Arun Agarwal

Professor Abha Belorkar

CIS 4496 - Projects in Data Science

January 18th, 2023 - May 3rd, 2023

Honors Contract Weekly Meeting Notes

Long-Term Tasks:

- Look into related work to get ideas of how to improve score
    - Also, Kaggle Competition competitors
- Perform preprocessing
- Create new features
- Perform EDA
- Try out various models
    - Tune models
- Produce final results
- Create graphics to explain results/performance
- Write Model-Performance Report
- Submit Project Folder, this document, and Model-Performance to Professor to complete Contract

Week 0 (1/18/2023 - 1/25/2023):

Completed:

- Reached out to Professor to figure out what to do for contract and met with Professor (20 minutes)
- Read into Professor's suggestion for Contract for Google Colab Notebook for Assignment (1 hour)
- Looked into various Kaggle competitions and potential research paper options for contract (1 hour 15 minutes)

Time Spent (Since Last Meeting): 2 hours 35 minutes

To-Do's (For Next Wednesday):

- Decide on project/paper/idea for contract
- Write contract

Feedback:

- Provided verbally in meetings after class

Week 1 (1/25/2023 - 2/1/2023):

Completed:

- Met with Professor to discuss final choice for project and requirements (10 minutes)
- Looked into NLP Disaster Tweet GitHub Link from previous team Professor sent me as well as the competition itself (30 minutes)
- Informed Professor on Contract Decision and Wrote Contract (45 minutes)
- Reread through NLP Competition in detail to determine what would need to be done (45 minutes)
- Started work on competition by downloading the dataset, setting up an organized file structure for the project, creating notebooks for the various stages of the project, and completing the tutorial for NLP (with relevance to the project), which allowed me to obtain my first estimated F1 score of approximately .6002 (1 hour)

Time Spent (Since Last Meeting): 3 hours 10 minutes

To-Do's (For Next Wednesday):

- Research into ways to improve baseline performance (and take notes)
- Research into types of features you can create (and take notes)
- Look into previous work on this project (GitHub link provided, as well as code on Kaggle)

Feedback:

-

Week 2 (2/1/2023 - 2/8/2023):

Completed:

- Wrote out this document to keep track of progress (20 minutes)
- Looked into previous work on project from Kaggle (2 hours 10 minutes)
    - https://www.kaggle.com/code/gunesevitan/nlp-with-disaster-tweets-eda-cleaning-and-bert
    - https://www.kaggle.com/code/szelee/a-real-disaster-leaked-label
    - https://www.kaggle.com/code/holfyuen/basic-nlp-on-disaster-tweets
    - https://www.kaggle.com/code/aaroha33/disaster-tweets-evaluation-with-nlp

What I can do:

- EDA:
    - ~~check shape,~~
    - ~~amount of missing values,~~
    - ~~number of unique keywords and locations (get target distribution graph),~~
    - check distributions of all created features (will help determine relation between training and test as well as 1 or 0),
    - distribution of target variable in train,
    - make count graphs for unigrams, bigrams, and trigrams
    - Top key words for disaster tweets and non-disaster tweets as two separat graphs
    - Make graph of top locations (will see that data is not clean bc USA and United States both exist as well as cities in U.S.)
- Theory: Training and test taken from same sample

- Location: not auto-generated; they are user input → means lots of unique values and is dirty → don't use
- Can use keyword as feature or as additional work to add to text
- Disaster tweets - more formal, longer words → makes sense as they would come from news agencies
- Non-disaster tweets - likely have more typos because coming from individual users
- Preprocessing:
    - Can try to clean location data through function
    - Fill in missing values with no_keyword
    - Figure out if you should drop duplicate rows
    - Create feature: word_count: number of words in text
    - Create feature: unique_word_count: number of unique words in text
    - Create feature: stop_word_count: number of stop words in text
        - Could use nltk corpus or manual list you found online
    - Create feature: url_count: number of urls in text
    - Create feature: mean_word_length: average character count in words
    - Create feature: mode_word_length: most common character count in words
    - Create feature: char_count: number of characters in text
    - Create feature: punctuation_count: number of punctuations in text
    - Create feature: hashtag_count: number of hashtags (#) in text
    - Create feature: emoji_count: dont know if dataset contains emojis, but if it does, might be a good idea to get a count of emojis
    - Create feature: slang_word_count: count of slang words
        - Need to figure out how I could make this
    - Create feature: mention_count: number of mentions (@) in text
    - Create features based off unigrams, bigrams, and trigrams
    - Text cleaning using embeddings: GloVe-300d-840B, FastText-Crawl-300d-2M
        - Separate #, @, !, ?, +, &, -, $, =, <, >, |, {, }, ^, ', (,),[,], *, %, ..., ', ., :, ; from words
        - Special characters attached to words should be completely removed
        - Expand contractions
        - Remove urls
        - Replace actual symbols with character entity references
        - Fix typos, slang, and informal abbreviations
        - Expand usernames manually (because no spaces, can't use embeddings)
            - Provide a lot of context
        - Remove additional white space
        - Create separate columns to hold lists of hashtags, mentions, and links
    - Deal with mislabeled samples (18 exist - labeled differently in their duplicates)
- Model:

    - Could apply target encoding on keyword and location, then count vectorize cleaned text, links, hashtags, and mention columns; TfidfVectorizer for text
    - You should first start with trying a Logistic Regression model with cross-validation (5 splits, test size = .2); also do a gridsearch
    - Perform cross-validation [StratifiedKFold] (2 folds maybe)
    - Set shuffle to true for extra training diversity
    - BERT: from TensorFlow Models repository on GitHub at tensorflow/models/official/nlp/bert
        - L = 12 hidden layers, H = 768 hidden size, A = 12 attention heads
        - It is pre-trained for English on the Wikipedia and BooksCorpus
        - Lower case input before tokenization into word pieces
        - Remove any accent markers before tokenization into word pieces
        - Must activate Internet on the kernel to use
    - Use FullTokenizer at tensorflow/models/official/nlp/bert/tokenization.py
        - Use parameters such as lr, epochs, and batch_size for controlling the learning process
    - No dense or pooling layers added after the last layer of BERT
    - SGD for optimizer since others have hard time while converging
- Metric: Macro Average F1 Score
    - Should also look at accuracy, precision, recall, and normal F1 score for performance grading → Can use classification report

Time Spent (Since Last Meeting): 2 hours 30 minutes

To-Do's (For Next Wednesday):

- Continue to explore what I can do to increase Macro F1 score

Feedback:

Week 3 (2/8/2023 - 2/15/2023):

Completed:

- Reviewed notes above and added to them (30 minutes)
- Debugged Coding Environment (issues with libraries like wordcloud and tokenization) (30 minutes)
- Performed some EDA - see Notebook (38 minutes)
- Performed Text Preprocessing in similar format as was done by a Kaggle solution (link to copied code provided in notebook) (15 minutes)
- Created some of the features outlined above (22 minutes)
- Performed Standard Scaling (10 minutes)
- Wrote a very simple logistic regression model to test the new features created (as a baseline model) - got a F1 score of .5463 (down from .60) (25 minutes)

Time Spent (Since Last Meeting): 2 hours 50 minutes

To-Do's (For Next Wednesday):

- Figure out when you should be doing text preprocessing. In kaggle notebooks, they created features first, then preprocessed the text. This seems backwards to me
- Look into more preprocessing steps that can be done
- Look into more features that would be good to create
- Read into Deep Learning Models created by top solutions on Kaggle
- Try to improve logistic regression model and try other non-deep learning models (for now)

Feedback:

Week 4 (2/15/2023 - 2/22/2023):

Preprocessing Ideas (based on GitHub Solution):

- Remove Emojis (represented with UTF encoding) and other non-ASCII characters (encoding issues confusing the model), but find a key that can translate emojis into phrases, and add those in place of emoji
    - modern language models have built-in emoji parsing capabilities, so there is no need to create a word representation or encoding, as the model can read and understand the UTF encoding for the different emojis
    - Some emojis will work in model but some won't. Thus, just remove the emojis that do not get rendered correctly (are not seen as ASCII characters)
- Deal with misclassified instances (top solutions in Kaggle noticed this as well)
- Consider text length in instance
- Consider number of words in instance
- Multi-hot encoding of emojis
    - Multi-hot encoding in general?
- Back translation (Google Translate's APO, Excel, MarianMT, Helsinki-NLP (English→French→English,…)
- Convert all text to lowercase
- Keep stop words in so that the models have more text from which to derive context
- Remove punctuation entirely
- Remove links/URL entirely
- Remove html entirely
- Use abbreviations dictionary to replace abbreviations with their meanings in the text (do this before removing punctuation)

Completed:

- Read through all reports (Performance, Model, Data, Overall, etc.) and code (baseline, final, etc.) in GitHub solution the Professor provided me and took notes on things that I may want to do for my attempt (above)
- Compared solution with top solutions on Kaggle
- Read up on some of the techniques done by GitHub solution

Time Spent (Since Last Meeting): 2 hours 10 minutes

To-Do's (For Next Wednesday):
- Apply new preprocessing steps discussed in GitHub solution
- Place code in GitHub repository

Feedback:

Week 5 (2/22/2023 - 3/1/2023):

Completed:
- Added relevant code from baseline Python notebook I made prior into preprocessing and EDA notebooks
- Did some debugging
- UTF-8 encoded the dataset
- Added in a few of the preprocessing steps from GitHub solution:
    - Removed emojis
        - Spent a lot of time also trying to figure out if emojis existed in the dataset in the first place
    - Removed non-ASCII characters
    - Converted text to lowercase
    - Kept in stop words
    - Removed punctuation (HAS CAUSED ISSUES)
    - Removed URLs
    - Removed HTML
    - Replaced Abbreviations

Time Spent (Since Last Meeting): 2 hours 30 minutes

To-Do's (For Next Wednesday):
- Fix punctuation problem
- Deal with trailing whitespace as well as whitespace within text (no more than one space should exist between words)
- Do back translation
- Figure out if you should drop duplicate rows
- Figure out if you should add keyword as an additional word in text
- Finish feature creation step
- Finish dealing with mislabeled samples after all preprocessing

Feedback:

Week 6 (3/1/2023 - 3/8/2023, Partial Spring Break):

Completed:
- Took a deep dive into dataset after first cleaning steps to see what might still be wrong
    - Also what changed to cause there to be more duplicate mislabeled instances now
- Fixed punctuation problem by just removing all punctuation

- Dealt with trailing whitespace as well as whitespace within text (no more than one space now exists between words)
- Created additional cleaning function (regex) to represent manual cleaning I decided should be done after doing examination of first 500 rows in "clean draft" dataset

Time Spent (Since Last Meeting): 2 hours
To-Do's (For Next Wednesday):

- Do back translation
- Figure out if you should drop duplicate rows
- Figure out if you should add keyword as an additional word in text
- Finish feature creation step
- Finish dealing with mislabeled samples after all preprocessing

Feedback:

Week 7 (3/8/2023 - 3/15/2023, Partial Spring Break):
Completed:

- Dealt with all mislabeled samples (had to do some research and create multiple functions to get this done as there were a lot of row instances to go through)
- Made versions of the datasets with mislabeled samples relabeled and not relabeled
- Made versions of the datasets with and without duplicates
    - It seems like we get higher performance with duplicates still in dataset, but I'm pretty confident that is just overfitting because there are almost 1000 duplicate rows
- Started adding basic code to "Model" notebook
    - Performed count vectorization and used a ridge regression model to see how performance compares to baseline performance
    - Mislabels and duplicates still in: Scores: [0.61169102 0.54718876 0.60039761]
    - Mislabels in but no duplicates: Scores: [0.5789153 0.53946621 0.59375]
    - No Mislabels but duplicates still in: Scores:[0.62434418 0.54313425 0.62040816]
    - No Mislabels and no duplicates: Scores: [0.5767098 0.53946621 0.5949214]
    - Added all work and file structure to Cloud/GitHub repository
    - If I were to guess, I should probably still remove duplicates, even though the test set has duplicates too
        - ASK PROFESSOR

Time Spent (Since Last Meeting): 2 hours
To-Do's (For Next Wednesday):

- Do back translation
- Figure out if you should add keyword as an additional word in text
    - For punctuation and numbers, remove but add space in place, then remove additional whitespace
- Finish feature creation step

- If you get above steps done, start writing model
- Check performance after: adding punctuation back in (or just some punctuation), adding keyword as extra word

Feedback:

Week 8 (3/15/2023 - 3/22/2023):

Completed:

- Began writing out Deep Learning model for problem (including tokenizer)

Time Spent (Since Last Meeting): 2 hours 30 minutes

To-Do's (For Next Wednesday):

- Fix bugs with model
    - Issues with versions most likely
        - Downgrade python to 3.6 or 3.7?
- Write an outline of final report(s)

Feedback:

Week 9 (3/22/2023 - 3/29/2023):

Completed:

- Continued working on bugs with models
- Fixed model but crashed due to usage of CPU instead of GPU or TPU

Time Spent (Since Last Meeting): 2 hours (mostly debugging)

To-Do's (For Next Wednesday):

- Get code working on HPC or Kaggle TPU

Feedback:

- Start writing portion of project

Week 10 (3/29/2023 - 4/5/2023):

Completed:

- With help of Pete, got code to run on HPC, but later ran into issue with bert_layer being downloaded from the internet
- Set up code on Kaggle to run on TPU and also to get a submission on Kaggle finally

Time Spent (Since Last Meeting): 3 hours

To-Do's (For Next Wednesday):

- Write Data and Performance Report

Feedback:

Week 11 (4/5/2023 - 4/12/2023):

Completed

- Wrote most of data report

Time Spent (Since Last Meeting): 2 hours

To-Do's (For Next Wednesday):
- Finish data report
- Start model and performance report

Feedback:

Week 12 (4/12/2023 - 4/19/2023):

Completed:
- Data report
- Started outline of model and performance report

Time Spent (Since Last Meeting): 2 hours

To-Do's (For Next Wednesday):
- Finish model and performance report

Feedback:

Week 13 (4/19/2023 - 4/26/2023):

Completed:
- Finished model and performance report

Time Spent (Since Last Meeting): 2 hours

To-Do's (For Next Wednesday):
- Organize github
- Create markdowns
- Upload everything to GitHub

Feedback:

Week 14 (4/26/2023 - 5/3/2023):

Completed:
- Created PowerPoint presentation of my work
- Uploaded presentation to GitHub
- Created markdowns of the two reports and uploaded to GitHub
- Added PDFs of two reports and this document to GitHub
- Created READMEs of all sections of repository

Time Spent (Since Last Meeting): 6 hours

To-Do's (For Next Wednesday):
- PROJECT COMPLETED

Feedback: