Immersive Python Workshop

August 15-16, 2024

Introduction to Sentiment Analysis

Ian Goodale
European Studies Librarian

TEXAS Libraries
The University of Texas at Austin
University of Texas Libraries

Presented by the
UT Libraries &
Open Source Program Office (UT-OSPO)

UT-OSPO
UT Austin Open Source Program Office

Sloan grant number: G-2023-20944

Overview

1. Introduction
2. Key Terms
3. Challenges
4. Common Approaches
5. Technology We'll Use
6. Survey of Other Tools
7. Coding Exercises

Introduction

- What is sentiment analysis?
  - Sentiment analysis provides a quantitative assessment of the emotional intensity of words and phrases within a text. Sentiment analysis tools generally process a given unit of text (a sentence, paragraph, etc.) and output quantitative scores or classifications to indicate whether the algorithm being used considers that text to convey positive or negative emotion. The tools we will use today can also quantify the degree of positivity or degree of negativity within a text.

- In short, sentiment analysis offers us a way to quantitatively assess the emotions expressed in a written form

Introduction (cont’d)

- Why perform sentiment analysis?
  - Analyze large amounts of data and return an easily understood output
  - Gain new insights into a dataset
  - Generate new ideas for analysis of a dataset
  - Produce visualizations

Key Terms

- Rules/lexicon-based analysis: A type of analysis that pre-categorized lists of words and phrases (lexicons) to label the words (e.g. positive, negative, or neutral) and detect sentiment.

- Machine learning based analysis (e.g., the Naive Bayes algorithm): A type of analysis that uses algorithms trained on pre-labeled texts to infer the sentiment of a given dataset of unlabeled text(s).

Common Challenges

- (In)accuracy of the technology
- Working with imprecise or complex textual data
- Multilingual texts

Common Approaches

- Using a pre-trained model available from an existing package
- Training your own model
- Different technological approaches:
  - Machine learning
    - Example: the Naïve Bayes classifier
  - Rules-based models
    - Example: Vader, which uses both lexicon and linguistic rules for sentiment classification

Technology We'll Use

- Pandas
- NLTK
  - Including its Naïve Bayes classifier, Vader sentiment analysis package (rules/lexicon-based), and tokenizers
- TextBlob
- WordCloud
- Matplotlib

Pandas

- An open source data analysis and manipulation tool for Python
- Uses a data structure called a data frame
  - These can be easily constructed from imported files (e.g., a JSON or CSV file)
- It also allows for easily manipulation and analysis of the data in these DF structures

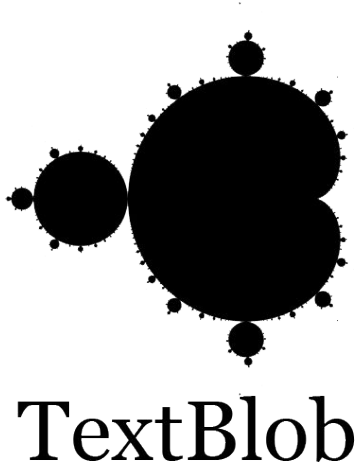
Natural Language Toolkit (NLTK)

- Platform for building Python programs to work with language data
- Large amount of corpora and lexical resources in addition to Python libraries
- Great documentation makes this easy to learn and experiment with
- Can easily be used with other Python libraries and tools

```
# Now loop over each line and tokenize it separately
        for sentence in sent_text:
            tokenized_text = nltk.word_tokenize(sentence)
            tagged = nltk.pos_tag(tokenized_text)

            languages_ratios = {}

            tokens = wordpunct_tokenize(sentence)
            words = [word.lower() for word in tokens]

            for language in stopwords.fileids():
                stopwords_set = set(stopwords.words(language))
                words_set = set(words)
                common_elements = words_set.intersection(stopwords_set)

                languages_ratios[language] = len(common_elements) # language "score"

                ratios = languages_ratios

                most_rated_language = max(ratios, key=ratios.get)

                language = most_rated_language


            if language == "danish":
                danish_sent.append(sentence)
                print language

            elif language == "english":
                english_sent.append(sentence)
                print language

            elif language == "french":
                french_sent.append(sentence)
                print language


        if danish_sent:
            dan_file = open("danish.txt", "w")
            for item in danish_sent:
                dan_file.write(str(item) + '\n')

        if dutch_sent:
            dut_file = open("dutch.txt", "w")
            for item in dutch_sent:
```

TextBlob

- *TextBlob* is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, and more.



- Its sentiment analysis functionality is easy to use, and utilizes the Naive Bayes algorithm

- https://textblob.readthedocs.io/en/dev/

Matplotlib

- A plotting library for Python that can easily integrate with Pandas
- Offers a wide variety of plot types to choose from
- Its plots are easily customizable and exportable
- Examples can be viewed here: https://matplotlib.org/stable/plot_types/index.html

Survey of Other Tools

- Stanza
  - https://stanfordnlp.github.io/stanza/sentiment.html
- SciKit Learn
  - https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
- spaCy (for working with text)
  - https://spacy.io/usage/models

Survey of Tools (cont'd)

- Rozha (for working with text)
  - https://github.com/ian-nai/Rozha
  - Demo: https://colab.research.google.com/drive/108FvBVuNllMneNzdNLYSVbu91D2dKcGs?usp=sharing

- Beautiful Soup - Python package for parsing HTML and XML documents; useful for web scraping.
  - https://beautiful-soup-4.readthedocs.io/en/latest/

Thank you!

Contact: ian.goodale@austin.utexas.edu

Coding Exercise

- First, let's locally install our packages:
  - pandas
  - nltk
  - matplotlib
  - textblob

- Link to Colab notebook: https://colab.research.google.com/drive/1lU8RZCfNCQO-P_q4rRkJleNkKM94MUQR?usp=sharing