GIFTS: Natural Language Processing with MATLAB

Michael Shteynberg, Northeastern University
Dr. Andrew L Gillen, Northeastern University

Andrew L. Gillen is an Assistant Teaching Professor at Northeastern University in the First Year Engineering Program and an affiliate faculty member to Civil and Environmental Engineering. He earned his Ph.D. in Engineering Education from Virginia Tech and B.S. in Civil Engineering from Northeastern University.

Dr. Kaylla Cantilina, Northeastern University

Kaylla Cantilina is an Assistant Teaching Professor in the First Year Engineering program at Northeastern University. While her teaching is centered around supporting student holistic growth, culturally sustaining pedagogies, and intertwining sociotechnical content in engineering education, her research is motivated by design as a means for social justice, and making engineering more accessible and supportive for minoritized students. She has a deep interdisciplinary background with degrees in industrial design, political science, industrial operations engineering, and design science, and earned her Ph.D. from the University of Michigan.

GIFTS: Natural Language Processing with MATLAB

Motivation and Objectives

Natural Language Processing (NLP) is a sub-branch of artificial intelligence that uses machine learning. It allows machines to understand, analyze, and generate responses that are easy for humans to understand. NLP already facilitates the interactions between our students and all sorts of artificial intelligence like chatbots (ChatGPT), smart assistants (Siri), and more. Calls for more integration of artificial intelligence into education grow louder by the day. For instance, a special committee was established in the US to make recommendations, including around AI in education [1]. Outside of academia, regular interaction with AI tools is becoming commonplace in industry. Scholars have already outlined a plethora of opportunities and concerns around applying this technology in the engineering classroom [2]. However, if we are to train the next generation of engineers, providing a background in NLP could help students better understand the potential and limitations of these powerful tools [3] . Since many first-year engineering programs teach MATLAB, python, or similar programming languages, it is a natural home for exploring the topic of NLP with students to lift the veil of mystery around the technology and provide practical applications of their coding knowledge.

This lesson can be used to guide students through implementing Natural Language Processing in MATLAB. At the end of the lesson, students should be able to (a) articulate the major steps needed to create a simple NLP program and the associated terminology as well as (b) apply these steps to an example using real-world data.

Practical Implementation Details

The following lists the student procedures for each step of the lesson. This could be guided step-by-step with the instructor, or assigned as a stand-alone module for students to complete outside of class in an introductory course in which MATLAB is taught. The corresponding code that goes with these instructions can be found in the Appendix of this paper. We suggest that students follow along in the example code while working through each step.

Step 1: Opening MATLAB and Installing Add-Ons

MATLAB is a computational tool for scientists and engineers. It can be used to work with and visualize large data sets. While it has its own programming language, it is more human readable than the more fundamental languages. MATLAB is a proprietary software but there are open source versions available such as Octave. Students do not need prior experience with MATLAB for this module, but should have prior classroom experience with the general syntax.

The student procedure for Step 1 entails: (1) opening MATLAB, (2) creating a new script file, (3) selecting the Home tab, (4) navigating to the Environment Block and selecting Add-Ons, (5) in the search bar, typing “Text Analytics Toolbox,” (5) selecting and installing this add-on.

Step 2: Loading in Desired Text

In order to practice natural language processing, the students need to be provided or identify text to use. In the example code shown in the appendix below, the text used is the Happy Birthday song stored in “textData.”

Step 3: Tokenization

Using “tokenizedDocument” splits the sentence in “textData” from Step 2 into individual words – this is called tokenization.

Step 4: Remove Stop Words

“RemoveStopWords” is a simple function that removes words like "a", "and", "to", and "the" from the tokenized sentence in Step 3.

Step 5: Lemmatization

“NormalizeWords” lemmatizes each word to its base form. When the style is set to lemma, NormalizeWords completes this step by checking each word with a dictionary-like list and replacing it with its base form. For example, "running" becomes "run" or "better" becomes "good."

Step 6: Sentiment Analysis

Students need to define the “positive” and “negative” words they want to use. This can be any list of words. They now need to convert the “tokens” variable into a string so it is easier to work with in future steps. The “ismember” function looks at all the words in the “tokenArray” string and compares it to the words they put in the positive/negative words list. It then sums this and counts the number of words that match. Now they need to subtract the number of negative words from the number of positive words and store it in sentimentScore. If the number is positive, the sentiment is positive and vice versa (if the number is 0, this either indicates a neutral sentiment or not having any positive/negative words. Lastly, they can display the sentiment score they calculated using the display command.

Step 7: Bag of Words

“BagOfWords” or “term-frequency counter” is a model that counts the number of times a word appears in a text regardless of order. Topkwords(bag,k) is another function that takes the top words from the “bagOfWords” model. Here we use 5 as k to look for the top 5 words used. Again we use display to show the top 5 words used in the text stored in “tokens.”

Step 8: Classifying Text

This is setting up another way to classify text as positive or negative. Students can write any series of positive/negative statements in the data section. We recommend putting 9 statements.

“Labels” is defined as an array where 1 represents a positive statement and 0 represents a negative statement. Same as before, this tokenizes the statements made in “data.”
Using “BagOfWords” counts the frequency of each word in the text of “tokenizedData.” “Full” converts the sparse matrix “bag.Counts” into a full/dense matrix. “Fitctree” is a function that uses a decision tree classifier to classify the text given from “fullCounts” as positive or negative

In “newTextData,” students can input any new text. “TokenizedDocument” again tokenizes the new text inputted. “Encode” takes the new text from “newTokens” and converts it into a bag of words model using the previously defined “bag” model. “Full” again turns the newly encoded data into a full matrix

“Predict” uses mdl which is a trained decision tree model on the new text and makes predictions based on the text. In this case the predictions will be 1 if it is a positive statement and 0 if it is a negative statement. Lastly, students can display the predictions made by the trained decision tree model.

Step 9: Visualizing NLP

“Figure” creates a figure to visualize different things in Matlab. “Wordcloud” creates a word cloud from tokens (word cloud is a visualization that depicts the frequency of words by making more frequent words bigger). “Title” sets the title to whatever text is in the quotes.

Assessment Methods

The following provides a full problem that students can undertake to analyze the sentiment of datasets found online.

Table 1. Steps and Description of Tasks to Analyze Online Dataset Sentiments

Steps	**Task Descriptions**
Collect Data	Find 5 – 10 datasets from public websites. These can be any type of review like movies, books, restaurants, etc. For example, *Rotten Tomatoes, Goodreads, and Yelp.*
Prepare the Reviews	Use the techniques from the module to prepare the reviews you gathered: ○ tokenizedDocument(textData); ○ removeStopWords(tokens); ○ normalizeWords(tokens, 'Style', 'lemma');
Calculating Sentiment	1. Create a list of predefined positive and negative words that you believe would fit in the reviews you gathered. 2. Use “sum” to calculate the number of positive vs. negative words. 3. Finally calculate the sentiment score of the datasets gathered. 4. Display the sentiment score.

Calculate Frequency	1. Use “bagOfWords” to calculate the frequency of different words used in the reviews you found. 2. Display the frequency counter.
Visualization	Create a word cloud that displays the words found in the reviews and how often they appear.

References

[1] National Artificial Intelligence Advisory Committee. (2024). Retrieved October 30, 2024, from https://ai.gov/naiac/

[2] Menekse, M. (2023), Envisioning the future of learning and teaching engineering in the artificial intelligence era: Opportunities and challenges. J Eng Educ, 112: 578-582. https://doi.org/10.1002/jee.20539

[3] Qadir, J. (2023). Engineering education in the era of ChatGPT: Promises and pitfalls of generative AI for education. In *2023 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1-9). Kuwait, Kuwait. https://doi.org/10.1109/EDUCON54358.2023.10125121

Relevant MathWorks Documentation

MathWorks. (2024). Machine Learning in MATLAB. Retrieved October 30, 2024, from https://www.mathworks.com/help/stats/machine-learning-in-matlab.html

MathWorks. (2024). Text Analytics Toolbox. Retrieved October 30, 2024, from https://www.mathworks.com/products/text-analytics.html

MathWorks. (2024). Decision Trees. Retrieved October 30, 2024, from https://www.mathworks.com/help/stats/decision-trees.html

MathWorks. (2024). Predict. Retrieved October 30, 2024, from https://www.mathworks.com/help/stats/linearmodel.predict.html

MathWorks. (2024). Wordcloud. Retrieved October 30, 2024, from https://www.mathworks.com/help/matlab/ref/wordcloud.html

MathWorks. (2024). Encode. Retrieved October 30, 2024, from https://www.mathworks.com/help/comm/ref/encode.html

MathWorks. (2024). Full. Retrieved October 30, 2024, from https://www.mathworks.com/help/matlab/ref/full.html

MathWorks. (2024). Fitctree. Retrieved October 30, 2024, from https://www.mathworks.com/help/stats/fitctree.html

MathWorks. (2024). Topkwords. Retrieved October 30, 2024, from https://www.mathworks.com/help/textanalytics/ref/bagofwords.topkwords.html

MathWorks. (2024). BagOfWords. Retrieved October 30, 2024, from https://www.mathworks.com/help/textanalytics/ref/bagofwords.html

MathWorks. (2024). IsMember. Retrieved October 30, 2024, from https://www.mathworks.com/help/matlab/ref/ismember.html

MathWorks. (2024). NormalizeWords. Retrieved October 30, 2024, from https://www.mathworks.com/help/textanalytics/ref/normalizewords.html

MathWorks. (2024). Visualize Text Data Using Word Clouds. Retrieved October 30, 2024, from https://www.mathworks.com/help/textanalytics/ug/visualize-text-data-using-word-clouds.html

Appendix

Corresponding MATLAB Code:

```
textData = ["Happy Birthday to you, Happy Birthday to you, Happy Birthday dear name, Happy Birthday to you!"];

tokens = tokenizedDocument(textData);

tokens = removeStopWords(tokens);

stemmedTokens = normalizeWords(tokens, 'Style', 'lemma');

positiveWords = ["happy", "birthday", "party", "presents", "balloons"];
negativeWords = ["bad", "terrible", "awful", "hate", "annoying"];

tokenArray = string(tokens);

positiveCount = sum(ismember(tokenArray, positiveWords));
negativeCount = sum(ismember(tokenArray, negativeWords));

sentimentScore = positiveCount - negativeCount;

disp('Sentiment Score:');
disp(sentimentScore);

wordCounts = bagOfWords(tokens);
topWords = topkwords(wordCounts, 5);

disp('Top Words:');
disp(topWords);

data = ["I love Birthdays", "Presents are amazing", "I hate surprise parties", "Unwrapping presents is annoying",...
        "Candles are fantastic", "I dislike having to write 'thank you' notes", "Setting up is challenging", "The party is great",...
        "I enjoy celebrating with friends"];

labels = [1 1 0 0 1 0 0 1 1];

tokenizedData = tokenizedDocument(data);

bag = bagOfWords(tokenizedData);

fullCounts = full(bag.Counts);
```

```
mdl = fitctree(fullCounts, labels);

newTextData = ["Birthdays make me feel happy and excited"];
newTokens = tokenizedDocument(newTextData);

newBag = encode(bag, newTokens);

fullNewCounts = full(newBag);

predictions = predict(mdl, fullNewCounts);

disp('Prediction for New Text Data:');
disp(predictions);

figure;
wordcloud(tokens);
title('Word Cloud');
```