Due Date Saturday October 22, 2022 (11:59pm)
Late Submissions 20% per day
Teams You can do the mini-project in teams of at most 3.
Teams must submit only 1 copy of the mini-project via the team leader's account.
Purpose The purpose of this mini-project is to make you experiment with machine learning and natural language processing.

Emotion and Sentiment Classification of Reddit Posts

For this mini-project, you will experiment with different machine learning algorithms for text classification. You will use a modified version of the GoEmotion dataset and experiment with a variety of classifiers and features to identify both the emotion and the sentiment of Reddit posts. As you will use many built-in functions from different libraries, the focus of this mini-project lies more on the experimentations and analysis than on the implementation.

1 Before you start

1. Make sure you use Python 3.8, and have installed the `scikit-learn`, the `gensim` and the nltk libraries.
2. Other Python libraries can be used if they are useful to you; for example `numpy` or `pandas`.
3. You can use Jupyter notebook if you prefer. In that case, submit the `.ipynb` files; otherwise submit the `.py` files.
4. In case of issues with your team dynamics, we will use GitHub to evaluate the contribution of each team member to the project. If you do use GitHub while developing, make sure your project is private until the project deadline.

2 Your Tasks

Use the ☐ as a check-list to make sure everything is done.

1. **Dataset Preparation & Analysis** (5pts)

   1.1. ☑ (0pts) Download the version of the GoEmotion dataset provided on Moodle.
   The original GoEmotion dataset, created by [Demszky et al., 2020], is a dataset of 58k human-annotated Reddit comments labeled with 27+1 emotion categories (eg. *admiration*, *amusement*, *anger*, *caring*, ...) and *neutral*. These emotions are themselves organized into 4 sentiments: POSITIVE (*admiration*, *amusement*, ...), NEGATIVE (*anger*, *annoyance*, ...), AMBIGUOUS (*confusion*, *curiosity*) and NEUTRAL (*neutral*)). This allows us to use the dataset for both:
   - emotion classification (into 28 classes), and
   - sentiment classification (into 4 classes).

   read these later

   For more information on the original dataset, you can read this blog and this paper.
   The dataset we will use for this assignment is a modified version of the original GoEmotion, where only posts annotated with a single emotion (and a single sentiment) are kept, and the data is formatted in json. The json file contains triplets made of the post, its emotion and its sentiment.

   1.2. ☑ (0pts) Load the dataset. You can use `gzip.open` and `json.load` to do that.

   1.3. ☐ (5pts) Extract the posts and the 2 sets of labels (emotion and sentiment), then plot the distribution of the posts in each category and save the graphic (a histogram or pie chart) in pdf. Do this for both the emotion and the sentiment categories. You can use `matplotlib.pyplot` and `savefig` to do this. This pre-analysis of the dataset will allow you to determine if the classes are balanced, and which metric is more appropriate to use to evaluate the performance of your classifiers.

2. **Words as Features** (35pts)

Let's first use a classic *word as features* approach to text classification.

2.1. ☐ (5pts) Process the dataset using `feature_extraction.text.CountVectorizer` to extract tokens/words and their frequencies. Display the number of tokens (the size of the vocabulary) in the dataset.

2.2. ☐ (2pts) Split the dataset into 80% for training and 20% for testing. For this, you can use `train_test_split`.

2.3. Train and test the following classifiers, for both the emotion and the sentiment classification, using word frequency as features.

2.3.1. ☐ (3pts) **Base-MNB:** a Multinomial Naive Bayes Classifier (`naive_bayes.MultinomialNB.html`) with the default parameters.

2.3.2. ☐ (3pts) **Base-DT:** a Decision Tree (`tree.DecisionTreeClassifier`) with the default parameters.

2.3.3. ☐ (3pts) **Base-MLP:** a Multi-Layered Perceptron (`neural_network.MLPClassifier`) with the default parameters.

use new classifier for this!!!

2.3.4. ☐ (3pts) **Top-MNB**: a better performing Multinomial Naive Bayes Classifier found using `GridSearchCV`. The gridsearch will allow you to find the best combination of hyper-parameters, as determined by the evaluation function that you have determined in step 1.3. The only hyper-parameter that you will experiment with is `alphafloat` with values 0.5, 0 and 2 other values of your choice.

https://bait5

2.3.5. ☐ (3pts) **Top-DT**: a better performing Decision Tree found using `GridSearchCV`. The hyper-parameters that you will experiment with are:

- `criterion`: gini or entropy
- `max_depth`: 2 different values of your choice
- `min_samples_split`: 3 different values of your choice

2.3.6. ☐ (3pts) **Top-MLP:** a better performing Multi-Layered Perceptron found using `GridSearchCV`. The hyper-parameters that you will experiment with are:

- `activation`: sigmoid, tanh, relu and identity
- 2 network architectures of your choice: for eg, 2 hidden layers with $30+50$ nodes and 3 hidden layers with $10+10+10$
- `solver`: Adam and stochastic gradient descent

2.4. ☐ (5pts) For each of the 6 classifiers above and each of the classification tasks (emotion or sentiment), produce and save the following information in a file called `performance`:

- a string clearly describing the model (e.g. the model name + hyper-parameter values) and the classification task (emotion or sentiment)
- the confusion matrix – use `metrics.confusion_matrix`
- the precision, recall, and F1-measure for each class, and the accuracy, macro-average F1 and weighted-average F1 of the model – use `metrics.classification_report`

2.5. ☐ (7.5pts) **Do your own exploration:** Do only one of the following, depending on your own interest:

- Use tf-idf instead of word frequencies and redo all substeps of 2.3 above – you can use `TfidfTransformer` for this. Display the results of this experiment.
- Remove stop words and redo all substeps of 2.3 above – you can use the parameter of `CountVectorizer` for this. Display the results of this experiment.
- Play with `train_test_split` in order have different splits of 80% training, 20% test sets and different sizes of training sets and redo all substeps of 2.3 above. Show and explain how the performance of your models vary depending on the training/test sets are used.

seperate this in a new step

3. **Embeddings as Features** (20pts)

   Now, instead of using word frequencies (or tf-idf) as features, we will use Word2Vec embeddings.

   3.1. ☐ (0pts) Use `gensim.downloader.load` to load the `word2vec-google-news-300` pretrained embedding model.

   3.2. ☐ (2pts) Use the `tokenizer` from `nltk` to extract words from the Reddit posts. Display the number of tokens in the training set.

   3.3. ☐ (5pts) Compute the embedding of a Reddit post as the average of the embeddings of its words. If a word has no embedding in Word2Vec, skip it.

   3.4. ☐ (3pts) Compute and display the overall hit rates of the training and test sets (i.e. the % of words in the Reddit posts for which an embedding is found in Word2Vec).

   3.5. ☐ (3pts) Train a **Base-MLP:** a Multi-Layered Perceptron (`neural_network.MLPClassifier`) with the default parameters.

   3.6. ☐ (3pts) Train a **Top-MLP:** a better performing Multi-Layered Perceptron found with whatever hyperparameters you want.

   3.7. ☐ (2pts) Display the performance of your classifiers using `metrics.classification_report` and add these to your `performance` file.

   3.8. ☐ (7.5pts) **Do your own exploration:** Rerun your best performing model but with 2 other English pretrained embedding models and compare the results. Many pre-trained embeddings are available on line (including in Gensim or at http://vectors.nlpl.eu/repository).

4. **Analysis** (8pts)

   In a separate file called `analysis`, explain in a few pages[1]:

   4.1. ☐ (2pts) An analysis of the dataset given on Moodle. If there is anything particular about these datasets that might have an impact on the metric to use or the performance of some models (see task 1.3), explain it.

   4.2. ☐ (5pts) An analysis of the results of all the models for both classification tasks. In particular, compare and contrast the performance of each model with one another, and with the datasets.

   Please note that your discussion must be analytical. This means that in addition to stating the facts (e.g. the macro-F1 has this value), you should also analyse them (i.e. explain why some metric seems more appropriate than another, or why your model did not do as well as expected.) Tables, graphs and contingency tables to back up your claims would be very welcome here.

   4.3. ☐ (1pt) In the case of team work, a description of the responsibilities and contributions of each team member.

[1] by *a few pages*, I mean 2 to 3 pages of single-space text (ie. not counting figures, tables and other fillers).

3 Deliverables

The submission of the mini-project will consist of 2 deliverables:

1. The Code & the Output Files:

   - ☐ Submit all files necessary to run your code. If you used a Jupyter notebook, submit the `.ipynb` files; otherwise submit the `.py` files.
   - ☐ Submit a `readme.md` which will contain specific and complete instructions on how to run your experiments. You do not need to submit the datasets. If the instructions in your readme file do not work, are incomplete or a file is missing, you will not be given the benefit of the doubt.
   - ☐ Submit the output files: the PDF plots, the `performance` files, and the `analysis` file.

2. The Demo

   You will have to demo your mini-project for ≈ 15 minutes. Regardless of the demo time, you will demo the program that was uploaded as the official submission on or before the due date. The schedule of the demos will be posted on Moodle. The demos will consist of 2 parts: a presentation (≈ 5) minutes and a Q/A part (≈ 10 minutes).

   - ☐ Prepare an 5-minute presentation to explain your Analysis (see part 4 above).
   - ☐ After your presentation, your TA will proceed with a ≈ 10 minute question period. Each student will be asked questions on the code/mini-project, and they will be expected to understand and explain any part of the code and any part of the analysis. Hence every member of team is expected to attend the demo.
   - ☐ At the end of the demo, your TA will give you a new dataset and ask you to train or run a few of your models on this dataset. The output generated by your program will have to be uploaded on EAS during your demo.

4 Evaluation Scheme

Students in teams can be assigned different grades based on their individual contribution to project.

The team grade will count for 90% and will be based on:

Dataset Preparation & Analysis	5	
Words as Features	38	(5+2+3+3+3+3+3+3+5+8)
Embeddings as Features	24	(2+5+3+2+2+2+8)
Analysis	8	(2+5+1)
Demo Presentation	10	
Output at Demo Time	5	
Total team grade	90	

Individual grades will count for 10% and will be based on:

1. the Q/A of each student during the demo.
2. a peer-evaluation done after the submission.
3. the contribution of each student as indicated on GitHub.

5 Submission

5.1 Team Leader

If you work in a team, identify one member as the team leader and make sure this information is indicated in the Groups on the Moodle page. The only additional responsibility of the team leader is to upload all required files (including the files at the demo) from their account and book the demo on the Moodle scheduler.

If you work individually, by definition, you are the team leader of your one-person team.

5.2 Submission Checklist

1. **Code, Output files and Presentation Material**
   Deadline: Saturday Oct. 22, 11:59pm

   - ☐ In your GitHub project, include a `README.md` file that contains:
     1.1. on its first line: the URL of your GitHub repository, what?
     1.2. specific and complete instructions on how to run your program.
   - ☐ Create one zip file containing all your code, the output files for the dataset on Moodle and the `README.md` file.
   - ☐ Name your zip file: `472_Assignment1_ID1_ID2_ID3.zip` where `ID1` is the ID of the team leader.
   - ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Programming Assignment Group 1 / Submission Number 1`.

2. **Demo Output files**

   During your actual demo with the TA:

   - ☐ Prior to your demo, make your GitHub repository public.
   - ☐ During the demo, generate the output files for the data set that the TA will give you.
   - ☐ Create a zip file called: `472_Demo1_ID1_ID2_ID3` where `ID1` is the ID of the team leader.
   - ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Programming Assignment Group 1 / Submission Number 2`

Have fun!

References

[Demszky et al., 2020] Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., and Ravi, S. (2020). Goemotions: A dataset of fine-grained emotions. arXiv preprint arXiv:2005.00547.