Data Genre Detection Between Fairy Tale and Mystery

Will Moore jwm18575@uga.edu
YuLong yw98883@uga.edu
Cheng Chen Cheng.C@uga.edu

Abstract

With more literature data sets available to users, heterogeneous classification has become increasingly important for computational linguistics to understand the semantic reasoning behind the text documents. We incorporate bag-of-words, TF-IDF, and Latent Semantic Analysis (LSA) to study ebooks taken from Project Gutenberg's top one hundred authors for retrieving and filtering the essential text information. Then we utilize the Naive Bayes, Logistic Regression, and Long Short Term Memory (LSTM) algorithms to detect between fairy tale and mystery genres. Results have achieved a different degree of success and accuracy. Further research into LSTM and establishing word relationships are necessary in order to advance the understanding of genres, what constructs one, and how to classify one.

1 Background

Genre detection starts from English text feature analysis [1, 2, 3], and each linguistic feature is generalized and evaluated as factor representations. Based on the positive and negative scores, each document groups shows a similar pattern, however, this conclusion is not capable to classify different genre of literature.

Comparing to feature analysis, discriminate Analysis [4] is another method to solve genre detection problems. However, their results have limited success, where it can show which parameter is more important than the others. With the exploration of machine learning techniques, logistic regression and neural nets have been used to classify different genres [5]. The classification result computed the accuracy with different types of genres. Their purposed model achieved 77% of average accuracy.

1.1 Project Gutenberg Digital Library

Data used for this experiment came from the Project Gutenberg digital library of ebooks. It is a collection of over sixty thousand free ebooks[6]. Books are from an assortment of authors, countries, languages, and genres [6]. Books are contained in easy to process text format, and author information is also included. The data set used in this experiment was originally pulled from the works of the top one hundred authors on Project Gutenberg for August twenty nineteen. The version of the data set used can be found on the DigiUGA github.

1.2 Genre Classification

Naive Bayes The naive Bayes classifier is the simplest of these models, in that it assumes that all attributes of the examples are independent of each other given the context of the class. This is the so-called "naive Bayes assumption." [7] In linguistics, there are millions of possible combinations of words so that each word does not impact its successive word which makes each feature independent and equal. In terms of math. the basic naive Bayes model tries to find the probability given event B and the occurrence of event A on top of B. For a literature text classification, the B would all

probabilities of all kinds of words of a book, and A would be category. Gaussian naive Bayes classifier would be used in this case since the distribution is presumed as normal distribution.

TF-IDF TF-IDF stands for term frequency inverse-document frequency. It is an improvement over the simple bag of words model. Bag of words only takes into account the frequency of the word count without looking at the impact of a word [8]. TF-IDF takes into account the impact of a rare word in a corpus as well as a words frequency[8]. The general form for TF-IDF as stated by Scikit Learn, who's packages are used for TF-IDF implementation is as follows:

$$tf - idf(t, d) = tf(t, d) \times idf(t) \tag{1}$$

where $tf(t, d) =$ frequency of term in document and $idf(t) = log(\frac{1+n}{1+df(t)}) + 1$. $df(t)$ is the number of documents in the set for which the term appears [8]. TF-IDF converts the integer values from the word counts into floats [8]. Each float is representative of the strength of the relationship for the word in terms of the entire corpus [8]. The higher the float value, the stronger the impact of the word [8].

Logistic Regression – Will Moore Logistic regression is a linear classification model [9]. It can implement different forms of classification, such as binary and multinomial [9]. For the purpose of our experiment, a binary model was implemented since there are only two classes to classify, mystery and fairytale. The Scikit Learn implementation of logistic regression used in this experiment optimizes the l2 cost function for the logistic regression model [9]. The equation is as follows:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1) \tag{2}$$

[9] solving the cost function gives the weights for each parameter of the model and can be solved in a variety of ways. Scikit learn offers five solvers, liblinear, newton-cg, lbfgs, sag, saga, of which liblinear is the default [9]. Once the weights for the model have been learn, the linear expression is evaluated for each input term. The result is a probability between zero and one. If the probability is greater than 0.5, the input gets the label corresponding to class 0, while a probability greater than or equal to 0.5 has the label corresponding to class 1. In the case of this experiment, 0 is fairytale and 1 is mystery.

Neural Network Genre detection has been widely used in multimedia documentation, for instance, television programs [10], literature [11], and online social media. Features can be extracted from visual information(i.e. the colour, pattern and motion), the structure information(i.e. the sentence punctuation, or the number of words), aural information(i.e. audio stream, or a speech) or cognitive information(i.e. a sketch drawing). Within the text structure information field, each genre contains an unique writing pattern, which can be trained as weights based on documents' unique words as features. Due to the non-linearity of text data sets, neural network approaches have been implemented to solve this issue by using the Keras package. Both artificial neural networks (ANN) and LSTM algorithms are attempted to learn the non-linear decision boundary to classify between fairy tale and mystery stories.

Latent Semantic Analysis Latent Semantic Analysis is a method to study the relationship within the collection of documents. The key idea of LSA is to convert documents from a high dimensional space to an eigenspace[12]. This process is also called lexical analysis, which vectorizes the semantic relations among documents. Then a singular value decomposition analysis will compute the eigenvalues and eigenvectors for the bag-of-words model, which is capable to improve the efficiency of filtering out the irrelevant documents[12]. Further, in probabilistic latent semantic analysis (PLSA), the Bayesian model combined with LSA is often used to solve updating problems for documents[13].

1.3 New Hypothesis

Machine learning has applications for a variety of uses. When combined with the literature domain, it enables us to solve certain problems. Of these problems is the problem of genre classification. That is, can a machine be able to take in a book and be able to tell what genre it is, like romance, horror, or action? Working with a group of domain experts in the literary field, we have come up with an experiment to test the application of machine learning in genre classification. If given enough words from both mystery and fairytale books, then machine learning will be able to correctly identify the genre of a book as wither mystery or fairytale.

2 Methodology

2.1 Data Pre-processing

Data used in the data set was first selected by experts in the literary fields from the collective Project Gutenberg data set of the top one hundred authors. Data was selected on a book by book basis with criteria of the book being either of the mystery or fairy tale genre by the experts. A data set of eighty four books was selected by the literary experts, of which contained fifteen fairy tales and sixty nine mysteries by various authors. All books were in text format and processed using the Natural Language Tool Kit, NLTK, library[14]. Each book contained a header and a footer that contained extraneous information for our experiment and were removed using python's built in input and output methods. After removal of headers and footers, books were then processed to remove punctuation, stop words, numeric words, and words with a length of one or less. All capitalization was removed from each book as well. Books were also tokenized into an array of strings, where each string represented one individual word in the book. The top one hundred unique words have been sorted and shown based on its appearance in Figure 1 & 2.

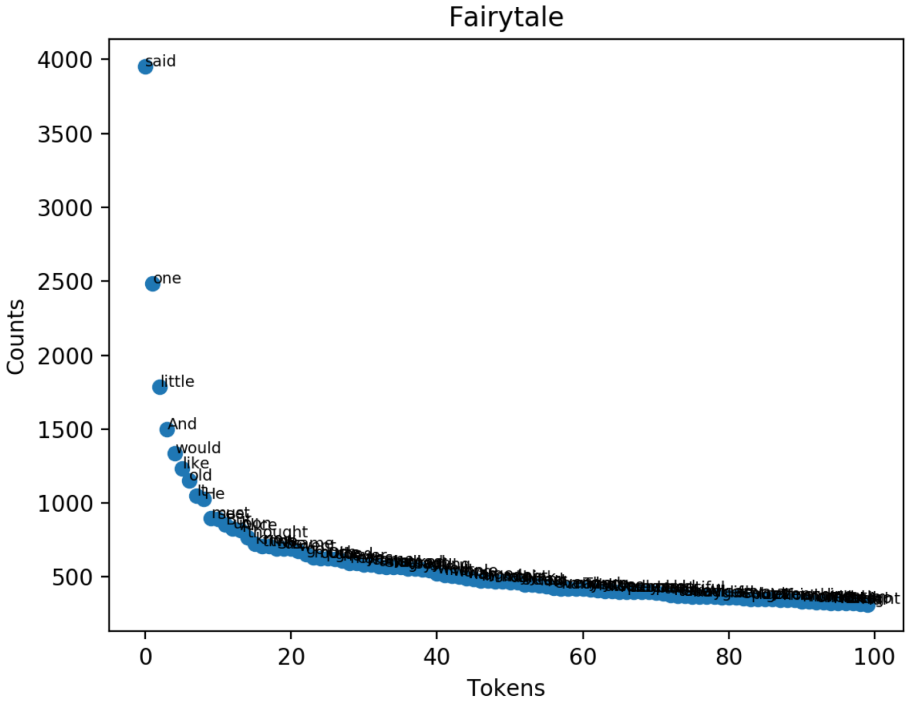

Figure 1: The high frequency words have been filtered and tokenized in descending order.

After the process of cleaning the books and tokenization, books were organized into a single array of documents using a bag of words approach. Utilizing the Scikit Learn method CountVectorizer, a sparse matrix containing the word counts of each unique word for each document was formed. CountVectorizer was passed the parameters specifying to only take words occurring in at least five documents and at most seventy percent of documents. A minimum occurrence of five was chosen in order to eliminate words that occurred too title to be telling about the book genre they represent. A maximum of seventy percent occurrence was chosen in order to prevent any universally identifying words to influence the models used. The result of the bag of words was a ten thousand parameter array of features for logistic regression, and the results for LSTM was an one thousand parameter array of features.

Firstly, it is assumed the documents are randomly ordered. Then we used the words to compute a term frequency (TF) and inverse document frequency (IDF) for a TF-IDF score. The TF is the bag-of-words model, which describes the most relevant words as higher values. However, IDF implies

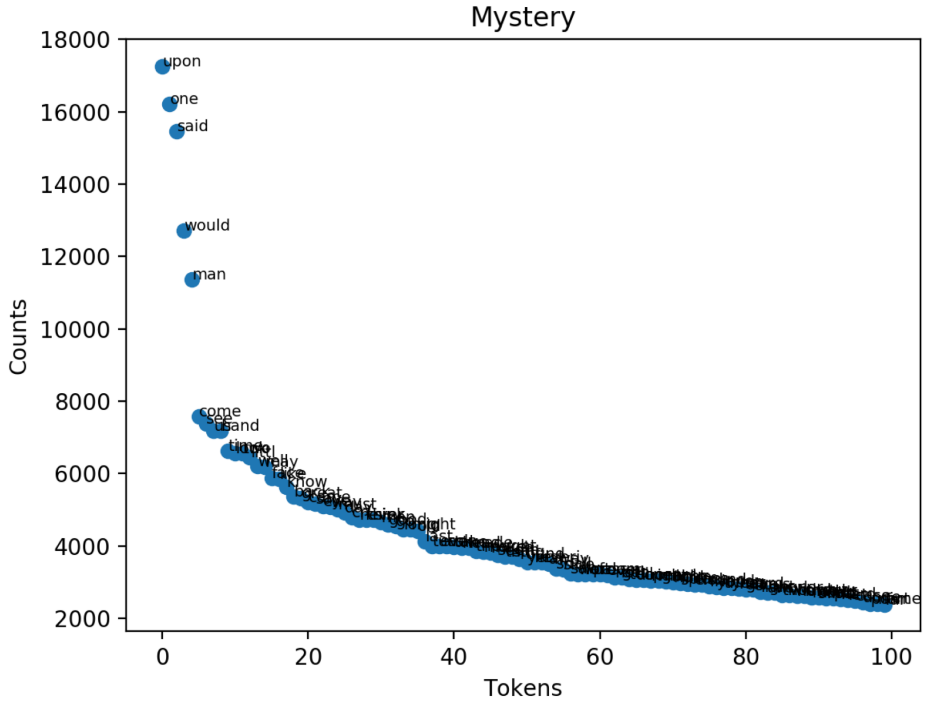

Figure 2: The high frequency words have been filtered and tokenized in descending order.

that the lower frequency words contains more information than those most repeated words. The log inverse frequency term is defined as $idf = log_{10} \frac{N}{df_t}$, where N is the total number of documents, and df_t is the number of documents contains each unique word.

In this work, we implemented the TF-IDF as input values which is a words-by-documents co-occurrence matrix. Since we want to capture most of the meaningful unique words to distinguish between those two genres, TF-IDF could enable classification algorithm to not just pay more attention to leaning high frequency words. IDF matrix would allow higher frequency words will have a lower weight value; this inferred that low-frequency words should have a higher weight. Singular value decomposition (SVD) has been performed for the output matrix. A truncated SVD has been used to compute the first and second principal components to study the correlations among documents in Figure 3 & 5. This technique can reduce the dimensionality which eliminates the noise and improve the computational efficiency. Figure 4 & 6 show the heatmaps visualization representing intrinsic correlations in eigenspace.

2.2 Naive Bayes

First, the raw data was processed to remove all irrelevant information like project information in the preface and the empty lines. SpaCy library was used to process the data so that linguistic text was converted into numerical vectors. Each word would have a unique vector and the semantically similar words would be close, like feline and cat. All books would be sliced with first ten thousand words as representative vector array training data. With ground truth, feed the training data into SKlearn GNB classifier function.

2.3 Logistic Regression

The processed data set and resulting bag of words model was run through a logistic regression model using SciKit Learn's logistic regression package . The model utilized a simple eighty percent training set and twenty percent testing set created using the SciKit Learn train-test-split method for random

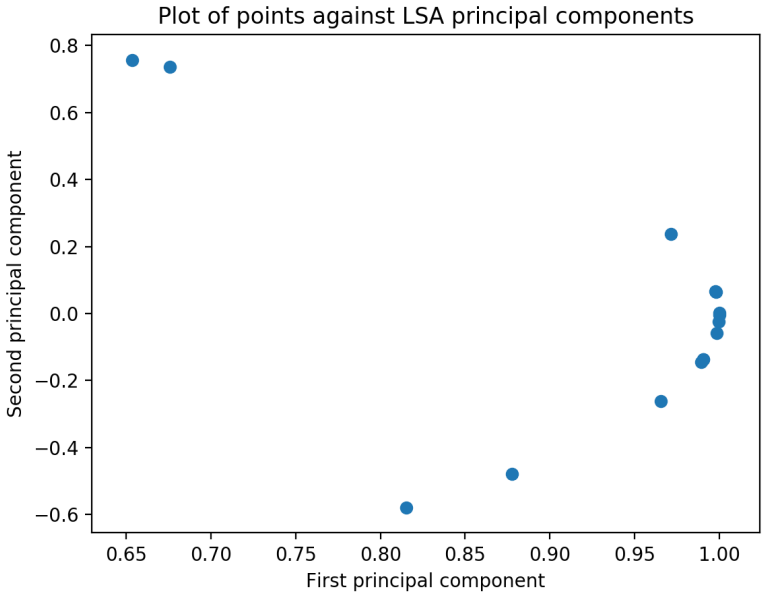

Figure 3: A fairy-tale dataset weighted by TF-IDF and converted in eigenspace

Figure 4: A heatmap plot of fairytale documents shows the similarity based on TF-IDF.

groupings each time the model ran. Ground truth for the model was the established genre for the book as determined by the literary experts. The model was then fit to two different version of the bag

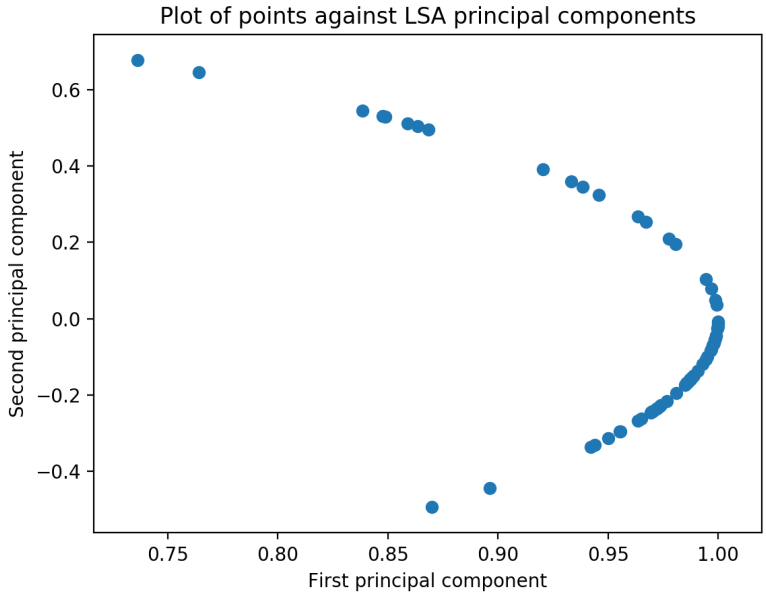

Figure 5: A mystery dataset weighted by TF-IDF and converted in eigenspace

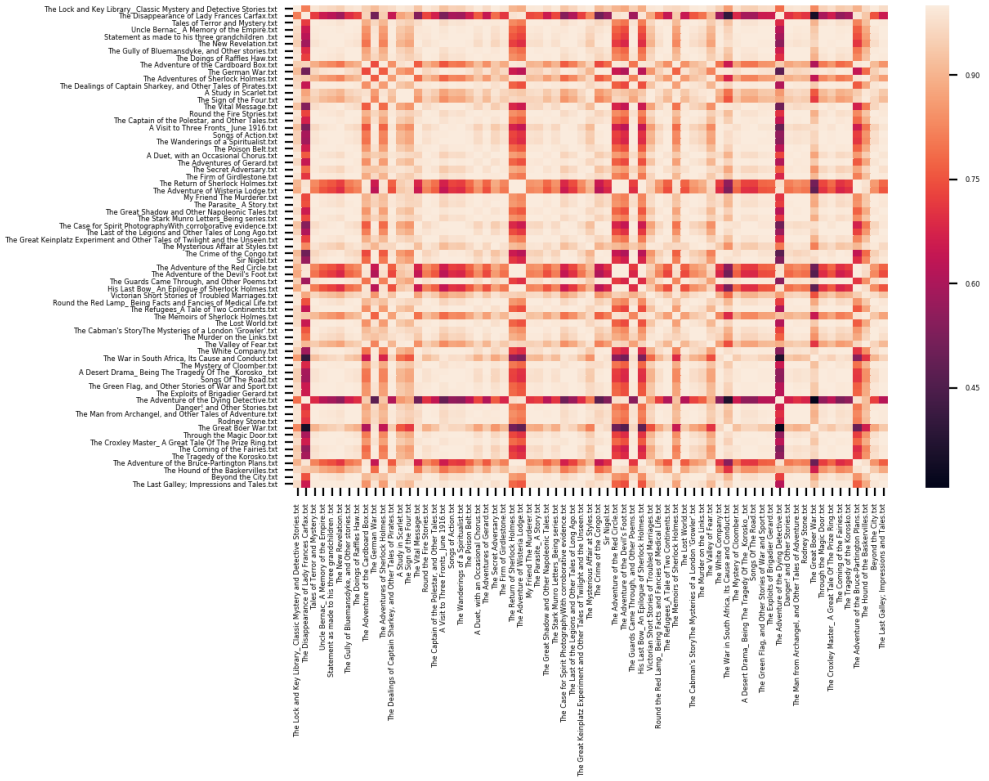

Figure 6: A heatmap plot of mystery documents shows the similarity based on TF-IDF.

of words. The first version used the unaltered bag. The second version ran the bag of words through the Term Frequency - Inverse Document Frequency,TF-IDF, algorithm in order to gauge the change in accuracy when looking at the importance of the word in the book to the entire corpus versus just

the count of the word. The General form of the TF-IDF algorithm used was:

$$tf - idf(t, d) = tf(t, d) \times idf(t) \tag{3}$$

[8] The logistic regression model was then optimized using the liblinear solver from Scikit Learn. Models then predicted genres using the testing set and were graded for accuracy. The cost function optimized was l2:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1) \tag{4}$$

2.4 Neural Networks

The corpus of text filtered by pre-processing was fed into ANN and LSTM as input values, using the Keras package. To balance the training data, there are fifteen documents vectorized by the bag-of-words model with corresponding true labels. Eighty percent of those documents are randomly assigned for the training data set, and the other twenty percent was used for testing. The cross-validation split ratio is also twenty percent. For the ANN model, there are four hidden layers with 128 neurons per each layer, and a dropout unit has been implemented after each fully connected layer to reduce the overfitting. The output layer used a softmax activation function to classify a binary output. A Stochastic gradient descent method, with decay learning rate and momentum term, has been used to compute the cross-entropy loss. Meanwhile, the structure of the LSTM model contains one embedding layer, SpatialDropout1D, LSTM, fully connected, and softmax layer. Adam optimization has been implemented to calculate a cross-entropy loss function. The prediction result were compared with ground truth labels. However, the prediction variance is very high for these neural network approaches, and further study is still needed to make a reasonable prediction.

3 Results

3.1 Accuracy

The accuracy of our various approaches yielded mixed results. Table 1 below provides a summary of the accuracy achieved broken down by approach. Of the approaches used, logistic regression using CountVectorizer to make a simple bag of words model appeared to be most effective with a 93.46% accuracy. Suprisingly TF-IDF LG had a lower average accuracy of 85.36%. The prediction of LSTM algorithm has a very large variance, where it can change from 50% to 90% for each simulation. However results for LSTM accuracy are limited due to instability of the code and hyper-parameter optimization. A further study on how to construct hidden layers and optimize the number of hidden units are also very critical to enhance the accuracy.

The lower accuracy of the TF-IDF approach could be due to an already well constrained space of books which was constrained further to a subset of words that fit our criteria. It might have also been influenced by the relatively small data set size, giving less information for the algorithm to determine relationships of individual words to the entire corpus. The higher accuracy of the CountVectorizer approach to logistic regression can possibly be attributed to the same idea. The size and simplicity of the data set might have predisposed the simpler bag of words model to excel better.

Table 1: Prediction accuracy among different methods

Algorithm	Accuracy
Naive Bayes	66.67%
Logistic Regression CountVectorizer	93.46%
TF-IDF Logistic Regression	85.38%

3.2 Interpretation

Due to the nature of logistic regression we can gain insights on what is going on with the data beyond accuracy alone. Table 2 below summarizes the highest weights assigned to the words in the general bag of words model.

Table 2: Largest weights in CountVector model

Word	weight
holmes	0.0868
cab	0.0618
driver	0.0575
police	0.0333
watson	0.0291

The table confirms the opinions of the literary experts that our team worked with. Their thoughts were that the mystery genre would show more words pertaining to crimes, such as detective, crime names, and repetition of a protagonists name. The repeated use of Watson and Holmes shows through in both of their places on the top five list. The larger weight of police also leans in on this idea that words related to crime are going to be stronger in identifying mysteries. Table 3 below looks at the 5 lowest weighted words from the model. Once again, the experts theorey on the fairytale genre held. The theorey held that the fairytale genre would see more of words pertaining to royalty, animals, court intrigue, and travel to other worlds. Princess has a strong weight to pull the classifier towards fairytale, supporting the royalty and court intrigue theorey. While not necessarily an animal, flowers are a part of the natural world and fit in with nature themes for fairytales.

Table 3: Lowest weights in CountVector model

Word	weight
cakes	-0.0294
clara	-0.0275
flowers	-0.0230
proffesor	-02180
princess	.-0.0218

3.3 Additional Concerns

While results for the logistic regression approaches seemed fruitful, the naive Bayes and LSTM approaches did not yield similar results. Further research and design of the models would be necessary to test the true performance of the models in this experiment. Time was a limiting factor in the experiment, so additional time would be needed to complete the other models to a stable level. As well, the imbalance of the data set used may have also caused bias in our results, as about eighty two percent of the data was mystery books. This could be the reason behind the lower accuracy of TF-IDF, as it could just be guessing mystery about eighty two percent of the time, which is close to its eighty five percent accuracy. However, since the bag of words model showed accuracy significantly higher than eighty two percent, the bias had less impact on that model, however it cannot be completely ruled out as an influence. As for ethical concerns, the models and information here should not be used in any way to classify anything outside of the literary realm and genres of mysteries and fairytales. They are not intended for application in human use or any sort of discriminatory classification use.

4 Conclusion & Future Work

4.1 Conclusion

The results of the experiment partially answered our hypothesis. The linear regression model was adequate in accuracy attempting to classify between mystery and fairytale, at 93.46% for the bag of words model and 85.38% for the TF-IDF model. Naive Bayes was significantly lower with a 66.67% accuracy, and LSTM was unstable in classifying the genres. While the bag of words model exhibited a high level of accuracy, an even higher accuracy would be needed to have confidence that the logistic regression model fully answered the hypothesis. As well, the imbalanced data set may have influenced the accuracy of the models, thus lowering the confidence in the ability to answer the hypothesis.

Even without fully answering the hypothesis, the experiment gave some insights about the problem as a whole. First it does suggest that a machine can detect patterns in word usage for genre detection at some level. It also picked up on patterns that domain experts suspected existed, as shown in the interpretation section. This is promising in considering the most effective approach relied on the independence of words without any sort of embedded relationship among words. It also goes to show the machine's ability to sort out a set of words most indicative to the genre, one of the earlier questions posed by the literary experts. While not fully confirming the hypothesis, the experiment showed that the ideas of the experts were correct in thinking and allows for more work into how a genre is defined, with this work as a basis.

4.2 Future Work

Future work for this experiment include working with the literary experts to increase the size of the data set, as well as ensuring it is balanced. Also among the future work to be done is further research into LSTM and how to effectively model it in a stable and accurate way. As well, more work is needed on the naive Bayes approach in order to see if accuracy can be boosted there. As well, of great boost into gaining insight on genres, further work needs to be done on creating some sort of embedding for the words in order to form relationships between them. Being able to look at the words dependently will help researchers and literary experts alike to be able to gain more insight into what constitutes a genre and how words in a book interact with each other. In addition to establishing relationships among words, further analysis of whether specific types of words influence the type of genre should be explored, such as nouns, adjectives, and verbs and their effect on classification.

References

[1] Biber, Douglas. *Spoken and written textual dimensions in English: Resolving the contradictory findings.* Language (1986): 384-414.

[2] Biber, Douglas. *Variation across speech and writing.* Cambridge University Press, 1991.

[3] Biber, Douglas. *The multi-dimensional approach to linguistic analyses of genre variation: An overview of methodology and findings.* Computers and the Humanities 26.5-6 (1992): 331-345.

[4] Karlgren, Jussi, and Douglass Cutting. *Recognizing text genres with simple metrics using discriminant analysis.* Proceedings of the 15th conference on Computational linguistics-Volume 2. Association for Computational Linguistics, 1994.

[5] Kessler, Brett, Geoffrey Nunberg, and Hinrich Schütze. *Automatic detection of text genre.* arXiv preprint cmp-lg/9707002 (1997).

[6] Project Gutenberg. (n.d.). Retrieved December 5, 2019, from *https://www.gutenberg.org/.*

[7] McCallum, Andrew, and Kamal Nigam. *A comparison of event models for naive bayes text classification.* AAAI-98 workshop on learning for text categorization. Vol. 752. No. 1. 1998.

[8] 6.2. Feature extraction. (n.d.). Retrieved December 50, 2019, from *https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction.*

[9] 1.1. Linear Models. (n.d.). Retrieved December 5, 2019, from *https://scikit-learn.org/stable/modules/linear_model.html.*

[10] Montagnuolo, Maurizio, and Alberto Messina. *Parallel neural networks for multimodal video genre classification.* Multimedia Tools and Applications 41.1 (2009): 125-159.

[11] Diri, Banu, and M. Fatih Amasyalı. *Automatic author detection for turkish texts.* Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP). 2003.

[12] Deerwester, Scott, et al. *Indexing by latent semantic analysis.* Journal of the American society for information science 41.6 (1990): 391-407.

[13] Chien, Jen-Tzung, and Meng-Sung Wu. *Adaptive Bayesian latent semantic analysis.* IEEE Transactions on Audio, Speech, and Language Processing 16.1 (2007): 198-207.

[14] Natural Language Toolkit. (n.d.). Retrieved December 6, 2019, from *https://www.nltk.org/.*