Assignment 4: Emotion Classification using LLMs

COMP 551 Fall 2024, McGill University
Contact TAs: Charlotte Volk and Abdelrahman Ayad

Released on November 11
Due on December 4 midnight

Preamble

- This assignment is **due on December 4th 11:59pm (EST, Montreal Time).** There is a penalty of 2^k percent penalty for k days of delay, which means your grade will be scaled to be out of $100 - 2^k$. No submission will be accepted after 6 days of delay.
- This assignment is to be completed in groups of three. All members of a group will receive the same grade except when a group member is not responding or contributing to the project. If this is the case and there are major conflicts, please reach out to the group TA for help and flag this in the submitted report. Please note that it is not expected that all team members will contribute equally. However every team member should make integral contributions to the project, be aware of the content of the submission and learn the full solution submitted.
- You will submit your assignment on MyCourses as a group. You must register your group on MyCourses and any group member can submit. See MyCourses for details.
- We recommend to use **Overleaf** for writing your report and **Google colab** for coding and running the experiments. The latter also gives access to the required computational resources. Both platforms enable remote collaborations.
- There are additional compute resources available for this project, including a SLURM cluster from McGill's School of Computer Science. Please check MyCourses for instructions on how to access those. A tutorial can be found in the recorded lectures.
- You should use Python for this assignment. You are free to use libraries with general utilities, such as matplotlib, numpy and scipy for Python, unless stated otherwise in the description of the task. In this assignment, you are allowed to use pre-existing implementations of the algorithms or functions as found in SciKit learn, and other packages. You are

also allowed to use PyTorch to load and finetune your LLM models. The description will specify this in a per case basis.

Synopsis

In this assignment, you will implement and finetune a Large Language Model (LLM), for example Bidirectional Encoder Representations from Transformers (BERT), using the existing Tensorflow or Pytorch libraries. You can then use these pre-trained weights to do the Emotion prediction task. Then you need to finetune your model or choose to modify the structure of the model to see whether it could help you to improve the final performance. You are not required to implement this model from scratch. You may use the pre-downloaded LLMs on the SCS GPU servers: BERT base model (uncased) `https://huggingface.co/google-bert/bert-base-uncased` and Distilled-GPT2 `https://huggingface.co/distilbert/distilgpt2`. You will also implement naive Bayes from scratch using bag of words representation of the data. You will evaluate your finetuned LLM against the pretrained version of the LLM without finetuning, against your naive Bayes implementation, and also against one baseline (Softmax Regression (SR), Random Forest (RF), or XGBoost) on the GoEmotions dataset.

The goal is to gain hands-on experience running the modern deep learning libraries and evaluating their performances on the real-world dataset against the more traditional methods.

1 Task 1: Preprocess dataset

The GoEmotions is a corpus of 58k carefully curated comments extracted from Reddit, with human annotations to 27 emotion categories or Neutral: `https://huggingface.co/datasets/google-research-datasets/go_emotions`.
Here is a link to the paper about the dataset: `https://arxiv.org/pdf/2005.00547`.
To simplify the data, you may drop all data points with more than one label (so you will end up with 85% of the dataset).

For the baseline method (SR, RF, or XGBoost), you can look at the sklearn Vectorizer functions to transform the text data into the relevant format, e.g. CountVectorizer or TfidfVectorizer. You should use these to transform the text data into numerical features.

For the naive Bayes method, you need to design the data preprocessing pipeline that turns the unstructured text data into numerical features. Specifically, you should use the bags of words representation using the scikit-learn function CountVectorizer1.

For your LLM, you can use the transformers package to tokenize the input text and convert the tokens into numerical features https://pytorch.org/hub/huggingface pytorch-transformers/. You are free to use any Python libraries you like to extract features and preprocess the data.

Same as in Assignment 2, you need to use only comments in the “train” folder for training, “validation” folder for tuning your model hyperparameters if needed, and report the performance from the “test” folder.

2 Task 2: Implement Naive Bayes & Finetune an LLM

For the baseline model (SR, RF, or XGBoost) you may use the sklearn implementation directly. However, please make sure to report the settings for each of these methods you run even if you are using the default settings.

You must implement the Naive Bayes model from scratch (i.e., you cannot use Scikit Learn or any other pre-existing implementations of these methods).

The BERT and GPT2 models on the SCS servers are already pre-trained for you on large corpus datasets such as Wikipedia. In this task, you will take one of these models and finetune it on an emotion classification task using the GoEmotions dataset.

To minimize your memory requirements, you can fine-tune only the last layer of the LLM.

2.1 Naive Bayes Model Details

For the Naive Bayes model, you must use Python and you must implement the model from scratch (i.e., you cannot use Scikit Learn or similar libraries). Using the numpy package is encouraged. Regarding the implementation, we recommend the following approach:

1. Implement naive Bayes model as a Python class. You should use the constructor for the class to initialize the model parameters as attributes, as well as to define other important properties of the model.
2. Your model class should have (at least) these functions:
   (a) Define a **fit** function, which takes the training data (i.e., X and Y)—as well as other hyperparameters (e.g., the learning rate and/or number of gradient descent iterations)—as input. This function should train your model by modifying the model parameters.
   (b) Define a **predict** function, which takes a set of input features (i.e., X) as input and outputs predictions (i.e., Yˆ) for these points.
   (c) Define a function **evaluate acc** to evaluate the model accuracy. This function should take the true labels (i.e., Y), and target labels (i.e., Yˆ) as input, and it should output the accuracy score.

2.2 LLM Model Details

You can access two different types of LLMs from the SOCS servers: bert-base-uncased and distilgpt2. The paths to these models are: **/opt/models/bert-base-uncased** and **/opt/models/distilgpt2**. Please do **not** try to install these LLMs locally, as you will run out of space. You can load the models using the transformers package. Here's an example of some code to get started with:

tokenizer = AutoTokenizer.from_pretrained("PATH_TO_LLM")
model = AutoModel.from_pretrained("PATH_TO_LLM")

Task 3: Run experiments

The goal of this project is to have you explore your LLM.

You are welcome to perform any experiments and analyses you see fit, **but at a minimum you must complete the following experiments in the order stated below**:

1. Report the classification performance for your finetuned LLM, the pretrained LLM, the Naive Bayes model, and **one of** SR, RF, or XGBoost on the GoEmotions dataset.
2. Examine the attention matrix between the words and the class tokens for some of the correctly and incorrectly predicted documents. You will need to choose one of transformer blocks and use a specific attention head for the multi-layer multi-headed transformer architecture in your LLM.

Note: The above experiments are the minimum requirements that you must complete; however, this project is open-ended. Here are some suggestions of what you could do:

1. Implement a small version of the LLM on your own from scratch (using PyTorch for example).
2. Explore the accuracy of different pre-trained LLM models (e.g., from here `https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb` or here `https://pypi.org/project/pytorch-pretrained-bert/`) on the emotion prediction.
3. You may also try different pre-training tasks such as MLM (masked language model) and NSP (next sentence prediction) directly on the emotions data and report the resulting accuracy after fine-tunijg.
4. Try word2vec (`https://wikipedia2vec.github.io/wikipedia2vec/pretrained/`) to project the tokens from the Reddit comments onto embedding space and then use RF, SR, XGBoost, and/or MLP to classify the embedded documents.
5. Compare two different LLMs (for example BERT, Mistral `https://huggingface.co/mistralai/Mistral-7B-v0.1`, or GPT-2 (`https://huggingface.co/gpt2`).

6. Try using the multilabel data from the GoEmotions dataset. Remember that softmax cannot do multilabel classification, so you will need a sigmoid activation on the final classification layer if you want to attempt this.
7. Implement multiple baselines for comparison. You could implement all three of SR, RF, and XGBoost, or try a different type of model.

As before, you do not need to do all of these things, but look at them as suggestions and try to demonstrate curiosity, creativity, rigour, and an understanding of the course material in how you run your chosen experiments and how you report on them in your write-up.

Deliverables

You must submit two separate files to MyCourses (**using the exact filenames and file types outlined below**, where k is your group number.):

1. `assignment4_group-k.ipynb`: Your data processing, classification and evaluation code should be all in one single Jupyter Notebook. Your notebook should reproduce all the results in your reports. The TAs may run your notebook to confirm your reported findings.
2. `assignment4_group-k.pdf`: Your **(max six page**) assignment write-up as a pdf (details below). Make sure to stay inside the page limit, anything longer than the page limit will not be graded. If you have extra figures or comments that you wish to put in an appendix, you may do so, but know that the TAs grading are under no obligation to look at your appendix in order to grade your report.

Project write-up

Your team must submit a project write-up that is a **maximum of six pages** (single-spaced, 11pt font or larger; minimum 0.5 inch margins, an extra page for references/bibliographical content can be used). We highly recommend that students use LaTeX to complete their write-ups. You have some flexibility in how you report your results, but you must adhere to the following structure and minimum requirements:

A note about the appendix:
Your main results should be in the main body of the paper. We should be able to grade the paper and follow the main steps and results without resorting to the appendix. We will penalize groups who add core results to the appendix. However, if you have supplementary results that you wish to put in the appendix, you may do so, and you may refer to these in the main text. We will not penalize you for referring to supplementary results in the main text.

Abstract (100-250 words) Summarize the project task and your most important findings

Introduction/Background Summarize the project task, the dataset, and your most important findings. You should include background information and citations to relevant work (e.g., other papers analyzing these datasets).

Methods Present your experimental design, including:

1. Data preprocessing & exploratory analysis
2. Describe how you implemented the Naive Bayes algorithm
3. Describe the LLM you chose and how you implemented the finetuning process
4. The settings you used for each model, e.g. regularization, number of trees, etc.

Results Describe the results of all the experiments mentioned in Task 2 and 3 (at a minimum) as well as any other interesting results you find.

Discussion and Conclusion Summarize the key takeaways from the project and possibly directions for future investigation. You can also consider the ethical implications of working with these large datasets, or think about what pretraining does that helps with this task in particular.

Statement of Contributions State the breakdown of the workload across the team members.

Evaluation

The assignment is out of 100 points, and the evaluation breakdown is as follows:

- Completeness (20 points)
  - Did you submit all the materials? (5 points)
  - Did you run all the required experiments? (5 points)
  - Did you follow the guidelines for the project write-up? (10 points)
- Correctness (45 points)
  - Is the Naive Bayes model implemented correctly? (15 points)
  - Is the LLM implemented and finetrained correctly? (15 points)
  - Are your reported results (accuracies, attention matrices, etc.) reasonable? (15 points)
- Writing quality (10 points)
  - Is your report clear and free of grammatical errors and typos? (5 points)

  - Do you effectively present numerical results (e.g., via tables or figures)? (5 points)
- Originality / creativity (25 points)
  - Did you go beyond the bare minimum requirements for the write-up (e.g., by including a discussion of related work in the introduction)? (5 points)
  - Did you go beyond the bare minimum requirements for the experiments? (20 points)
  - **Note:** Simply adding in a random new experiment will not guarantee a high grade on this section! You should be thoughtful and organized in your report.

Final remarks

You are expected to display initiative, creativity, scientific rigour, critical thinking, and good communication skills. You don't need to restrict yourself to the requirements listed above - feel free to go beyond, and explore further. You can discuss methods and technical issues with members of other teams, but **you cannot share any code or data with other teams.**