Sentiment Analysis of The King James Bible*

*Sentiment Analysis of The Bible: A Comparative Study of Techniques

1st Hoots
Montclair State Linguistics Department
APLN 552
Plainfield, United States
hootsd1@montclair.edu

***Abstract*—Using lexicon-based sentiment analysis methods we generated ground truth labels for every verse in the king James bible. Then using multiple machine learning models, we analyze sentiment in the King James Version. To find the best models, we trained and tested many models. SVM was the most effective, achieving 80% accuracy. In some cases, however, it had limitations. Our findings indicate that machine learning can be applied to religious texts and provide a foundation for future research to improve sentiment analysis.**

I. Introduction

This final project compares different sentiment analysis techniques and their results when applied to the Bible text. In this study, we aim to evaluate the performance of each method and identify the most effective method for sentiment analysis of religious texts.

Sentiment analysis of the King James Version can also help us better understand religious communities' beliefs and perspectives. Analyzing the Bible's sentiments can improve faith-based education and better understanding religious culture. Sentiment analysis helps interpret the Bible's language as well.

Natural language processing algorithms and supervised machine learning algorithms will be used to identify sentiments in the Bible text. We will use unsupervised machine-learning techniques such as clustering and topic modeling to detect religious texts' sentiments.

The results of each technique will be compared after we have identified the text's sentiment. To measure sentiment analysis methods' effectiveness, we will compare their results with human annotations.

Based on our final results, we can provide insights into the effectiveness of various sentiment analysis techniques when applied to religious texts. Our findings will improve linguistic interpretation and sentiment analysis accuracy for religious texts.

II. Data

A. Data Acquisition

Our datasets are from Kaggle:

We have several options for choosing which Bible dataset we will use.

We will review the advantages and disadvantages of the two options considered for the project.

The first dataset we will look at is the **"Bible Corpus" by OSWIN RAHADIYAN HARTONO** dataset

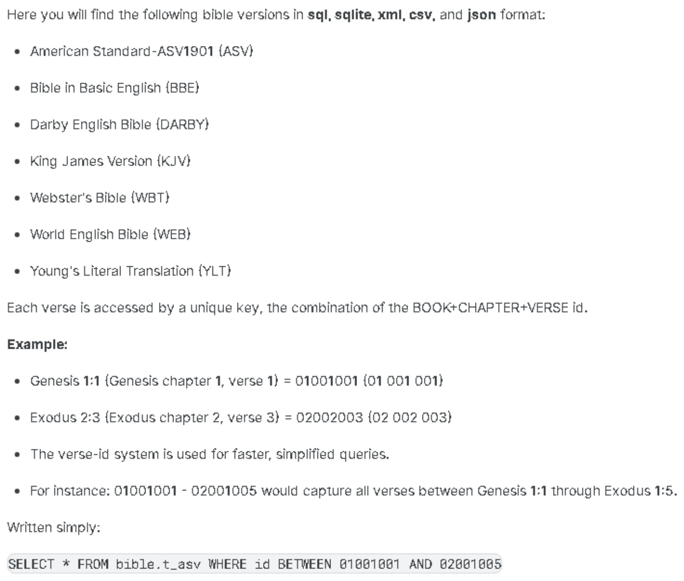
https://www.kaggle.com/datasets/oswinrh/bible

This dataset offers the following **advantages**:

- There are several popular Bible translations included in the dataset, such as the King James Version (KJV), American Standard Version (ASV), and World English Bible (WEB). It is possible to compare different versions and the nuances of different translations.
- In addition to using a unique verse identification system, the dataset has a unique verse retrieval system that simplifies and speeds up the querying process. This makes it easier for users to access and work with specific verses within the dataset.
- A cross-reference table to facilitate related scriptures can be found within the dataset, along with coordinating tables for book names, genres, and abbreviations. These additional resources can help analyze the Bible comprehensively.
- Data pre-processing and formatting: The dataset can be used in several formats, including SQL, SQLite, XML, CSV, and JSON, providing flexibility and compatibility with different programming languages and software tools.
- The cross-reference table, created from the Open Bible project, provides a valuable resource for identifying related scriptures and enhancing Bible study.
- A simple web-based application to search for and display verses may be created using the PHP files included in the dataset.

This dataset has the following **disadvantages**:

- A dataset containing only English Bible translations is unsuitable for research involving other translations or languages.
- The dataset does not contain sentiment labels: The dataset does not contain sentiment labels, requiring researchers to analyze and label the data before training deep learning or machine learning algorithms.
- Further processing may be required: The dataset text may need tokenization, stopwords removed, and lemmatized before sentiment analysis.


Here you will find the following bible versions in **sql, sqlite, xml, csv,** and **json** format:

- American Standard-ASV1901 (ASV)
- Bible in Basic English (BBE)
- Darby English Bible (DARBY)
- King James Version (KJV)
- Webster's Bible (WBT)
- World English Bible (WEB)
- Young's Literal Translation (YLT)

Each verse is accessed by a unique key, the combination of the BOOK+CHAPTER+VERSE id.

Example:

- Genesis 1:1 (Genesis chapter 1, verse 1) = 01001001 (01 001 001)
- Exodus 2:3 (Exodus chapter 2, verse 3) = 02002003 (02 002 003)
- The verse-id system is used for faster, simplified queries.
- For instance: 01001001 - 02001005 would capture all verses between Genesis 1:1 through Exodus 1:5.

Written simply:

```
SELECT * FROM bible.t_asv WHERE id BETWEEN 01001001 AND 02001005
```

Fig. 1. First dataset table of contents

The second dataset we will look at is the **"Explore King James Bible Books" by GABRIEL PREDA** dataset

https://www.kaggle.com/code/gpreda/explore-king-james-bible-books

This dataset offers the following **advantages**:

- The dataset may have more depth and analytical information as it is restricted to the King James Version of the Bible.
- Based on the dataset, several visuals are generated.
- This program displays the ten most frequently used words, which can be helpful when searching for stop-words and removing them.
- A pre-prepared sentiment analysis may provide a valuable basis for training supervised models.
- Additionally, it enables us to identify topics in each Bible book quickly. Instead of analyzing each book sentence by sentence, using this method for a broader sentiment analysis might be helpful. We may better understand the Bible's overall sentiment through this.

This dataset has the following **disadvantages**:

- The fact that this dataset contains only one Bible version limits our ability to test our trained models on other religious texts.
- Sentiment analysis has already been performed on this dataset, so examining how sentiment values were calculated is necessary. Due to the extensive dataset processing, further expansion may be complex.
- Since we are primarily concerned with defining a sentiment analysis of the Bible and evaluating each verse to determine its sentiment, we are not interested in analyzing many of the analytics already contained in the dataset.

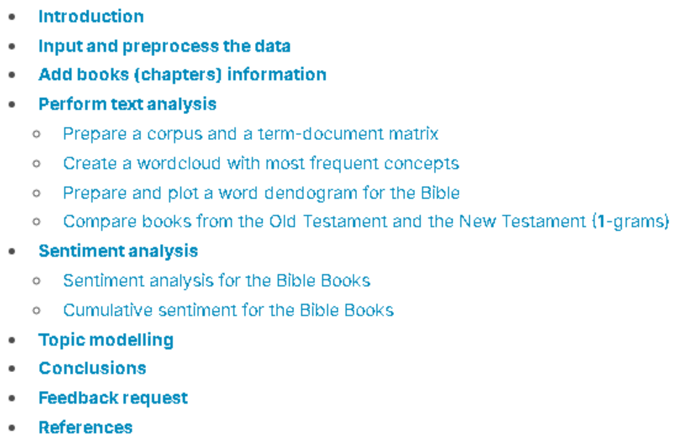

- **Introduction**
- **Input and preprocess the data**
- **Add books (chapters) information**
- **Perform text analysis**
  - Prepare a corpus and a term-document matrix
  - Create a wordcloud with most frequent concepts
  - Prepare and plot a word dendogram for the Bible
  - Compare books from the Old Testament and the New Testament (1-grams)
- **Sentiment analysis**
  - Sentiment analysis for the Bible Books
  - Cumulative sentiment for the Bible Books
- **Topic modelling**
- **Conclusions**
- **Feedback request**
- **References**

Fig. 2. Second dataset table of contents

In conclusion, we will work with the "Bible Corpus" by OSWIN RAHADIYAN HARTONO for this study because the dataset is already separated by verses which means we do not need to process the Bible and split it into parts. It also allows us to start sentiment labeling from scratch, which means we will have a greater understanding of the results and be more confident in testing the results against other versions of the Bible and even other religious texts.

We will use "Explore King James Bible Books" by GABRIEL PREDA dataset to check against our results and see if we come up with a similar result as this one. Which ever labeling method delivers a trend close to what this dataset shows, we will deem to be the most accurate.

B. Data Preparation

1) Data cleaning: We will use the following techniques to clean the data. We will assume that the Bible has no spelling errors, and that all information is valuable for context/sentiment labeling. We will remove any extra information attached to the dataset since we only need a list of the verses.

2) Text pre-processing: We will:

- Tokenize the text: Separating it into individual words or phrases.
- Remove the stop words: remove common words with little analytical value
- Lemmatize the text: reduce words to their simplest form.

3) Feature extraction and labeling: We will train and compare Four different lexicon-based methods for Sentiment Analysis. We will use:

- **VADER** - VADER Sentiment Analysis is a lexicon and rule-based sentiment analysis tool designed to analyze social media texts [1] [2].

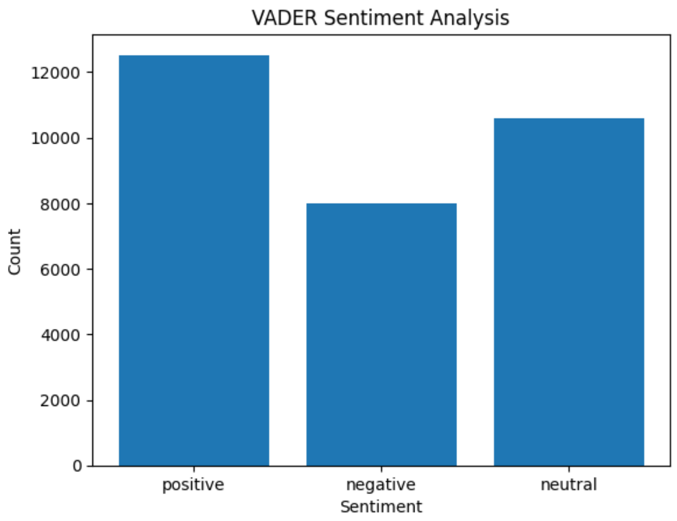


Fig. 3. Vader Sentiment Analysis Results

- **TextBlob** - TextBlob is a Python library for processing textual data [3]. 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, translation, and more [3].

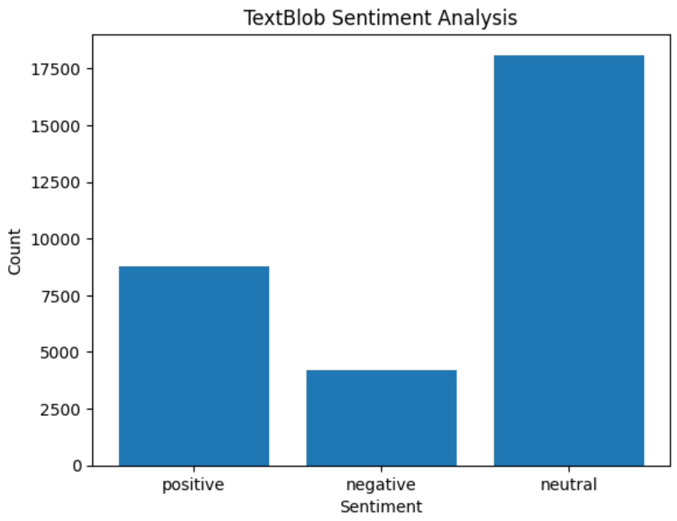


Fig. 4. TextBlob Sentiment Analysis Results

- **SentiWordNet** - SentiWordNet is a lexical resource for opinion mining [5]. It assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity [5]. SentiWordNet is the first lexical resource which provides such specific level of detail (the word sense represented by a synset) and such broad coverage (all the 115,000+ WordNet synsets) [4].

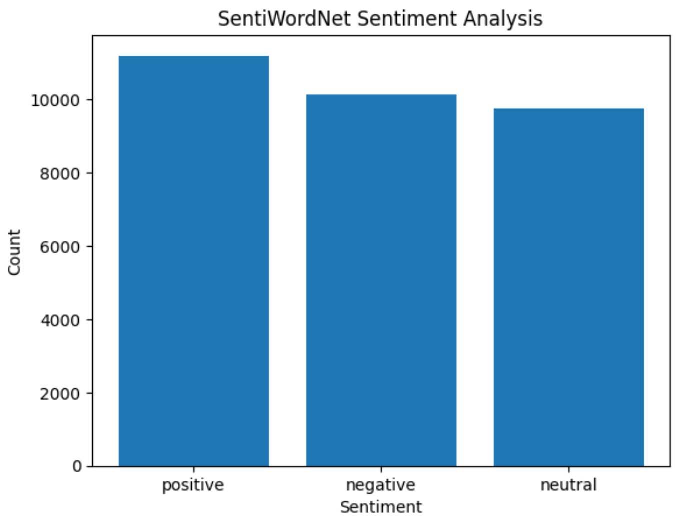


Fig. 5. SentiWordNet Sentiment Analysis Results

- **Sentimentr** - an R package designed to quickly calculate text polarity sentiment in English at the sentence level and optionally aggregate by rows or grouping variable(s) [7]. It is a dictionary-based sentiment analysis tool that considers valence shifters such as negators, amplifiers (intensifiers), de-amplifiers (downtoners), and adversative conjunctions while maintaining speed [8]

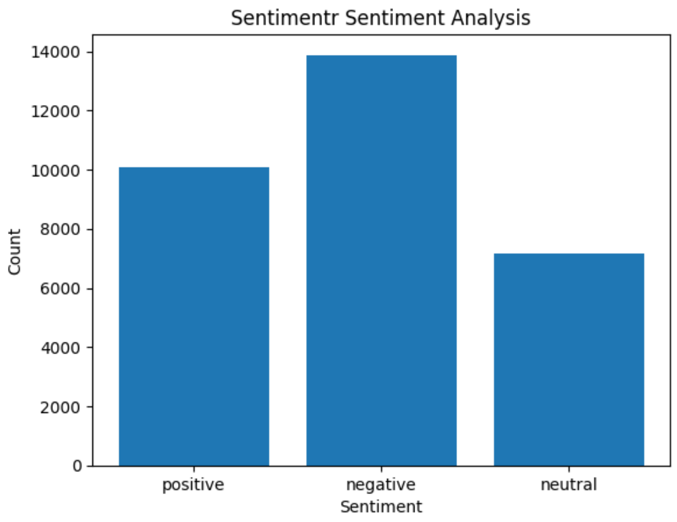


Fig. 6. Sentimentr Sentiment Analysis Results

Studying the results from the "Explore King James Bible Books" study by GABRIEL PREDA, we find that while they separated their results into three sentiments, they looked at the overall sentiment of the Bible books. It is different for us as we intend to analyze the overall sentiments of the Bible based on individual verses instead of allowing books that are longer or with varying verse lengths to have a more substantial impact on the results (for example, Psalms, the longest book, also contains the most extended verse). We may lose many sentimental impacts by counting only one sentiment for a book. Therefore, we will apply the same formula used for sentiment analysis, using verse-by-verse instead of book-by-book.

C. Quality assurance

1) Comparing Results: Below we see that labeling gets varying results.

- Vader - Vader's analysis was mostly Positive and Neutral. Positive 40.3%, Neutral 34.1%, Negative 25.7%. These results make sense when we look at how Vader was trained on social media texts and reviews. Bible verses are probably a similar size making Vader ideal for labeling.
- TextBlob - TextBlob seems to favor Neutral Sentiment. Positive 28.3%, Neutral 58.2%, Negative 13.6%. These results from TextBlob aren't unexpected considering that it prioritizes frequency using a Naive Bayes Classifier trained on movie reviews using a Bag of Words model. While some verses in the bible may have similar lengths to movie reviews, the language used is vastly different.
- SentiWordNet - SentiWordNet analysis was the most balanced. Positive 36.0%, Neutral 31.4%, Negative 32.6%. Being that SentiWordNet is developed specifically for calculating positive, negative, and objective scores it performed the most balanced and is a top pick for our labeling results. Later on, we will look at further research suggesting that we may need a combination of results from multiple analyses to get overall accurate labeling.
- Sentimentr - The Sentimentr result favors a negative sentiment with Positive 32.4%, Neutral 23.0%, and Negative 44.6%. However, this can be misleading. Our analysis has moved many results, which might have been positive in the paper "Bible Corpus" by OSWIN RAHADIYAN HARTONO, to neutral. Sentimentr is ideal for this task due to its ability to calculate sentiment polarity at a sentence level since most bible verses do not contain more than two sentences.

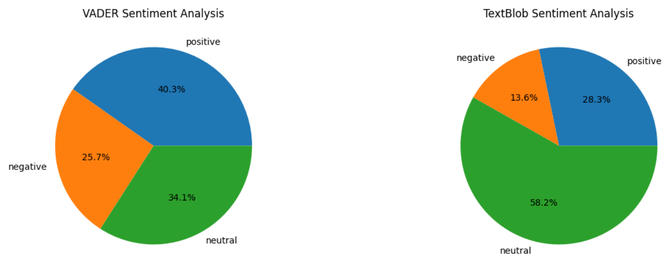


Fig. 7. Sentiment Counts for Vader and TextBlob

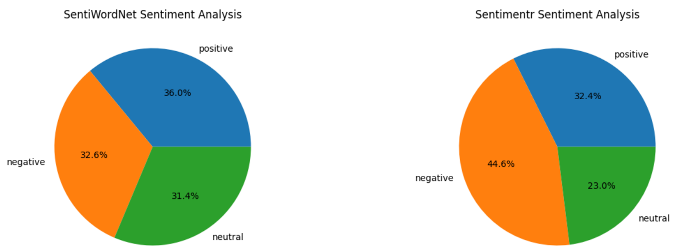


Fig. 8. Sentiment Counts for SentiWordNet and Sentimentr

In the paper "Bible Corpus" by OSWIN RAHADIYAN HARTONO, the sentiments are broken into three parts: min, max, and mean, and the analysis method is Sentimentr. The final score of a book was determined by whether the mean was negative or positive. This meant that each book had either a negative or positive sentiment. As part of our results, we would like to include a neutral sentiment so that verses that may only list off family trees, for example, will not dilute positive and negative outcomes.

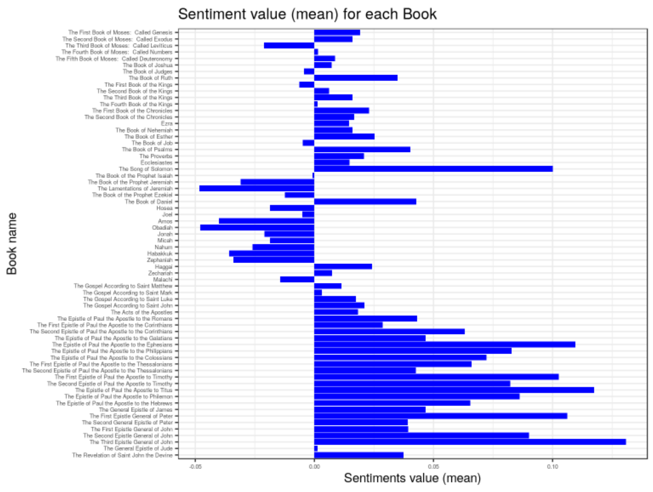


Fig. 9. Sentiment results from "Bible Corpus" by OSWIN RAHADIYAN HARTONO

Our varying analysis results mean we need another input to eliminate some choices. We can look at the paper "TEXT AND SENTIMENT ANALYSIS ON THE WORLD'S MOST READ BOOK" by Weslet Shen. They conclude that "Overall, the sentiment of the Bible has a balance of both positives and negatives. Words frequently used by the authors to characterize God are great, good, holy, and love. And words such as evil, death, and sin are used the most to characterize Satan, the earthly world, and the human condition." [9] The author reads the bible verse-by-verse but only really focuses on the most frequent words and their context in order to come to a conclusion.

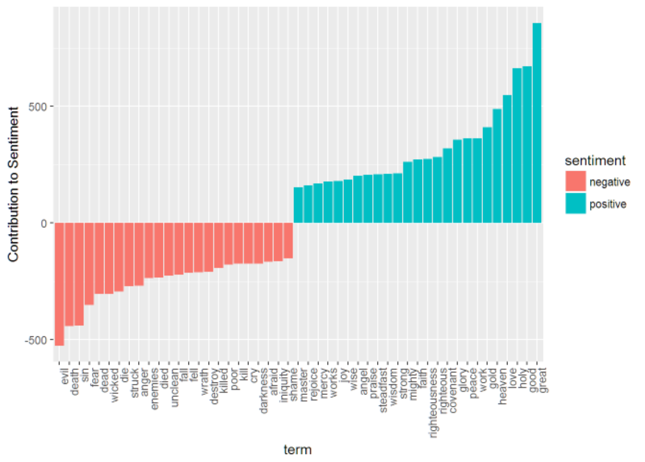
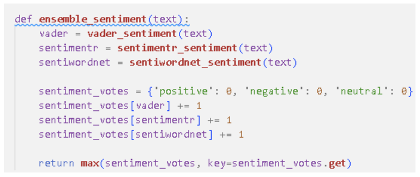


Fig. 10. Sentiment results from ”Bible Corpus” by OSWIN RAHADIYAN HARTONO

D. Generating Ground Truth Labels

1) Ensemble Method: Majority Voting System: Since our analysis has varying results, we must create a plan for our definitive ground labels. We are looking for results that reflect the following findings from the analysis: ”We can see that the Bible starts with a strong positive and a sudden strong negative, followed by a negative trajectory towards the middle, and then ending in positivity in the New Testament” [9]. So our results should reflect that analysis. Mostly positive, some negative, with neutral being the least.



```python
def ensemble_sentiment(text):
    vader = vader_sentiment(text)
    sentimentr = sentimentr_sentiment(text)
    sentiwordnet = sentiwordnet_sentiment(text)

    sentiment_votes = {'positive': 0, 'negative': 0, 'neutral': 0}
    sentiment_votes[vader] += 1
    sentiment_votes[sentimentr] += 1
    sentiment_votes[sentiwordnet] += 1

    return max(sentiment_votes, key=sentiment_votes.get)
```

Fig. 11. Majority Voting System Code Snippet

Algorithm 1 Returning the most frequent value in a set (Mode Calculation)

```
Require: $S = \{s_1, s_2, s_3\}$ {Sentiment Result from functions}
Require: $D = 0$ {default value}
 1: Initialize $C = \{0, 0, 0\}$ {count array}
 2: for $i = 1$ to $n$ do
 3:     $C[s_i] = C[s_i] + 1$
 4: end for
 5: $m = \arg\max(C)$ {find the index of the maximum count}
 6: if number of maxima in $C > 1$ then
 7:     return $D$
 8: else
 9:     return $s_m$
10: end if
```

In order to prevent a stalemate in our voting system, we will only use three of our four results. Since our Sentimentr and Textblob have the most varied results from the other two, we will run two tests. Whichever comes closest to our ”mostly positive, some negative, with neutral being the least” expectation will be the labels we will consider our ground labels. if somehow all three of our inputs are the same, we will assign the neutral sentiment.

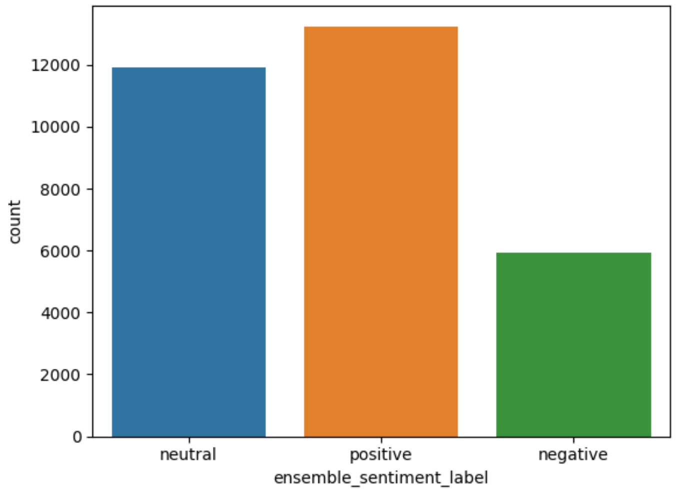


Fig. 12. Vader, TextBlob, SentiWordNet ensemble results

The above results are unfortunately not usable for our purpose as we need the results to go in descending order of positive, negative, then neutral. We will now see what happens when we replace the TextBlob, with Sentimentr.

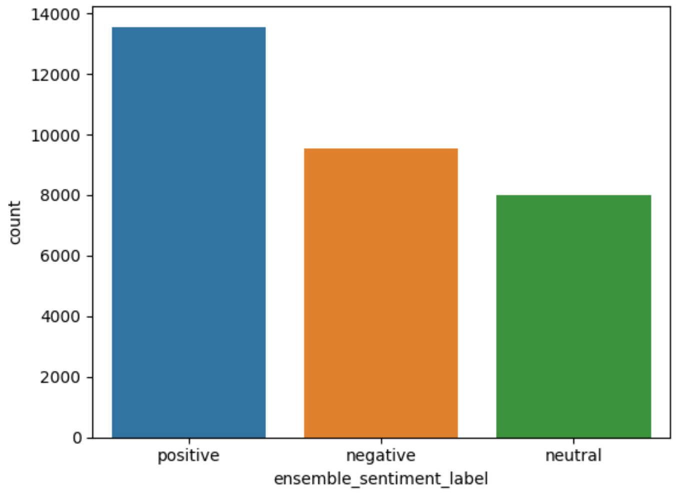
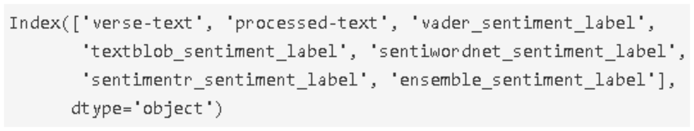


Fig. 13. Vader, Sentimentr, SentiWordNet ensemble results

The above gives us a labelling distribution that goes right along with our predictions and so moviing forward we will utilize the labels generated from this combination of reults.

E. Finalized dataset

With our now-generated ground truth labels, we finally have a dataset we can work with and begin testing to see which training method and models give us the most consistent results.



```
Index(['verse-text', 'processed-text', 'vader_sentiment_label',
       'textblob_sentiment_label', 'sentiwordnet_sentiment_label',
       'sentimentr_sentiment_label', 'ensemble_sentiment_label'],
      dtype='object')
```

Fig. 14. Vader, TextBlob, SentiWordNet ensemble results

III. Training and Testing

A. Training: Machine Learning Models

We will train and test six other models to determine the optimal model for sentimental analysis on the King James Version of the Bible.

First, we will train and test the following machine-learning methods:

- Naive Bayes
- Support Vector Machines (SVM)
- Random Forest

We will use the methods listed above since they are common machine-learning methods. We aim to see which of the following approaches has the highest accuracy when trained on our recently created ground truth labels.

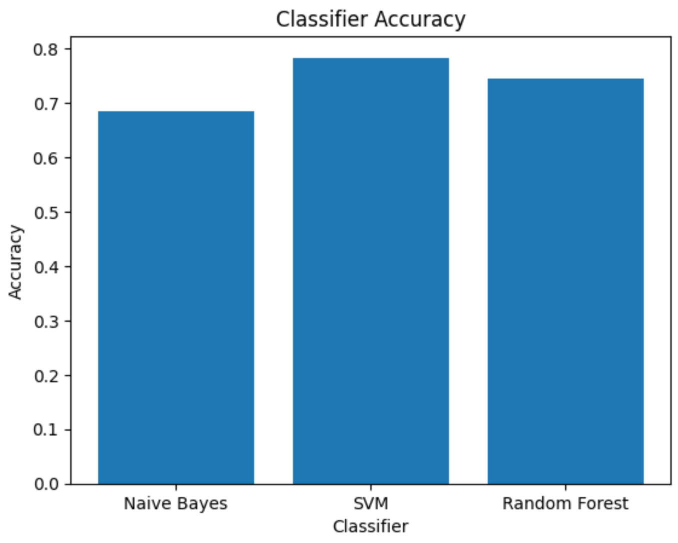


Fig. 15.

From our tests, we can see that SVM gets the highest accuracy rating. Below we list some reasons why this might have been the case.

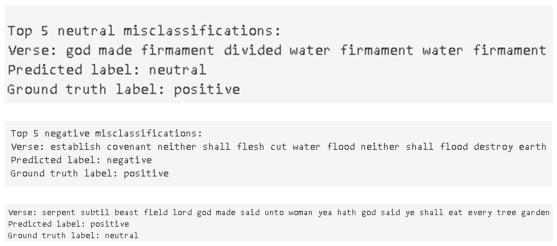
- SVM's perform better on non-linear and higher dimensional data sets. Even though our sentiment analysis only has 3 classes, predicting which one of these classes a piece of religious text falls into might not always be a linear operation since context may lead to some text falling very close into multiple labels.
- SVM's have a strict margin which helps prevent overfitting much better than the other methods.
- An SVM can utilize a combination of features found within the text in order to help it in it's decision process. This is ideal when looking at religious texts due to their complex nature.

B. Testing: SVM

Our SVM model returned almost 80% accuracy. We will now look at some misclassifications and see if we can figure out where the discrepancies are.

A few observations from looking at our misclassifications:

- The model has a higher chance of misclassifying text that is not commands or advice.
- The model misclassifies sentences with many words that carry positive semantic weight regardless of context.
- Many misclassified sentences have to do with creation or birth.
- In the Old Testament, a sentence containing too many names has a higher chance of misclassification.
- Misclassification of questions.



```
Top 5 neutral misclassifications:
Verse: god made firmament divided water firmament water firmament
Predicted label: neutral
Ground truth label: positive
```

```
Top 5 negative misclassifications:
Verse: establish covenant neither shall flesh cut water flood neither shall flood destroy earth
Predicted label: negative
Ground truth label: positive
```

```
Verse: serpent subtil beast field lord god made said unto woman yea hath god said ye shall eat every tree garden
Predicted label: positive
Ground truth label: neutral
```

Fig. 16. First 5 misclassifications in each sentiment

What are some solutions to the above observations?

- **Do not remove stop words or lemmatize:** The Bible is purposely written so that every detail is needed to understand the full context. Even something as small as removing stop words or breaking words down into their base form may flip context.
- **Manual labeling**: While our labeling statistically matches what we should expect from a semantic look at the Bible, some verses in our ground truth labeling may not be correctly labeled, resulting in misclassification in our model. If we see that certain sections of the Bible are heavily misclassified, we may need to label them ourselves manually.
- **Create a labeling system:** If we develop a system that creates labels for anything that is a noun, we can train our model to treat sentences with a high name count differently.

Test 1: Random-Generated Sentences We will now test our SVM against randomly generated sentences from our corpus. The sentences will only contain text from a single sentiment.

Here are some examples that were generated:

- verse: wealth fugitive rescue fiery astonished honor accept deepness plenty
  - sentiment: Positive
- verse: deceit burn dread steal waste woe dimness condemnation',

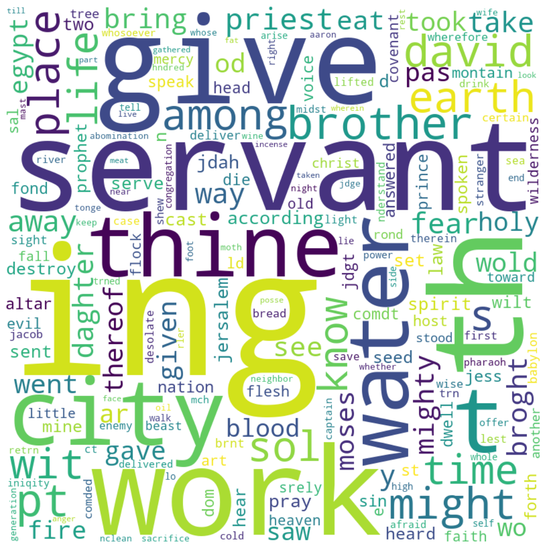


Fig. 17. Word Cloud of top misclassified words

- sentiment: Negative
- banded almondiblathaim malchus cleanseth
  - sentiment: Neutral

As stated earlier, our ground truth labels have moved most nouns into the neutral sentiment, which is what we wanted.

The following are our results:

- verse: wealth fugitive rescue fiery astonished honor accept deepness plenty
  - SVM prediction: Positive
  - ground truth label: Positive
- verse: deceit burn dread steal waste woe dimness condemnation,
  - SVM prediction: Negative,
  - ground truth label: Negative
- verse: banded almondiblathaim malchus cleanseth
  - SVM prediction: Neutral
  - ground truth label: Neutral

Our SVM returns accuracy of 90% Only three of the sixty generated sentences were misclassified:

- verse: forwardness assure prudence zealously valor helping overwhelmed ability
  - SVM prediction: Neutral
  - ground truth label: Positive
- verse: adventure beautify safe smart
  - SVM prediction: Neutral
  - ground truth label: Positive
- verse: industrious curious humbleness awe acceptance
  - SVM prediction: Neutral
  - ground truth label: Positive

Test 2: Random-Generated Sentences Now we will text our SVM on sentences generated using GPT2. Here is an example of a generated verse:

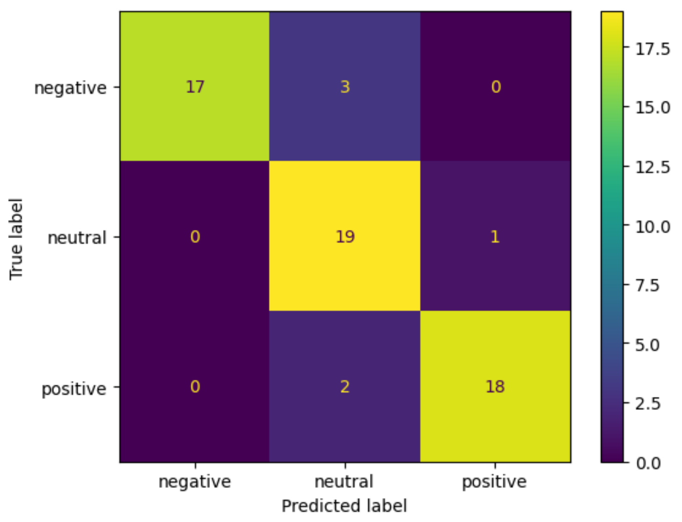


Fig. 18. Confusion Matrix from Test 1

- verse 1: But he hath given him a charge, if he walk according to the word of the LORD and my God. Thus saith the Lord GOD; The LORD hath delivered us from the hand of the wicked. For he hath set us up a throne on this side Jordan, and a throne on that side the river Euphrates.
  - sentiment: positive

While the above verse was correctly predicted. sixteen of the sixty-four generated verses were misclassified giving the model an accuracy of 75% . Here is an example:

- verse: athaliah said unto lord god see entered temple thou yet pray thee till make record matter answered said nay may see thee lord thing thou hast done sa
  - SVM prediction: Positive
  - ground truth label: Negative

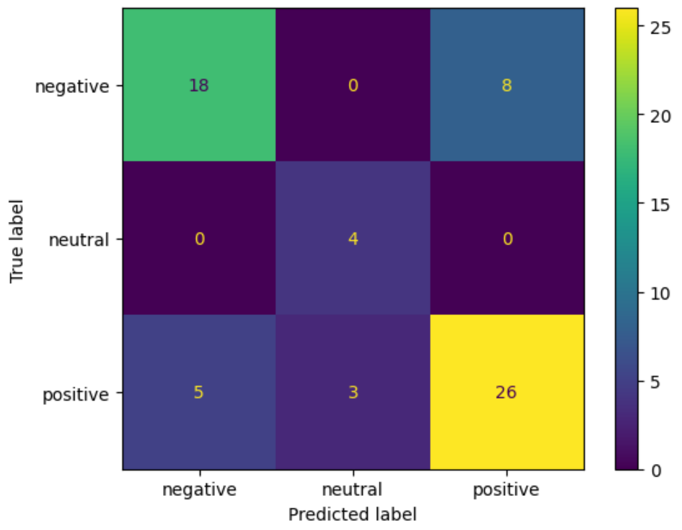


Fig. 19. Confusion Matrix from Test 2

It struggles to accurately predict sentiments in sentences that are not long enough or are fragmented.

IV. Conclusion

Machine learning models were used in this study to analyze the King James Version of the Bible. The Support Vector Machine (SVM) model achieved the highest accuracy rate at nearly 80 percent when predicting the bible verses against the ground truth labels. When predicting against randomly generated sentences (no grammar or coherency considered) our SVM achieved an accuracy rating of 96%. The SVM has difficulty classifying sentences with positive connotations but without explicit positive words. For example, words like ”forwardness”, ”assure”, ”prudence”, ”zealously”, ”valor”, ”helping”, ”overwhelmed”, and ”ability” do not explicitly express positivity but rather imply it, which could be difficult for an SVM to recognize. When predicting sentences generated by a GPT2 model fine-tuned to our bible dataset, our SVM achieved an accuracy of 75%. The above issues are shown further here in situations where explicit language is not used. There is also the possibility that there was more of one sentiment present, and so regardless of context, the SVM went with the highest frequency sentiment.

V. Future Research

Future research could enhance this model’s ability to handle diverse linguistic structures and vocabulary by expanding the training dataset to include different Bible versions. In addition, a more sophisticated noun labeling system could improve sentence processing. Attaching different weighted scores to particular places and names could help tailor labeling further without manual labeling.

There is also the possibility of developing models using other machine learning or deep learning approaches, which could yield more accurate results. The model’s misclassification trends could be studied in greater detail, leading to models that are more specifically tailored to the unique language of the Bible.

Also, exploring some deep learning methods for classification may yield much more accurate results.

Lastly, future research could also focus on improving ground truth labeling by employing experts in biblical studies to label more complex or ambiguous verses manually. This could enhance the quality of the training data and improve the model’s performance.

References

[1] Hutto, C.J. vaderSentiment. GitHub, 2014, https://github.com/cjhutto/vaderSentiment.

[2] ”Sentiment Analysis Made Easy Using Vader.” Analytics India Magazine, 22 Nov. 2018, https://analyticsindiamag.com/sentiment-analysis-made-easy-using-vader/.

[3] TextBlob: Simplified Text Processing — TextBlob 0.16.0 documentation. https://textblob.readthedocs.io/en/dev/.

[4] Sulic, Alessandro. SentiWordNet: The SentiWordNet Sentiment Lexicon. GitHub, 2021, https://github.com/aesuli/SentiWordNet.

[5] ”Sentiment Analysis using TextBlob - Towards Data Science.” Towards Data Science, 7 Feb. 2019, https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524.

[6] https://www.kaggle.com/code/gpreda/explore-king-james-bible-books/report#sentiment-analysis

[7] Borra, Santiago, and Tyler Rinker. ”Sentimentr: Calculate Text Polarity Sentiment.” RDocumentation, version 2.9.0, 2020, https://www.rdocumentation.org/packages/sentimentr/versions/2.9.0.

[8] Borra, Santiago, and Tyler Rinker. sentimentr: An R Package for Sentiment Analysis. GitHub, 2021, https://github.com/trinker/sentimentr.

[9] Shen, Wes. ”Sentiment Analysis: Most Read Book.” RPubs, 14 Sept. 2020, https://rpubs.com/wes-shen/sentiment-analysis-most-read-book\#:∼:text=Overall\%2C\%20the\%20sentiment\%20of\%20the,world\%2C\%20and\%20the\%20human\%20condition

VI. Project Link

Project GitHub:
https://github.com/DavaughnHoots/KJBSentimentAnalysis.git