# ISA 414 – Managing Big Data

## Lecture 15 – Text Mining (Part II)

*Sentiment Analysis*

Dr. Arthur Carvalho

arthur.carvalho@miamioh.edu

MIAMI UNIVERSITY

# Announcements

- Midterm project
  - To be released on Tuesday, October 19th
  - Deadline: 11:59 pm on October 22nd
  - Individual or in groups of two
    - Send me an email in case you already have a defined group
    - Send me an email in case you want to be paired up with a random person
      - No guarantee there will be another person available

# Lecture Objectives

- Quick review of Assignment 3
- Learn descriptive-analysis techniques involving textual data
  - Sentiment analysis
- Learn how to build classifiers using textual data

# Lecture Instructions

- Download the notebook "*Lecture 15.ipynb*" available on Canvas
  - Open the above file with VS Code

- Download the following files to the same folder our notebook is:
  - starbucks.csv
  - IMDB.csv
  - negative-words.txt
  - positive-words.txt

# Text Mining

- Previous lecture
  - We learned one approach to preprocess textual data
    - Bag of words
      - DTM
  - Counting metrics
    - TF
    - IDF
    - TFIDF
  - Other preprocessing techniques
    - N-grams
    - Stop words

# Text Mining

- Today, we focus on data analysis
  - Textual data
  - Techniques:
    - Sentiment analysis
    - Statistical modeling (classification)

# Text Mining

- Background story #1: **Starbucks and refugees**
  - Starbucks pledged to hire 10,000 refugees on January 29th, 2017

*"There are more than 65 million citizens of the world recognized as refugees by the United Nations, and we are developing plans to hire 10,000 of them over five years in the 75 countries around the world where Starbucks does business. And we will start this effort here in the U.S. by making the initial focus of our hiring efforts on those individuals who have served with U.S. troops as interpreters and support personnel in the various countries where our military has asked for such support."*

Howard Schultz (CEO)

  - Several people voiced their opinions on social media
    - From support to boycott calls
    - But we are not going to the same in this course
      - Your opinions on this matter are yours
      - Here, we are unbiased/unopinionated data scientists

# Text Mining

- Background story #1: **Starbucks and refugees**
  - Previous students taking the ISA 414 course decided to analyze how the sentiment towards Starbucks changed from before to after the announcement
    - Case study part of the paper "*Off-The-Shelf Artificial Intelligence Technologies for Sentiment and Emotion Analysis: A Tutorial on Using IBM Natural Language Processing*" available on Canvas


  - They collected posts on Starbucks Facebook page 14 days before to 14 days after the announcement
    - Let's work with that data, but with a shorter time window (01/29 to 01/31)
    - Data set "*starbucks.csv*"

# Text Mining

- Preliminaries
  - Load the required packages

    ```
    import pandas
    ```

  - Load the data set

    ```
    raw_data = pandas.read_csv("starbucks.csv")
    ```

  - Let’s now preprocess our data
    - We shall build a TF-based DTM, *i.e.*, no IDF
    - Why? In our first analysis, we only need to know whether a word is present in a text



MIAMI UNIVERSITY

# Text Mining

- Data preprocessing
  - Vectorizing data and creating a corpus

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer   = TfidfVectorizer(use_idf = False)
tf_values = vectorizer.fit_transform(raw_data["message"])
```

  - Recall that tf_values is a sparse matrix
    - Notation ($x$, $y$) $z$ means the score $z$ the word indexed by column $y$ in document $x$ receives
    - In our analysis, we are interested in knowing the precise words represented by column index $y$ in each document $x$ that have score $z > 0$

# Text Mining

- Data preprocessing
  - From our previous class, we learned about the function get_feature_names() that gives us all the words in our corpus
    - To get the words in a document, we have to subset the result from get_feature_names()
  - Example: let's retrieve all the words in the first document of our preprocess corpus

```
all_word_names  = vectorizer.get_feature_names()
doc0_word_index = tf_values[0,:].nonzero()[1]
doc0_word_names = [all_word_names[w] for w in doc0_word_index]
```

# SENTIMENT ANALYSIS

# Text Mining

- Sentiment analysis
  - Goal: obtain the overall sentiment (positive/negative) behind texts
    - Many different approaches that take linguistic aspects into account
    - In spirit, the approach we learn in this course is similar to the bag-of-words approach
      - Consider the difference between the number of positive and the number of negative words in a text

# Text Mining

- Sentiment analysis
  - Big picture
    - Break each text into individual words
    - Count the number of positive and negative words
    - Calculate the difference between the above counts
      - Score > 0: the overall sentiment behind a document is positive
      - Score < 0: the overall sentiment behind a document is negative
      - Score = 0: the overall sentiment behind a document is neutral

# Text Mining

- Sentiment analysis
  - List of positive and negative words available on Canvas
    - Officially called a **lexicon**
    - Thanks to Dr. Bing Liu
  - Let's start by loading these lists

```
file = open('positive-words.txt', 'r')
positive_words = file.read().splitlines()

file = open('negative-words.txt', 'r')
negative_words = file.read().splitlines()
```

# Text Mining

- Sentiment analysis
  - Example: estimating sentiment behind the 80th text
    - “*To the Starbucks Corporation, as the wife of a uniformed service member, I want to applaud your pledge to employ refugees. Thank you for the ray of hope in a frightening time, and for demonstrating humanity in the realm of business.*”

```
doc80_word_index = tf_values[79,:].nonzero()[1]
doc80_word_names = [all_word_names[w] for w in doc80_word_index]
count_positive_words = [positive_words.count(word) for word in doc80_word_names]
count_negative_words = [negative_words.count(word) for word in doc80_word_names]
score = sum(count_positive_words) - sum(count_negative_words)
```

# Text Mining

- Sentiment analysis
  - Example: estimating sentiment behind the 248$^{th}$ text
    - *"I have been a loyal consumer of Starbucks products for over 10 years. But unfortunately recent events and comments by upper management have disgusted me to the point that I will no longer use your products."*

```
doc248_word_index = tf_values[247,:].nonzero()[1]
doc248_word_names = [all_word_names[w] for w in doc248_word_index]
count_positive_words = [positive_words.count(word) for word in doc248_word_names]
count_negative_words = [negative_words.count(word) for word in doc248_word_names]
score = sum(count_positive_words) - sum(count_negative_words)
```

# Text Mining

- Sentiment analysis
  - The previous approach to estimate sentiment is rather naïve
    - *E.g.*, it does not take into account sarcasm
  - Nonetheless, it tends to work quite well with short texts like tweets (less so for Facebook posts)
  - There are many lexicons out there
    - SenticNet: http://sentic.net/senticnet-4.0.zip
    - MPQA: http://mpqa.cs.pitt.edu

# Text Mining

- A different approach to performing sentiment analysis is to build classifiers
  1. Manually label all the documents in a corpus
     - *I.e.*, to manually set the target variable as either positive or negative
  2. Build a model to predict the label (target)
  3. Use the trained model to determine the sentiment of new, unlabeled documents

- We explore this approach next

# STATISTICAL MODELING

# Text Mining

- Background story #2
  - Suppose that Rotten Tomatoes wants to include movie reviews from reviewers who do not explicitly report numerical ratings
  - Goal: develop a system that predicts whether a movie is “fresh” (positive review) or “rotten” (negative review) based on a textual review
    - Data: 1800 reviews <u>manually labeled (positive/negative) by an expert</u>

# Text Mining

- Let’s start by loading and processing the data
  - Note that we now build a TFIDF DTM, and we remove stop words to make our DTM smaller

```
import pandas
from sklearn.feature_extraction.text import TfidfVectorizer

raw_data = pandas.read_csv("IMDB.csv")

vectorizer   = TfidfVectorizer(stop_words = "english")
tfidf_values = vectorizer.fit_transform(raw_data['text'])
```

# Text Mining

- Since we will build a model (recall we didn't do it before), we need now to explicitly create a data frame

```
dtm = pandas.DataFrame(tfidf_values.toarray())
dtm.columns = vectorizer.get_feature_names()
```

- How many columns does dtm have?
  - Recall that there are only 1800 rows
  - Can you see how quickly a dtm can grow in size?
    - Look at today's notebook for the code that calculates the actual size

# Text Mining

- Thus far, we transformed the text column in the original data into a TF-IDF matrix, *i.e.*, we created a DTM
- Let's add a target (class) to our DTM
  - Recall that class was created by humans when manually labeling the data

```
dtm['target_var'] = raw_data['class']
```

- Our data frame now contains close to 37,500+ variables, including one target
  - Let's build a classifier (random forest)
  - No need to create dummies (why?)

# Text Mining

- Splitting data into training and test sets

```
from sklearn.model_selection import train_test_split

x = dtm.drop(columns=["target_var"])
y = dtm["target_var"]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.34)
```

# Text Mining

- Training a random forest (this might take a while)

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=200)
model = model.fit(x_train,y_train)
```

- Evaluating our model

```
from sklearn import metrics
y_pred = model.predict(x_test)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

# Text Mining

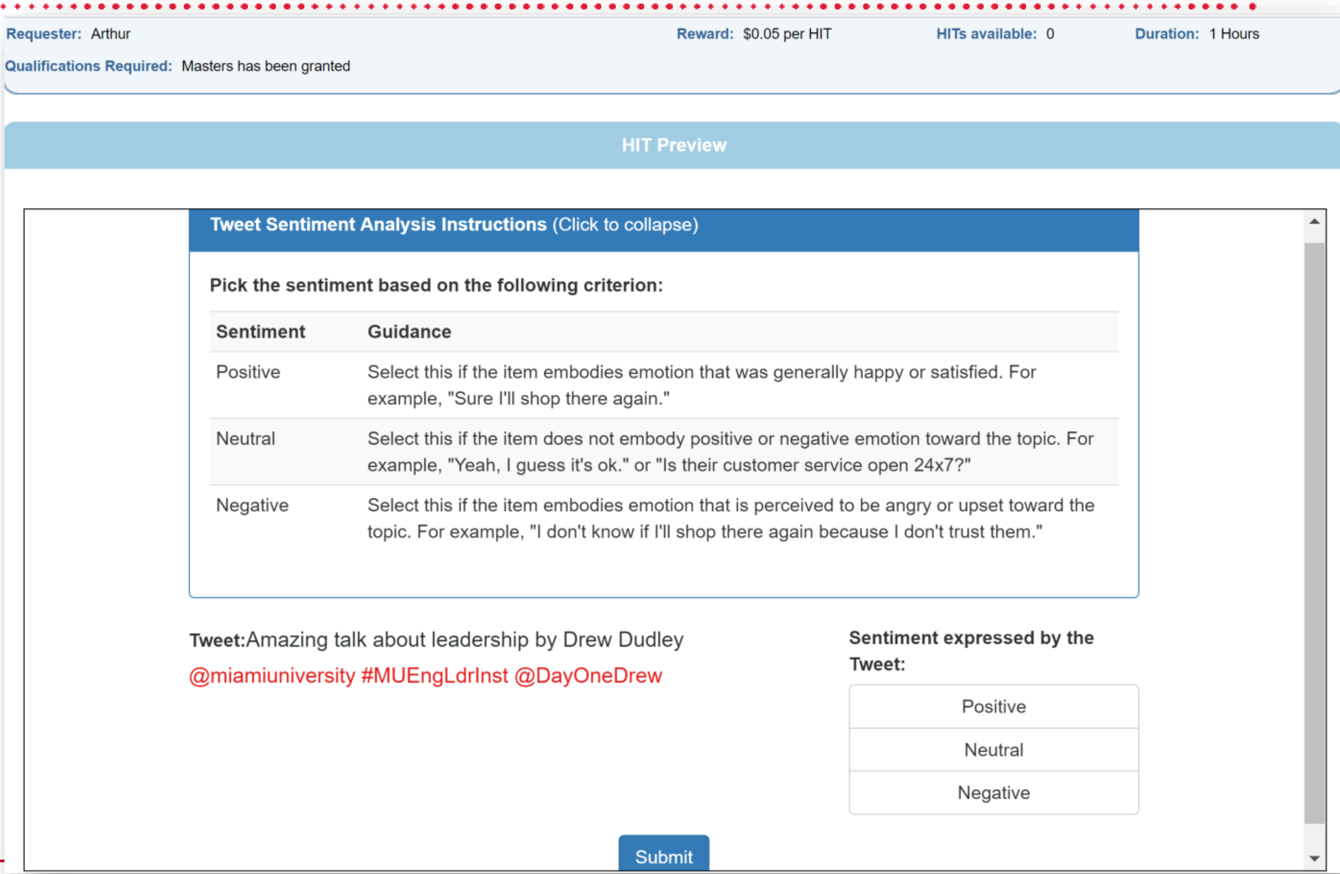
- The model's accuracy is around 73%-75%
  - Better than guessing
  - A lot of room for improvements
    - More powerful models (*e.g.*, more trees in the forest)
    - More meaningful variables
      - Manually remove uninformative words
    - Better preprocessing
      - *E.g.*, n-grams, stemming, …

# Text Mining

- Assumption in our previous analysis: target values are available
  - Manually created by an expert
    - Expensive and time consuming
  - Alternative: crowdsourcing
    - Non-experts
    - Platforms: Amazon Mechanical Turk and CrowdFlower
    - Quick, cheap, but data quality is often an issue

# Text Mining

- Example
  - Amazon Mechanical Turk



# Analysis of Unstructured Data

- We are focusing on texts, but there are many other forms of unstructured data
  - Image analysis
  - Video analysis
  - Speech analysis

# Homework #7

- Let's go back to the Starbucks problem
  - How many comments are positive?
    - Score > 0
  - How many comments are negative?
    - Score < 0

- Write a script that answer the above questions and report it on Canvas
  - Remember: do **NOT** use IDF when creating a DTM
  - Hint: write a FOR-loop that iterates over the length of raw_data

# Summary

- We learned a few techniques to analyze textual data
  - Sentiment analysis
  - Modeling

- Next class
  - Topic modeling