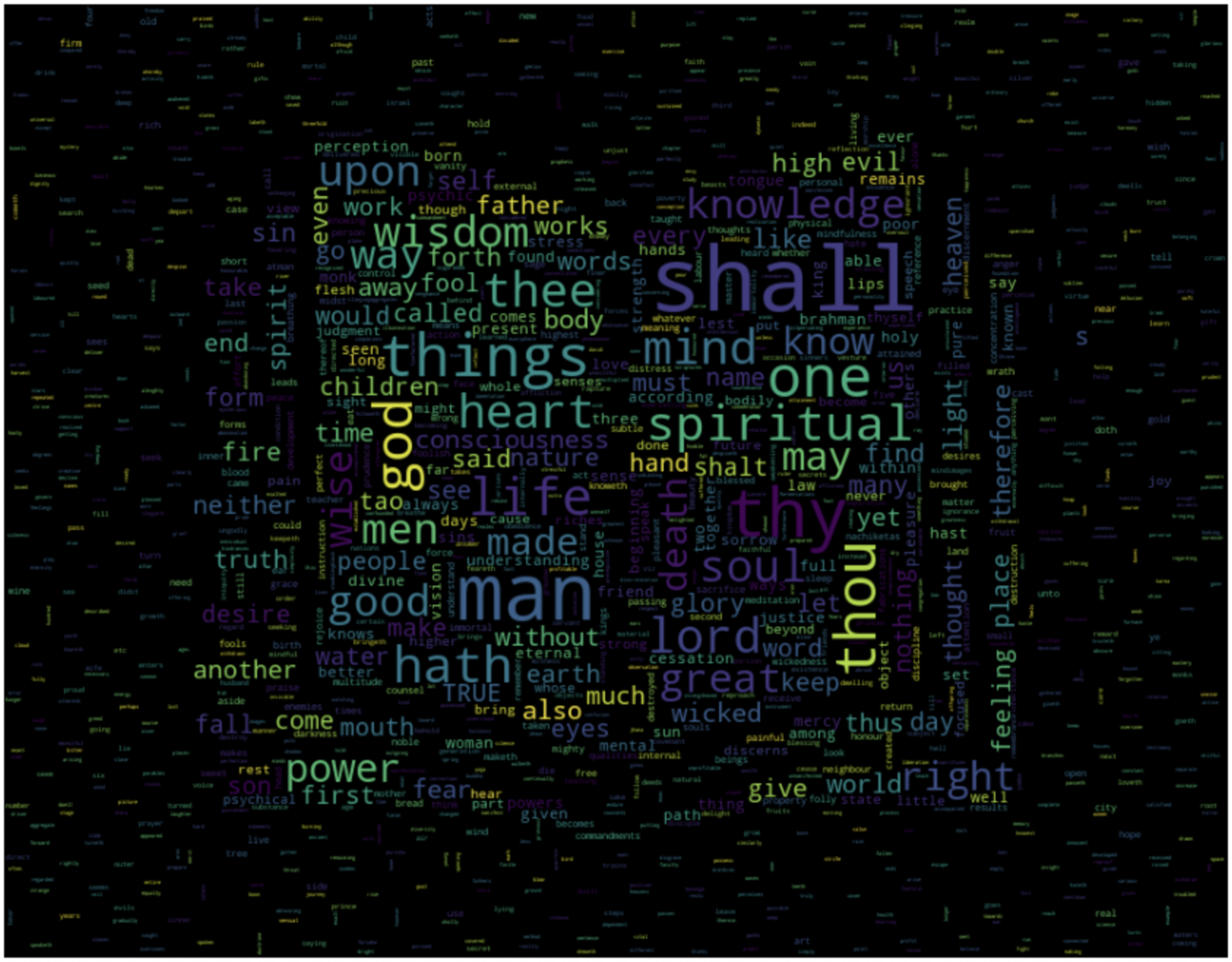
Word Cloud for all books



[Figure 1: A wordcloud of all words found within the religious texts data set, in which their relative size is equal to their overall frequency within the data set. Fun fact: This word cloud is shaped like a book! :D]

DATAHACKS 2021: BOOLEANS-R-US

Presented By: Jasmine Allred, Frans Timothy Juacalla,
Arjun Malleswaran and Tai Nguyen

Intro:

When presented with the religious texts data set, many questions arose: What are the most common words in the data set, how can these words be quantified to give the texts meaning, and how can future predictions be made given these texts? The solution to these questions came in many parts: Find and visualize the top twenty words of all books; Assign positive or negative values to every word to obtain a sentiment score; Use Naive Bayes' Classification Algorithm to predict which book a chapter is likely to fall into given its words. Using these solutions, production of word clouds which represent word frequency arose, a way to compare two texts such as the Buddhism and Taoism texts arose, and sentimental classification arose, along with the Bayesian classification of chapters to books.

Data Cleaning/Pre-Processing:

```
# fill missing values
data_train[:] = data_train.fillna(0)
data_test[:] = data_test.fillna(0)

# rename the unnamed column to Chapters, '# foolishness' to just foolishness
data_train.rename(columns = {'Unnamed: 0' : 'Chapters'}, inplace = True)
data_test.rename(columns = {'# foolishness' : 'foolishness'}, inplace = True)

# make sure everything is an integer
data_test = data_test.applymap(int).astype(int)
data_train.iloc[:, 1:] = data_train.iloc[:, 1:].applymap(int).astype(int)
```

```
# Data validation stuff, more cleaning

all(data_train.iloc[:,1:].dtypes == 'int64') # True (All entries are integers)
all(data_test.dtypes == 'int64') # True (All entries are integers)

all(list(map(lambda a: ' ' not in a, data_train.iloc[:,1:].columns))) # Each column is one word
all(list(map(lambda a: ' ' not in a, data_test.columns))) # Each column is one word
```

[Figure 2: An Example of Data Cleaning]

To begin the data cleaning, each column was checked for whether it was of the right type (since the respective columns' content were all word counts, they should all have been integers). Once that was done, the columns were checked for any typos or mistakes. After all of this, all the missing NAN values were filled with 0s. The column name for the labels (chapters) was changed to 'Chapters' for convenience. We noticed that the book Ecclesiasticus was misspelled, and instead was written as 'Eccleasiasticus', however, we chose to keep it this way because it might have been intended. In addition, we found that there were 5 words that were only 1 letter characters, but decided to not remove them because they were likely to be real words that were unbeknownst to us.

```
# Data to work with

books = [i[0] for i in list(map(lambda a: a.split('_'), data_train['Chapters'].unique())) if i[1] == 'Ch1']

# dataframes segregated by chapters
book_list = [data_train[[j in i for i in data_train['Chapters']]] for j in books]

# top 20 words for each book in a list containing 8 series
top_20_books = [j.sort_values(ascending = False) for j in [i.iloc[:,1:].sum() for i in book_list]]

# total words in each book
total_words_book = dict(zip(books, [i.sum() for i in top_20_books]))

# total words in each chapter of each book (dictionary)
total_words_chapter = dict(zip(books,[pd.Series(data=i.iloc[:,1:].sum(axis=1).tolist(), index=i.iloc[:,0]) for i in

# Top 20 words for all books
all_20 = data_train.iloc[:,1:].sum().sort_values(ascending = False)

# total words in the dataframe
total_words = all_20.sum()

# proportion of words in the top 20 over total words in each book
prop_20_book = dict(zip(books,[top_20_books[i][:20].sum() / [j.sum() for j in top_20_books][i] for i in range(8)]))
```

[Figure 3: Data organization]

For pre-processing, the data was aggregated several ways. We organized and combined the data in various ways. We calculated important statistics such as the total number of words in the dataset, which was **60609** words. We extracted the 8 books that were contained in the dataset by parsing through the many chapters. These books were named: **Buddhism, Yogasutra, Upanishad, Tao Te ching, Book Of Proverbs, Book Of Wisdom, Book Of Ecclesiastes,** and **Book of Ecclesiasticus**. With this information, we were able to calculate the total number of words in each book, the top 20 most frequent words in each book (More in the later sections) and other statistics

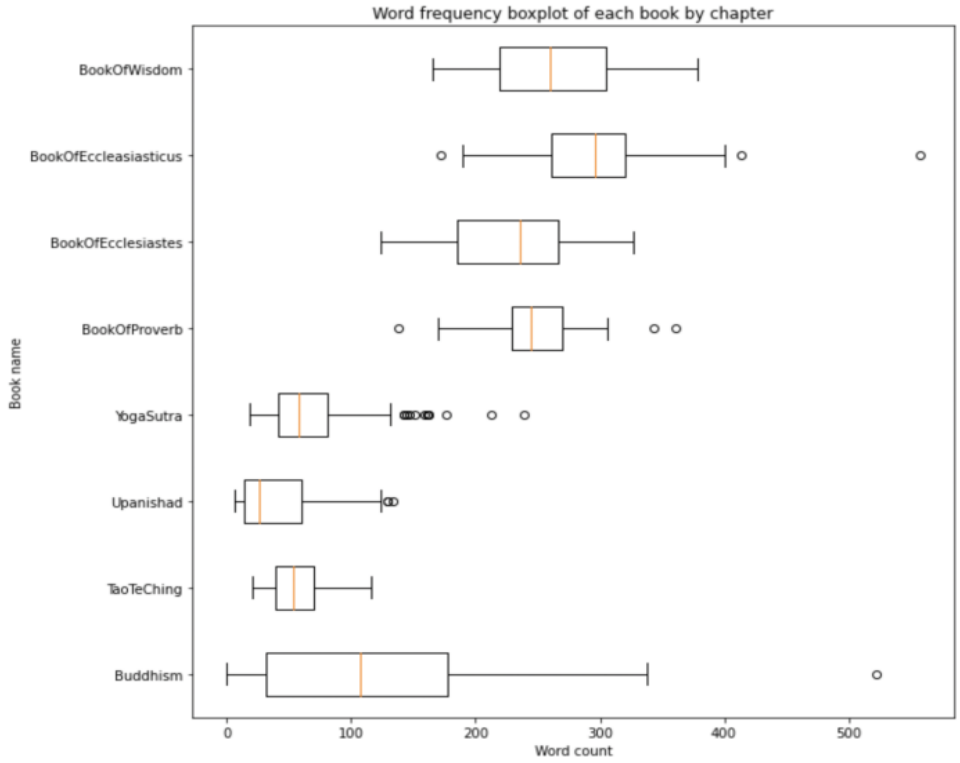
Visualizations:

Our primary visualization was a word cloud, in which the size of the word correlated with the frequency of the word throughout every book. That is, the most common words were larger in size than those less common in the text. Another one of our visualizations was an array of word clouds of each book's top twenty words. Additionally, we used a bar chart of words per chapter, color-coded by book, as well as a boxplot which visualized the same idea. Notably for these visualizations, we removed an outlier from the data set in order to make the visualizations more understandable. The removed value in question was Chapter 37 of the Buddhism text, which was 1194 words long. We have also shown the word frequency of each book by chapter in box-plot form, as well as a bar graph of the top 20 words of each book as proportions.

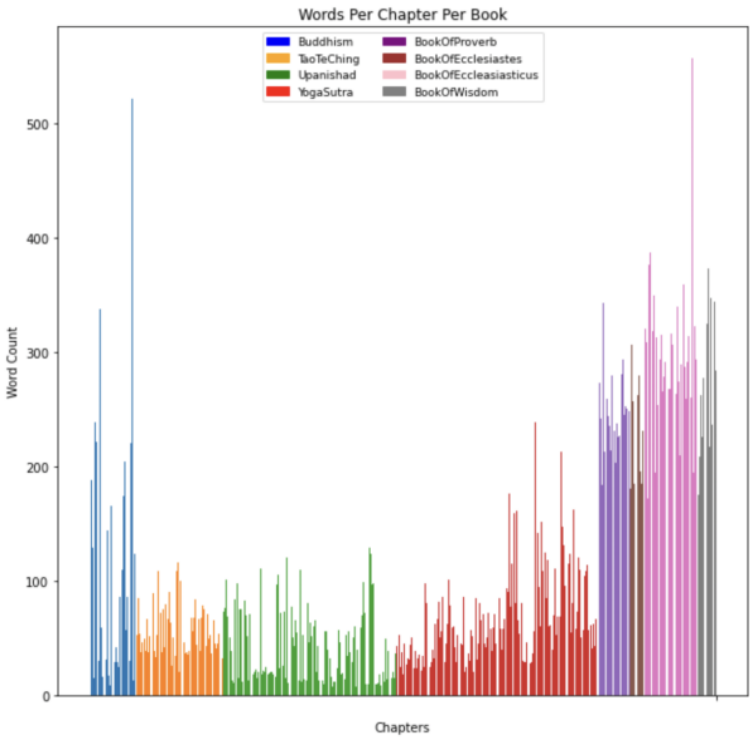
```
# This code basically takes all scores calculated from the sentiment analysis text file,
# in order to help with sentiment score calculations

sentiments = open('sentiment_analysis.txt', 'r')
scores = [float(line.split(' ')[3].strip()) for line in sentiments.readlines() if 'score' in line and 'Document' in
sentiments.seek(0)
magnitude = [float(line.split(' ')[3].strip()) for line in sentiments.readlines() if 'magnitude' in line and 'Docume
sentiments.seek(0)
keys = [line.strip() for line in sentiments.readlines() if ' ' not in line and line.rstrip()]
scores_dict = dict(zip(keys,scores))
magnitude_dict = dict(zip(keys,magnitude))

# function to calculate sentiment scores
book_scores = list(map((lambda j: sum([scores_dict[k] * j[i] for i,k in enumerate(j.index.tolist())]) / sum(j.tolist
magnitude_scores = list(map((lambda j: sum([magnitude_dict[k] * j[i] for i,k in enumerate(j.index.tolist())]) / sum(
top_20_scores = list(map((lambda j: sum([scores_dict[k] * j[:20][i] for i,k in enumerate(j[:20].index.tolist())]) /
top_20_magnitude = list(map((lambda j: sum([magnitude_dict[k] * j[:20][i] for i,k in enumerate(j[:20].index.tolist()

# formatting the scores for graphing/visualization
scores_20 = dict(zip(books,list(top_20_scores)))
magnitude_20 = dict(zip(books,list(top_20_magnitude)))
book_scores_dict = dict(zip(books,list(book_scores)))
book_magnitude_dict = dict(zip(books,list(magnitude_scores)))
```

[Figure 4: Calculating the Sentiment scores]

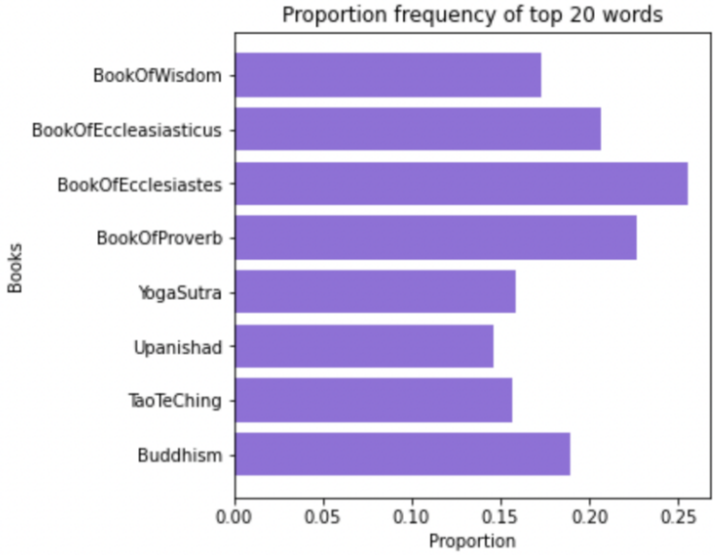
A visualization we also included was a bar graph of average sentiment of words in a book, plotted against the average sentiment of the top 20 words of the respective book. The sentiment analysis API was obtained from Google Clouds. In the documentation, sentiment is defined as a number between -1 and 1, where -1 is strongly negatively sentimental, while 1 is strongly positively sentimental. In this way, “death” takes a value of -0.9, while “love” takes a value of 0.9. Finding the average sentiment of each book allowed us to make conclusions about the overall tone of each text.



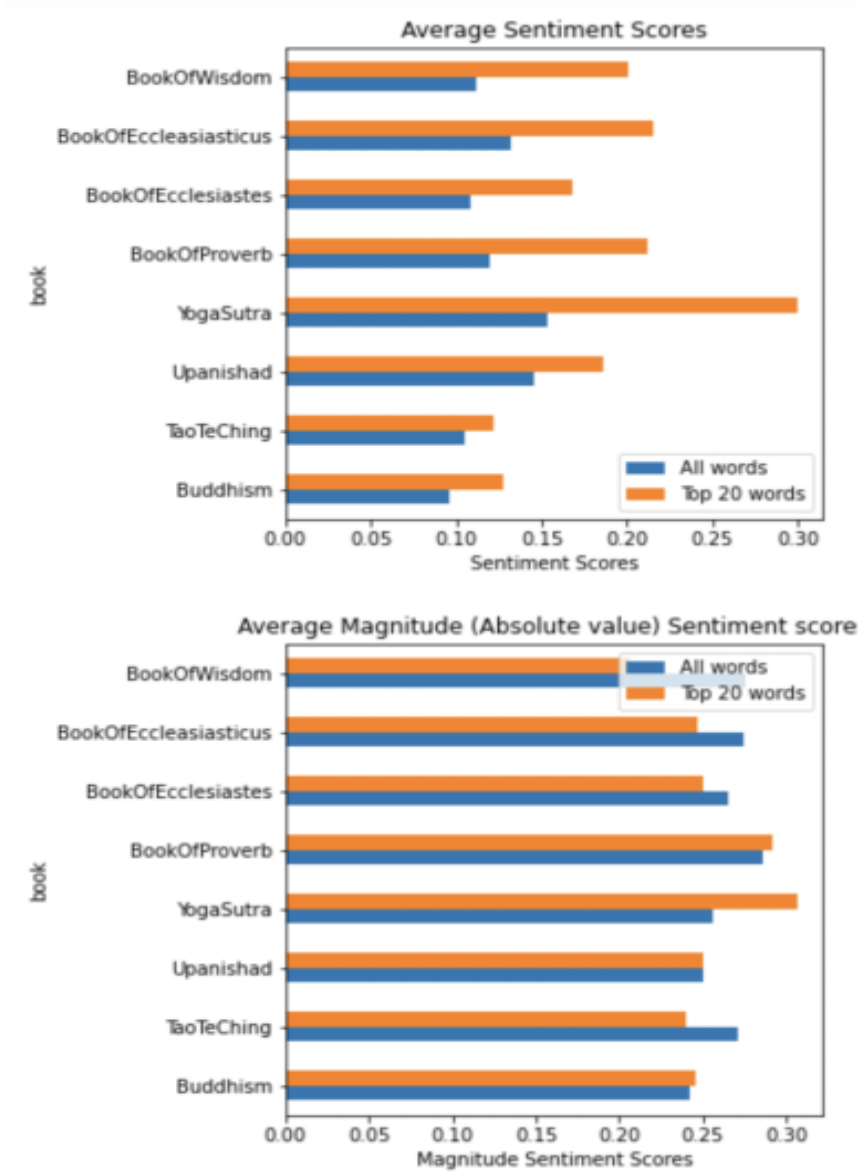
[Figure 5: A boxplot in which each data point is the total count of words per chapter. The chapters are then plotted into separate boxplots segregated by book.]



[Figure 6: A general histogram of the entire dataset. The graph is detailing 8 books corresponding to the 8 different colors. Each bin in this histogram represents a chapter in a book, with its height being the total number of words found in said chapter. It is interesting to note that the Old Testament books (Proverb, Ecclesiastes, Ecclesiasticus, Wisdom) seem to have less chapters compared to Eastern books (barring Buddhism and more specifically ch37) while having significantly more words in it.]

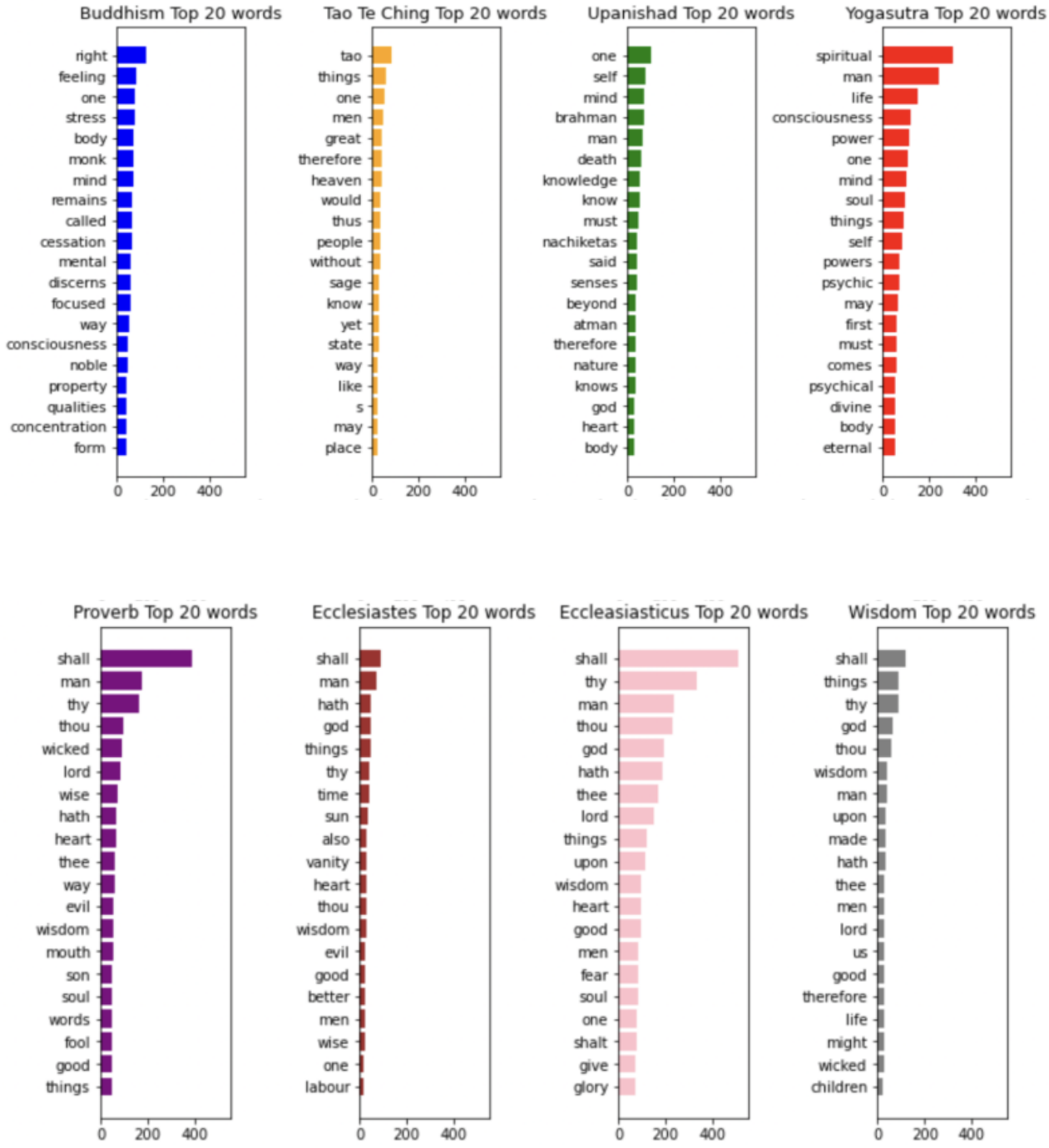


[Figure 7: A bar chart detailing the proportion frequency of the top 20 words in each book. It is also interesting to note that the top 20 words for all 8 books make up around 15%-25% of all the words in each respective book.]



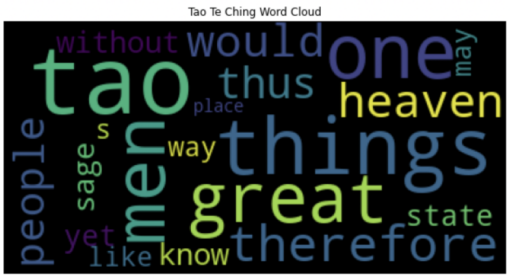
[Figure 8: The first bar chart details average sentiment scores for each book (blue) plotted with the average sentiment scores exclusively for the top 20 words in each book (orange). The second bar chart displays the average magnitude of the sentiment scores for each book (blue) plotted with the average magnitude of the sentiment scores exclusively for the top 20 words in each book (orange). With the inclusion of absolute value sentiment scores, we could see that there is an increase from the regular sentimentality scores, suggesting that the scores are polarized, meaning that instead of having mostly neutral scores close to 0, the books have relatively high positive and negative sentiment scores that are cancelling each other out.]

[Figure 9: Eight word clouds consisting of twenty words, representing the most frequent words in each book.]



[Figure 10: The top twenty most common words for each of the eight books in the data set. They are separated into separate barcharts, and are in descending order. The Eastern texts take the top row, and the Western texts take the bottom]

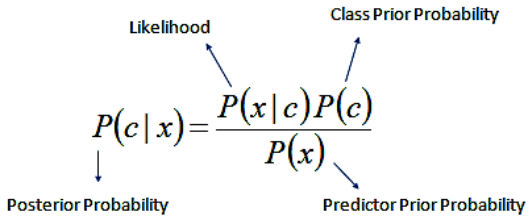
Buddhism Vs. Taoism:

Buddhism:	Tao Te Ching:
Buddhism as a religion focuses much on the self. It can be defined as by four basic truths: (1) existence is suffering; (2) suffering has a cause, namely craving and attachment; (3) there is a cessation of suffering, which is nirvana; and (4) there is a path to the end of suffering, the eightfold path of right views, right resolve, right speech, right action, right livelihood, right effort, right mindfulness, and right concentration. Buddhism characteristically describes reality in terms of process and relation rather than entity or substance.	Taoism, similar to Buddhism, focuses much on the self, and the need for the soul to become free from the body. Contrary to Buddhism, however, the main goal of Taoists is to achieve balance in life and reach immortality through Tao, translated as The Way. According to the Taoist beliefs, Tao is the first cause of the universe, a force that flows through all life that exists. The primal forces of the feminine and the masculine, or the yin and the yang, have an important role in the Taoist creation myth.
	
This can be easily reflected in the top twenty words found in the text, as seen in Figures 8 and 9. Words such as **mind**, **mental**, **focused**, **consciousness**, and **concentration** show that the religion focuses heavily on the individual as opposed to an outside force. On the other hand, **stress**, **monk**, **right**, and **cessation** (the end) reflect the belief that there exists suffering, with the goal of resolution.	The **Tao** is the central idea behind Taoism, and is expectedly the most common word found within Tao Te Ching (Figure 8, Figure 9). Unlike Buddhism, Taoism believes in the afterlife, and that when the soul becomes one with Tao, it reaches **heaven**, a word which we see commonly used within the text. Taoism focuses on the objective more on the objective, reflected in the words **things**, **state**, **way**, and **place** being common-place within the text.

These facts can also be reflected in the sentiment score of the text, which shows a very neutral tone (Figure 7), one which equally reflects on the positives and negatives of life and the afterlife. As the religion and its text do not shy away from the hardcomings of life, topics of both positives and negatives are common within the text, and the sentiment score reflects such.	Similar to Buddhism, the Tao Te Ching text has a sentimental score which is fairly low (Figure 7), indicating the presence of conflicting beliefs in the text which leave it neither strongly positive nor negative. This is to be expected as the idea of Yin and Yang reflects both the positive and the negative, and can be seen as words such as great and without both have high frequency throughout.

Proverbs, Ecclesiastes, and Wisdom in the Old Testament:

Given the problem of comparing the three parts of the old testament, a solution which is guided by machine learning that we came to was classification through the use of the Naive Bayesian model. Given a data set of chapters who's labels are their book titles, and their set of features defined as their word count, one can use the Naive Bayesian classification model to predict which book a chapter would fall into given their set of words.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

[Figure 11: The formula for a Naive Bayesian algorithm]

The problem uses the formula present in Figure 11, and applies it three times for each label. Then, the maximum of these three probabilities is selected, and the book which it represents is the predicted label which is given to the chapter which was passed through the algorithm. We modified this algorithm in order to account for the presence of a word with a frequency higher than one, by multiplying the probability of that feature given the label by the count of that label. Another way in which this algorithm was modified was by excluding any words from the calculation which had a frequency less than one. This is due to the fact that, even

after correcting for these small values, these proportions would be so small that they cause the probabilities of the classification to become so small that they are equal to zero. The code which executes this function can be seen in Figure 12.

```
# PREDICTION #
def book_classifier_prediction(input_chapter):
    #Probability of Proverb Given Features
    proverb_prob_product = prob_proverb
    for feature in range(len(top_books[proverb_index])):
        if input_chapter[feature] > 1:
            #print(input_chapter[feature])
            prob_feature_given_proverb = (proverb_feature_list[feature] * (input_chapter[feature] + 1))
            #print(prob_feature_given_proverb)
            proverb_prob_product *= prob_feature_given_proverb
            #print(proverb_prob_product)


    #Probability of Ecclesiastes Given Features
    eccle_prob_product = prob_eccle
    for feature in range(len(top_books[eccle_index])):
        if input_chapter[feature] > 1:
            prob_feature_given_eccle = (eccle_feature_list[feature] * (input_chapter[feature] + 1))
            eccle_prob_product *= prob_feature_given_eccle

    #Probability of Wisdom Given Features
    wisdom_prob_product = prob_wisdom
    for feature in range(len(top_books[wisdom_index])):
        if input_chapter[feature] > 1:
            prob_feature_given_wisdom = (wisdom_feature_list[feature] * (input_chapter[feature] + 1))
            wisdom_prob_product *= prob_feature_given_wisdom

    # CLASSIFICATION #
    if max(proverb_prob_product, eccle_prob_product, wisdom_prob_product) == proverb_prob_product:
        return "BookOfProverb"
    elif max(proverb_prob_product, eccle_prob_product, wisdom_prob_product) == eccle_prob_product:
        return "BookOfEcclesiastes"
    else:
        return "BookOfWisdom"
```

[Figure 12: A section of the code used to execute the Naive Bayesian Classification Algorithm. It calculates the probability of the chapter taking each label given its words, and then labels the chapter with the book which had the highest probability of occuring]

Proposal:

Given more time, an analysis of the data set which would be interesting to complete would be using an API in order to characterize each word by its part of speech, and then comparing these parts of speech between each text. This method was attempted by our group, but ended up unfruitful due to the labeling method not being extensive enough. However, given the information that was collected regardless, conclusions can be drawn about the types of speech most often present in each book. This would lead to more detailed inferences, especially when looking at the evolution of the books that are part of the Old Testament.

Conclusion:

Many observations can be drawn from the report presented above. Words that stand out in the word clouds of the Western texts are terms like “lord”, “man”, and “god”, while the Eastern texts emphasize “feeling”, “knowledge”, and “spiritual”. The Western texts are more authoritative while the Eastern portray individualism. The Western texts tout obedience, while the Eastern emphasize focus. These stark contrasts seem very obvious to the eye when looking at the plotted graphs; Sentiment analysis on the other hand tells a different story. It shows that these texts all portray slightly positive tones, all very close in value to one another. Buddhism and Taoism share very similar sentiments, while the others show slightly more passionate or positive notes. Through the eye of a computer, the emotional impacts of the words that pop out at us, seem casual in comparison.

Comparably, observing the calculations of the magnitude, we can see polarizing results, as these values appear much higher than their sentiment counterparts, pointing towards a large range of emotional language which sways both positively and negatively. These texts address the hardships and the darker sides of life, while also attempting to highlight the beauty of their respective religions. The juxtaposition of these views stand out prominently in the analysis of the texts, and reflect the many complexities that are attributed to life and death.

These texts illuminate the double-sided nature of religions, as well as the contrast between religious standards in the East and West.

Bibliography:

"Buddhism: Basic Beliefs and Practices." *Infoplease*, Infoplease, www.infoplease.com/encyclopedia/religion/eastern/buddhism/buddhism/basic-beliefs-and-practices.

Pircher, Author: Richard, and Richard Pircher. "Taoism vs Buddhism: Primary Differences & Similarities: Handmadewriting." *Handmadewriting.com*, 23 Mar. 2021, handmadewriting.com/blog/samples/taoism-vs-buddhism-primary-differences-and-similarities/#:~:text=Major%20differences%20between%20Taoism%20and%20Buddhism,-How%20do%20Taoism&text=Buddhists%20share%20a%20belief%20in,universe%20that%20guides%20everything%20impersonally.

R, Sridhar C. *Naive Bayes Algorithm*, Blogger, 3 Nov. 2017, computer-lords.blogspot.com/2017/11/naive-beyes-algorithm.html.