JSC270 Assignment 4

Natural Language Processing (with Twitter data)

March 8, 2021

Note: This assignment is designed to be completed in groups of 2. If class enrolment near the end of semester does not permit even division into pairs, then we may allow some groups of 3. Once you have your team, please add your information to the attached spreadsheet. Once groups are finalized, a presentation schedule will be created.

Note: For Question 2, obtaining approval for use of the Twitter API takes around 2 days. Please plan accordingly when completing the assignment.

Submission Deadline (Report/Analysis): Friday, April 9th at 11:59pm EST

Presentation Dates: Monday April 5th, between 1:00PM and 3:00PM EST (during class), and Wednesday April 7th, between 12:00PM and 2:00PM EST (during lab)

What and where to submit:

- A PDF report containing your responses, including any figures and derivations. **In the beginning of the report, please specify the breakdown of your work**. For example, if two of you did most of the analysis, and another was mostly responsible for the presentation, include this information.
- Colab Notebook(s) containing any relevant code
- Slides for the presentation of Question 2 (in PDF format)
- Your custom Twitter data for Question 2 (in CSV format)

All materials for this assignment are to be submitted via **GitHub**. Only one submission is required for the group. Before getting started, please do the following:

1. Create a GitHub repository for this assignment. You may call it whatever you wish, but please use something sensible (ie JSC270A4). Make sure all project partners and both of your TAs (Lauren: `larunerdman` and Matt: `12mre1`) are also added as contributors.
2. Data extraction with the Twitter API will involve the use of an individual private key (more on this in Q2). For this reason, please make sure your repo is private.
3. In the spreadsheet of groups listed above, please add your group information and the URL for your repo.

Grading Scheme: This assignment contains 3 sections, for a total of 55 marks, with 1 optional bonus point.

1. Sentiment Analysis with a Common Twitter Dataset (20 pts)

For this question, you'll be using a Twitter dataset containing just under 45,000 tweets related to COVID-19. These data come from a fairly recent Kaggle competition, which you can read more about here. To help with modeling, we've simplified the data in the following ways:

- The tweets are split into training (`covid-tweets-train.csv`) and test sets (`covid-tweets-test.csv`). The training set contains 41197 observations, while the test set contains 3798. This corresponds to (roughly) a 91.5/8.5 split. These files are available in the assignments section of the course website.
- The dataset now contains only the tweets (as strings) and labels $y_i \in \{0, 1, 2\}$, corresponding to sentiment {negative, neutral, positive}.

In this question, you'll be training classifiers to predict whether the tweet is positive, negative, or neutral, based only on the tweet itself. This is called *sentiment analysis*, and it is a common ML problem. To give you a sense of how sentiment is labelled, below are tweets from each of the three classes:

> **Positive**: *Hey guysss! We have a very important message to all our friends, family amp; viewers out there. Be aware of Corona Virus and follow these procedures to help keep you and your family safe.#aadyasitara? #aadya #sitara #coronavirus #coronaalert covid_ 19 #staysafe #sanitizer #covid https://t.co/pHWGsygy06*
>
> **Neutral**: *The top food items to stock up on in case you are quarantined #Coronavirus home stays are on people's minds. Here are ten foods for stocking a pantry to support physical and mental health. https://t.co/jrvxRIb57L*
>
> **Negative**: *Just thinking if we have to close schools, how many children may lose access to main source of meals. And what will happen to demand at food banks. #trusselltrust #coronavirus #foodbanks #poverty #uk https://t.co/Mw71RzC6z1*

Note: Each of the preprocessing steps that follow should be applied to both the training and test sets. You may find it useful to write these steps as functions that can be applied to both datasets. None of these preprocessing steps should require more than a few lines of code.

A) (1 pt) Consider the training data. What is the balance between the three classes? In other words, what proportion of the observations (in the training set) belong to each class?

B) (1 pt) *Tokenize* the tweets. In other words, for each observation, convert the tweet from a single string of running text into a list of individual tokens (possibly with punctuation), splitting on whitespace. The result should be that each observation (tweet) is a list of individual tokens.

C) (1 pt) Using a *regular expression*, remove any URL tokens from each of the observations. *Hint: In this dataset, all such tokens begin with 'http'.*

D) (2 pts) Remove all punctuation (,.?!;:'") and special characters(@, #, +, &, =, $, etc). Also, convert all tokens to lowercase only. Can you think of a scenario when you might want to keep some forms of punctuation?

E) (1pt) Now stem your tokens. This will have the effect of converting similar word forms into identical tokens (e.g. run, runs, running → run). Please specify which stemmer you use. *Note: There are several different stemmers available through* `nltk` *and* `Scikit-learn`. *I recommend the Porter stemmer, but you may use a different one if you wish.*

F) (1pt) Lastly, remove stopwords. Using the english `stopwords` list from `nltk`, remove these common words from your observations. This list is very long (I think almost 200 words), so remove only the first 100 stopwords in the list.

G) (2 pts) Now convert your lists of words into vectors of word counts. You may find Scikit-learn's `CountVectorizer` useful here. What is the length of your vocabulary? *Hint: The matrix of counts will be* $D \times V$*, where* D *is the number of documents (tweets), and* V *is the number of features (word counts).*

H) (4 pts) Recall the definition of the Naive Bayes model. If each document (tweet) is a collection of words $(w_1, \cdots, w_N)$ belonging to class C_k $(k = 0, 1, 2)$, then the Naive Bayes approach models the probability of each tweet belonging to class k:

$$\begin{aligned} P(C_k|w_1, \cdots, w_N) &\propto P(w_1, \cdots, w_N|C_k)P(C_k) \\ &= P(C_k)\Pi_{i=1}^{N} P(w_i|C_k) \end{aligned}$$

Note the last equality follows from our 'naive' assumption that words are conditionally independent given class. The probabilities are estimated using the frequencies of words within each class (bag of words), and we assign the class label according to which of the 3 posterior class probabilities $(P(C_k|w_1, \cdots, w_N))$ is the highest.

Fit a Naive Bayes model to your data. Report the training and test error of the model. Use accuracy as the error metric. Also, report the 5 most probable words in each class, along with their counts. You might find Scikit-learn's `MultinomialNB()` transformer useful. Use Laplace smoothing to prevent probabilities of zero.

I) (2 pts) Would it be appropriate to fit an ROC curve in this scenario? If yes, explain why. If no, explain why not.

J) (2 pts) Redo parts G-H using TF-IDF vectors instead of count vectors. You might find Scikitlearn's `TfidfVectorizer()` transformer useful. Report the training and test accuracy. How does this compare to the accuracy using count vectors?

K) (3 pts) Recall lemmatization converts each word to its base form, which is a bit stronger than simply taking the stem. Redo parts E-H using TF-IDF vectors instead of count vectors. This time use lemmatization instead of stemming. Report train and test accuracy. How does the accuracy with lemmatization compare to the accuracy with stemming? *Note: Like stemmers, there are multiple lemmatizers you might use. We recommend the WordNet lemmatizer offered by* `nltk`.

Bonus (1 pt): Is the Naive Bayes model generative or discriminative?

2. Open-Ended: NLP with the Twitter API (20 pts)

For this question, you'll extract your own Twitter data using Twitter's python API. The free version of this interface contains several different search parameters, but only allows for extraction of tweets from the past seven days (exactly). This means that each group's dataset will be unique.

You'll devise a research question that can be answered using your custom data (and possibly additional data if you require it). You'll then answer that question using a machine learning model of your choice. The analysis should be guided by the five components listed later in the section.

The Twitter Python API

The python wrapper for Twitter's API is called `tweepy`. The standard(free) version allows you to extract tweets from up to 7 days ago, with a limit of 18,000 tweets in a 15 minute window. In order to use this package, you'll first need to obtain some credentials from Twitter:

1. You'll need to sign up for a Twitter Account. Even if you already use Twitter, I recommend creating a separate account for this project.
2. You'll need to apply to be a Twitter developer. To do so, visit `https://developer.twitter.com/en/apps` and click the `Apply` button in the top right corner. You can choose the `Academic` → `Student` option. The default is to sign up for a team account. This allows multiple users (ie. your and your project partner).

   The application will ask questions about the nature of your research project. Be honest and thorough. Note that you will not be required to 'display aggregate twitter data outside of twitter', so you can select NO for this option. After agreeing to terms and conditions, you'll submit your application. After verifying your email (check your inbox), Twitter typically takes a day or two to review and approve your developer status.
3. You'll need to create a new app. Going back to site listed above (assuming you're signed in to your account), you can create a new application on the developer dashboard (select `create project`). When you do this, Twitter will automatically generate 4 key pieces of information for your project:
   - A Consumer API Key
   - A Consumer API Secret Key
   - An Access Token, and..
   - An Access Token Secret

   Record this information somewhere secure. We'll use it to connect to Twitter through `tweepy`. Note that the developer dashboard will only show you these 4 pieces of information once. For security, if you ever want to view them again, Twitter will generate new ones for you.

Once these three steps are complete, you'll be ready to work with the Twitter API directly in python. How to do this will be covered in depth during the Lab on March 10th.

Analysis Structure

Although each group will have a different analysis, the general structure should look the same. Specifically, you should cover each of the following 5 components:

(I) Problem description and motivation. Describe the question you will answer. Why is this question difficult to solve? How will your dataset help solve it? Are there any literature related to existing approaches? Why will your analysis be different? Make sure you provide proper citation of any other approaches you reference in your analysis. *Note: It's okay if another analysis has taken an approach similar to yours. You'll have a strong baseline comparison for your results.*

(II) Describe the data. Describe the data you've extracted. What exactly were the the extraction parameters you used in the API, and why? State the number of observations and the number of features. Have other similar data been analysed, and if so, how? Did other similar works of research generate conclusions similar to yours? What are the limitations of your data? *Note: If, due to computational constraints, you can only carry out your analysis using a subset of the data you extract, that's okay. Just be clear about which data you selected and why.*

(III) Exploratory data analysis. Describe the basic statistics and properties of your data. Perform any necessary preprocessing steps to ensure your data is ready for model-fitting. Many of these steps will probably be similar to what was done in Q1 (that's okay). You might generate new features based on the raw text. This is a good opportunity to generate some visualizations for your presentation. Report any interesting findings.

(IV) Describe your machine learning model. Identify the machine learning task you'll be using to answer your question. Explain how the model works. Why is this model an appropriate choice for your research problem? Is this a supervised or unsupervised model? What are its strengths and weaknesses? How will you evaluate the model's performance? Are there any baseline models to which you can compare your results? You can use any of the models discussed in class, but you are not constrained to just those (some additional examples will be listed later in this section). *Note: For supervised learning models, you will probably have to generate the labels yourself.*

(V) Results and Conclusions. Describe the results of your analysis. How well does your model perform? Did you run into any issues (e.g. overfitting), and if so, how did you manage them? What is the significance of your results? Do your model parameters have a meaningful interpretation? Why did your model work better than existing approaches (or if it failed, why did it fail)? If you had more time, or could collect more data, what would you do differently?

Although you may format your submitted responses however you wish, you may find it helpful to include separate sections for each of these components in the Q2 section of your report.

Additional Topics

In addition to what will be covered in lectures and labs, there are many other Natural Language Processing models you can choose from. Here are some ideas for groups looking to try something new:

- **Latent Dirichlet Allocation**. This is a generative unsupervised approach designed to group documents into topics, where topics are simply collections of similar words. In this setting, the number of topics is a hyperparameter, and the topics themselves depend on the data. This is a good method for finding themes/topics within large bodies of text. You can learn more from the original paper here.

- **Language Models**. Language is often considered 'self-labelled', because even if we don't have any other features, the sequential nature of raw text always allows us to predict the next word given the previous words in a sentence. Because of the complexity involved in predicting any possible sequence of words, these models are often high in complexity (e.g. random forests, but usually neural nets).
- **Text Features for Non-text Prediction**. Another common approach is to use a text classification task (e.g. sentiment analysis) to derive features that can be used as input in another machine learning setting. A classic example is the use of sentiment analysis of business news articles to predict changes in stock prices.

If you have other ideas for your analysis, feel free to discuss them with one (or more) of the teaching team during office hours (or you might schedule a separate appointment). This list is not exhaustive, and we encourage creativity.

3. Presentation of Results (15 pts)

During the either last lecture (Mon, April 5th) or last lab (Wed, April 7th), groups will be required to give a brief presentation of **up to 5 minutes** (no longer). This presentation will cover the analysis in Question 2 (Question 1 should not be presented; include Q1 results in the written report). Following each presentation, there will be a brief 1-2 minute question period. Each group will be graded based on how well their presentation covers each of the five components described in Question 2 **(1 pt per section, for 5 pts total)**.

Additionally, students will be graded on the quality of the presentation. Specifically, we will be looking for:

- Volume and pacing of speech **(1 pt)**
- Coherence and organization of slides **(2 pts)**
- Clarity of content **(2 pts)**
- Quality of visuals (e.g. graphs, tables) **(2 pts)**
- Concluding within the time limit **(1 pt)**
- Effectively answering questions **(2 pts)**

Near the end of March (roughly 2 weeks before presentations), we will circulate a presentation schedule, where groups can sign up for specific time slots.