Capstone Project Proposal

1. Problem Statement

This project seeks to answer the research question:

Can a pre-trained transformer-based model (i.e. a large language model or LLM) be used to improve the performance of non-transformer-based (NTB) classifier models (e.g. logistic regression or a neural network classifier) by augmenting its original training data?

The following hypothesis will be used in attempting to answer the research question:

```
(performance of non-augmented training data) <
(performance of augmented training data)
```

1.1 Definitions and Metrics

non-augmented - This is the base case where only the originally provided data is used to train the NTB classifier models.

augmented - In this scenario, the original training data is doubled in size by prompting an LLM to provide a sample that is similar to each of the original training samples regardless of class. The LLM decides what "similar" means.

metrics - The ROC AUC on the sample held out from the test data (described in the Data section) and the overall accuracy the unlabeled test data provided by kaggle will be the metrics used to evaluate the hypothesis against two models: 1) a logistic regression and 2) a single hidden layer neural network.

2. Data

2.1 Base Data

Two types of data will be used in the project. The first type will be referred to as "base data" which is obtained from the ongoing kaggle competition titled "*Natural Language Processing with Disaster Tweets*" which is currently available at: https://www.kaggle.com/competitions/nlp-getting-started/data

The base data consists of two files: **train.csv** and **test.csv** which have the following characteristics:

train.csv - This file consists of 8562 lines of raw text. Each actual sample (row) has the following 5 fields (columns):

- **id -** integer, unique identifier for each row which should always have a value
- **keyword -** string, a particular keyword from the tweet which may be blank
- **location -** string, the location the tweet was sent from which may be blank
- **text -** string, the text of the tweet
- **target -** integer, 1 or 0 representing a binary label to be classified and denotes whether a tweet is about a real disaster (1) or not (0)

test.csv - This file consists of 3700 lines of raw text. Each actual test sample (row) has the same first 4 fields (columns) as the train.csv file: **id**, **keyword**, **location** and **text**. There is no **target** column because competition competitors are expected to predict and record this prediction as part of their submissions.

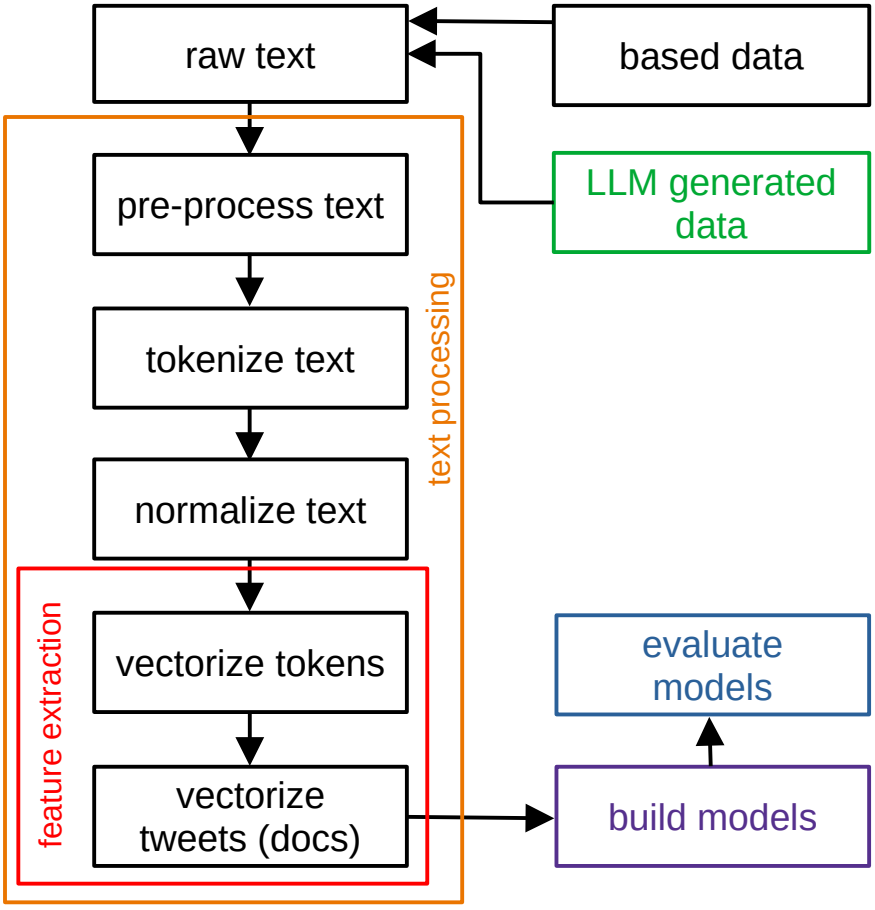
Because test.csv data is unlabeled, a 20% random sample from the training data will be set aside to evaluate ROC AUC to assess model performance. The unlabeled test data will be used to evaluate accuracy under the experimental conditions described in the *Problem Statement* section.

2.2 Augmented Data

The second type of data is referred to here as "augmented data". This data is generated from a LLM based on a pre-determined prompt. A 20% random sample from this data will also be set aside as part of the ROC AUC assessment. The prompt will be designed to create samples that are similar to each of the base data samples regardless of class label.

3. Methodology

The workflow can be described by the diagram right:



pre-processing - This is the processing that needs to happen before tokenization. For example, most of the base data samples are contained on one line. However, since some of the samples spill over to multiple line, these need to be fixed prior to feature extraction and building models.

tokenize text - Tokenization divides text into its smallest meaningful parts such as words, punctuation and white space.

normalize text - The main goal of normalization is to convert tokens into forms that allows the next vectorization steps to retain as much information at possible. Things like lemmatization which converts words to their base form (e.g "walk" is the base word for "walking"), cleaning up contractions and removing stop words are done in this step.

vectorize tokens - Word tokens are encoded as vectors using either one-hot encoding or pre-trained gloVe embeddings (https://nlp.stanford.edu/projects/glove/).

vectorize tweets - Word tokens are aggregated into a single vector representing the tweet. For one-hot encoding, this will simply be the sum of all the token vectors in the tweet. For gloVe embeddings, a concatenation of the minimum, mean and maximum coordinates of the token vectors will be used.

build models - Vectorized tweets will be inputs to training two models: a logistic regression and a single hidden-layer neural network.

evaluate models - Models will be evaluated by ROC AUC on the 20% hold out from the training data and the accuracy on the unlabeled test data scored by kaggle.

4. Resources

Software

The project will be implemented in a Python jupyter notebook with a project module of helper functions intended to keep the notebook code clean and focused on the higher level aspects of the project. Free and open-source libraries such as sklearn and pytorch will be used to build the logistic regression and neural network models described in the *Methodology* section. The spaCy and NLTK libraries will be used for tokenization, lemmatization and other text processing tasks.

A Hugging Face model (https://huggingface.co/models) is expected to be used as the LLM to generate the augmented data. Based on my current understanding of the terms of use, a free account should be sufficient to meet the goals of the project.

The following github project has been set up to do version control throughout development:

https://github.com/MichaelSzczepaniak/llmamd

Hardware

Most of the work is expected to be done on a Ryzen 3700 workstation with 64 Gb of RAM and NVidia GeForce 2060 graphics with 6 Gb dedicated GPU memory. Because the neural network classifier is currently only expected to have 1000 hidden layer nodes, training this model should not be excessively compute-intensive even for a system without GPU resources.

5. Project scope

If time allows, additional prompts will be explored in an attempt to improve performance. Items such as hyper-parameter tuning of the neural network is not expected to be done in this project.