val_ds), test set (test_ds)
- prepares label mapping for discriminative methods
- prints label distribution
- returns dataset partitions and label mappings

Notice: the performance of all three agents must be computed in terms of accuracy on the **test** set, hence all the three agents should call this function to get data for model evaluation. Some agents must also call this function for getting **training** and **validation** data. Remember that test data must not be provided to the model during training.

4. Resources

The project can be developed using standard computers with **CPU** with a few cores (e.g., any modern i5 or i7) and 8–16 GB RAM (16 GB are suggested).

Optionally, a small GPU (4–6 GB VRAM) could be used to speedup the training, but it is not required.

Google Colab (https://colab.research.google.com/) is a suitable tool if you do not want to use your personal computer. It also provides free GPU (with some limitations).

Libraries: the ***build-nanogpt*** conda environment is a good starting point to run the project. Further libraries could be added if needed.

5. Tasks and output

The following tasks must be performed

1. Data preprocessing and exploratory analysis (already implemented in datapreparation.*py*)
2. Development of the three routing methods in a **notebook** (use Virtual Studio Code or Google Colab to generate the notebook file)
3. Comparison of the performance of the three routing methods on dataset *test_ds* in terms of accuracy, computational time and memory. The results of the comparison must be shown in the notebook as a table in the form

Gpt2+ prompting	Distillgpt2+ prompting	Gpt2+ fine tuning	DistilGpt2+ fine tuning	DistilBERT+ classifier	Other models if you want to test
Accuracy 1	Accuracy 2	Accuracy 3	Accuracy 4	Accuracy 5	…

   and a related chart.
4. Generation of a **short report** (<5 pages) containing a short description of the solutions and the results/comparison.

Hints:

- Check how the performance change depending on the prompt
- Show also other metrics than accuracy if you think they are useful to understand the results (e.g., precision, recall)
- Optional: Compare the results also with other LLM methods and non-LLM classifiers

6. Deliverables

Three days before the oral presentation, you must deliver the following material:

1. **Code**: *notebook.ipynb* (notebook file), *datapreparation.py* (data preparation file file), *build-project-conda-environment.yml* (conda environment configuration file), and other files needed to run the project, must be sent via email to alberto.castellini@univt.it
2. **Short report**: sent via email to alberto.castellini@univt.it
3. **Link to the Colab notebook:** sent via email to alberto.castellini@univt.it

7. Organization

- The project must be developed in **groups of 2 or 3 students** (students that prefer to work alone should inform the teacher and motivate it).
- The project will be **presented orally**. During the presentation I can ask **questions** about the implementation and the related theory (see slides about Transformers presented during the course). The **final score** (from 0/30 to 30/30 e lode) will consider i) the quality of project implementation ii) the quality of the results (model performance), iii) the preparation of the student to explain the implementation/results/theoretical background and to motivate implementation choices. The final grade must be >= 18 for passing the exam.
- **NLP Exam Score:** The score for the complete NLP exam is the average between the score of this project (LLM module) and the score of the other project (provided by Prof. Cristani).
- Each group can organize the **oral presentation** independently, by sending an email to the teacher when they have concluded the implementation of the project **(if you have strict deadlines, mainly during the summer session when there are scholarship deadlines, please inform me at least 20 days in advance).**
- On **February 12th from 8.30** I can organize a global presentation session with **1-point bonus**. Students who want to participate are asked to book by sending an email to alberto.castellini@univr.it by February 4th.

8. datapreparation.py code

```
import torch

from datasets import load_dataset
```

```
from collections import Counter
import numpy as np

RANDOM_SEED = 42

def load_and_prepare_data():
    dataset = load_dataset("Tobi-Bueck/customer-support-tickets")
    ds = dataset["train"]

    # Filter only English tickets
    ds = ds.filter(lambda ex: ex["language"] == "en")

    # Select only 5 departments "Technical Support", "Customer Service", "Billing and Payments", "Sales and Pre-Sales", "General Inquiry",
    target_queues = [
        "Technical Support",
        "Customer Service",
        "Billing and Payments",
        "Sales and Pre-Sales",
        "General Inquiry",
    ]

    ds = ds.filter(lambda ex: ex["queue"] in target_queues)

    # Shuffle and split train/val/test
    ds = ds.shuffle(seed=RANDOM_SEED)
    train_test = ds.train_test_split(test_size=0.2, seed=RANDOM_SEED)
    test_valid = train_test["test"].train_test_split(test_size=0.5, seed=RANDOM_SEED)

    train_ds = train_test["train"]
    val_ds = test_valid["train"]
    test_ds = test_valid["test"]
```

```
# Label mapping for discriminative methods
label_list = sorted(list(set(train_ds["queue"])))
label2id = {lab: i for i, lab in enumerate(label_list)}
id2label = {i: lab for lab, i in label2id.items()}

print("Label distribution (train):")
print(Counter(train_ds["queue"]))

return train_ds, val_ds, test_ds, label_list, label2id, id2label
```