Supervised Machine Learning for the Categorization of Physics Articles

Courtney Nelson
Department of Computer Science
Occidental College
Los Angeles, California 90041
Email: cnelson2@oxy.edu

Abstract—**This experiment aimed to classifying physics articles as technical or non-technical (appropriate level for a general audience). This issue of classification is important for individuals looking to find physics articles at a generally appropriate reading level. With a rapidly increasing surplus of available text on physics at a variety of reading levels, often without sufficient categorization, a tool to discriminate between articles for a general audience and article for a technical audience is needed. This experiment used two supervised machine learning methods to classify the physics articles. The supervised machine learning approaches were the Naive Bayes algorithm and the Logistic Regression Algorithm. In an effort to increase accuracy n-gram techniques, lemmatization, and term frequency - inverse document frequency (TF-IDF) weighting. The main finding of this experiment was that a Logistic Regression approach using TF-IDF weighting, lemmatization, and n-grams ranging from n values of one two three provided that highest accuracy of the methods tested, which was 93%. This finding indicates that Logistic Regression may be a viable candidate for those looking to classify physics articles base on reading level.**

1. Introduction

The goal of this project is to create a system to categorize physics articles into two categories, technical and non-technical. Physics research papers fall into many subcategories that depend on both the intended audience and the subject. Often physics articles are not categorized based on reading level and it puts a burden on the reader two sift through articles that are not at an appropriate reading level. There is room in the field of physics as in many other field for better automate document classification techniques. Due to the very large amounts of physics articles publicly available it can be difficult for those who are interested to find appropriate articles. For organizations that manage large amounts of physics articles like the ArXiv, a potential use for a supervised machine learning to sort incoming articles and to search through articles that have not been previously sorted or may have been incorrectly sorted. The techniques applied in this paper for discriminating between technical and non-technical articles can be applied for any set of binary categories and could be used for other categorization goals for physics documents. The documents used in this experiment were provided by the ArXiv through Google Cloud Storage. The primary method of sorting used in this experiment are a Logistic Regression classifier and a Naive Bayes classifier both provided by Scikit Learn. The pre-processing techniques used in this experiment utilized the PDF Miner libraries as well as the Natural Language Processing Toolkit. This experiment aimed to test existing methods for text classification on a new document set, a large compilation of physics articles, with the goal of determining the supervised machine learning algorithm, of those that were tested, with the greatest accuracy.

2. Related Work

In Pranckevicus and Marcinkevicius 2017 paper entitled ”Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification,” they studied the relationship between the size of a training set and the number of n-grams (a method for accounting a set of words or characters) where n is the number of adjacent terms that are analyzed. The n-gram process is discussed further in Section 3.3. Although Pranckevicus and Marcinkevicius are working with data set of costumer reviews their methods and findings are directly relevant to this experiment since they are looking to classify a large number of documents with a variety of supervised machine learning methods. In their data pre-processing Pranckevicus and Marcinkevicius employ the methods of parsing the reviews ensuring they have an equal number of reviews per each category, and using the Natural Language Processing Toolkit (NLTK) to remove stop words and to stem tokens. This method of text processing is similar to the methods employed in this experiment with small differences in the corpus used for stop words and the method of stemming [1].

Their findings given their data concluded that Logistic Regression was the best method for classifying the elements of their data set. Pranckevicus and Marcinkevicius employed a multi-class classification method which varies from the binary classification described in this paper [1]. The binary classification approach was chosen for this experiment because it is less computationally expensive and has been shown to give more accurate results. Additionally, Pranchevicus and Marcinkevicius showed that for their data set when testing n-gram models from n=1 to n=3, for all supervised machine learning methods analysed, the most successful n-gram method was the collection of all unigrams, trigrams, and bigrams. This approach of incorporating all lower values of n (n value) n-grams into the calculation is widely supported [2]. Although not found to be statistically significant, Pranchevicus and Marcinkevicius found that the classification algorithm with the second highest results was the Naive Bayes method. Additionally Pranchevicus and Marcinkevicius found that 5000 reviews per classification was sufficient for modeling and that the accuracy for their data set had greater dependence on the n-gram system used.

Amini and Gallinari propose an alternative approach to the Logistic Regression classification algorithm in their 2002 paper, ”Semi-Supervised Logistic Regression.” Their approach introduces a new semi-supervised algorithm that combines Logistic Regression and Classification Expectation Maximization. Their approach begins by applying the Logisitc Regression algorithm followed by the unsupervised application of the the Classification Expectation Maximization algorithm, which clusters and estimates the data and the likely hood of its classification. The results of Amini and Gallinari's experiment show a higher average precision for the Logistic-Classification Expectation Maximization algorithm over the standard Logistic Regression algorithm when under 90% of the data has been labeled for their given data set. It is important to note that the largest difference in accuracy between the two approaches is approximately 10% [3]. Their approach uses an interesting combination of standard machine learning techniques to achieve higher accuracy and would be a compelling approach to apply to the physic article data set. This experiment only invokes supervised learning models including that of Logistic Regression.

In Shah's 2020 article on ”A Comparative Analysis of Logistic Regression, Random Forrest and KNN Models for Text Classification,” Shah aims to tackle the issue of document organization using an assortment of machine learning algorithms and data from BBC News. Shah employed multiple methods of text pre-processing before applying the machine learning algorithms to the data. The pre-processing preformed includes removing stop words, replacing the words with their stem form using the Porter Stemmer, and weighting words within the text using Term Frequency - Inverse Document Frequency which is discussed in Section 5.6. Similar to Shah's experiment, the methodology using in this experiment also removes stop words and looks at the impact of Term Frequency - Inverse Document Frequency on the data set. In contrast, this experiment replaces individual words with their lemma form rather than with their stem. A comparison between the Natural Language Processing Toolkit's Lemmatizer and Porter Stemmer are discussed in section 5.2. Shah concluded the for four out of five of their data sets the Logistic Regression algorithm had a greater accuracy than the Random Forrest algorithm and K-Nearest Neighbours algorithms. In one of the five data sets the K-Nearest Neighbours algorithm had the highest accuracy. Shah's 2020 paper supports the notion of Linear Regression being the most accurate algorithm compared to similar models. Linear Regression was chosen as the primary means of classifying the physics articles in large part because of its consistently high accuracy compared to other similar models.

3. Background

To make the text data in the correct format for modeling with machine learning two primary steps must be taken. The text must be tokenized, parsed into sets of single or multiple strings that are meaning for modeling. For example, as in this experiment, the tokens do not include punctuation or numbers. An example of tokenization is the following.

Raw text form:
[”Coulomb's Law (1738-1806):
The force between the two electric charges reduces to a quarter of its former value when the distance between them is doubled.”, ”Gravity is a force.”]

Tokens:
[['coulombs', 'law' , 'the', 'force', 'between', 'the', 'two', 'electric', 'charges', 'reduces', 'to', 'a', 'quarter', 'of', 'its', 'former', 'value', 'when', 'the, 'distance', 'between', 'them', 'is', 'doubled'], ['gravity', 'is', 'a', 'force']]

The next primary steps is to vectorize or extract features from the data by encoding the tokens as integers. This stores the tokens as a vector where the value of an element in the vector represents the number of times a word has occurred in the document.
The bag of words or collection of all the tokens used in the corpus is the following. The tokens are sorted into alphabetic order.

Bag of Words = ['between', 'charges', 'coulomb', 'distance', 'doubled', 'electric', 'force', 'former', 'gravity', 'is', 'its', 'law', 'of', 'quarter', 'reduces', 'the', 'them', 'to', 'two', 'value', 'when']

Then each string or set of tokens can be encoded into integers. For the first element in the vector the number of occurrences of 'between' is recorded. As shown below, the Sentence 1 contains two occurrences of the word 'between' and Sentence 2 contains zero occurrences of the word between. This pattern is applied for all of the tokens in the bag of words.

Sentence 1 (Vectorized) = [2 1 1 1 1 1 1 1 0 1 1 1 1 1 1 3 1 1 1 1 1]

Sentence 2 (Vectorized) = [0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0]]

Once the strings are in the form of vectors they can be processed.

The vectorized sentences are split into random training and testing subsets where the ratio of trained to tested subsets is specified.

3.1. Naive Bayes

Naive Bayes classifiers that apply Bayes theorem. Bayes theorem is the following:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (1)$$

Where P(c—x) is the probability that a class (c) corresponds to a given input(x), P(x—c) is the probability that the x is a member of the class c, P(c) is the probability of the class itself, and P(x) is the probability of the input itself. The Naive Bayes classifier assumes that the elements being classified are independent from one another. Although the Naive Bayes algorithm is simple it can get relatively accurate results that are comparable with other more sophisticated methods with less computational load.

3.2. Logistic Regression

Logistic Regression is a supervised machine learning algorithm that classifies elements be predicting the probability that they are a member of a particular class. Although, the Logistic Regression algorithm includes regression in its name it is not a regression algorithm. It is a classification algorithm. The elements are being assigned to discrete categories. The logistic function is the following:

$$y = 1/(1 + e^{-x}) \quad (2)$$

Logistic Regression is an appropriate algorithm to use on this data set because the data falls into discrete categories as labeled by the arXiv. Binary Logistic Regression only contains two categories. Either a training element is a member of one category or another. This is the main technique that will be applied to the data set. Additionally a multinomial approach can be taken which attempts to sort elements into greater than two categories.

3.3. N-Gram

N-Gram models look at adjacent words to compute similarity. For example the n-grams of the sentence below are the following:

$$< s >\text{ Gravity is a force. }< /s >$$

The monogram is a list individual words ['gravity', 'is', 'a', 'force']. This method does not store the order of the words and therefore is limited in its ability to capture the sentiment of the statement. A given sequence of words may be important for sorting documents.

A bigram method stores sets of two words, [[$< s >$, gravity],['gravity', 'is'], ['is', 'a'],['a','force'],['force',$< /s >$]], where <s> represents the beginning of a sentence and ¡/s¿ represents the end of the sentence. The logic applies to n-grams of any value of n. Using n-grams to more effectively compare documents, but they can also be computationally taxing without much increase in the accuracy of the algorithm.

4. Methods

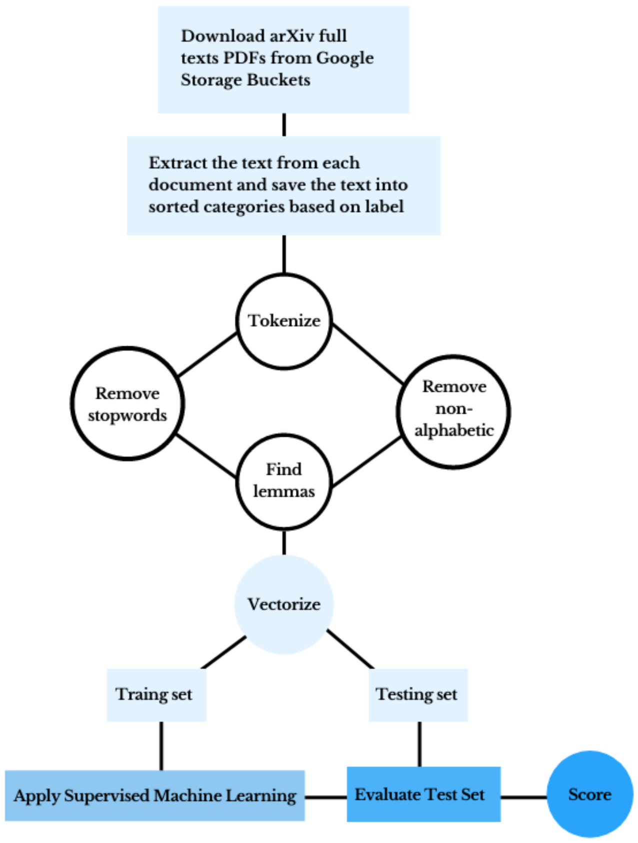
Download arXiv full texts PDFs from Google Storage Buckets
Extract the text from each document and save the text into sorted categories based on label
Tokenize
Remove stopwords
Remove non-alphabetic
Find lemmas
Vectorize
Traing set
Testing set
Apply Supervised Machine Learning
Evaluate Test Set
Score

Figure 1. Flow chart of the data collection, processing, and experimentation of the physics documents.

5. Data Set

The data set of research papers was provided by the arXiv, which is an open archive that includes and extensive collection of physics scholarly papers and is managed by Cornell University. Articles submitted to the arXiv are moderated by a team that classifies articles into specific subject area. The arXiv is a reliable data set as it it moderated by a committee that ensures the article is of value and relevance to physics and is sub-field. These are the categories of Physics that don't fall into the main areas.

The sub-fields of the physics data set are describe in the following list, where the subjects have been sorted into two categories which have been defined for the purpose of this project as suitable for a "Non-Technical Audience" or a "Technical Audience". The General Physics, Popular Physics, History and Philosophy of Physics, Physics and

Society, and the Physics Education sub-fields were grouped into the general audience category. The remaining sub-fields were grouped into the technical physics audience category. The purpose of this organization is to make a tool for users to discriminate between documents based on the audience's level of prior knowledge.

Non-Technical Audience:

- General Physics
- History and Philosophy of Physics
- Physics and Society
- Physics Education
- Popular Physics

Technical Audience:

- Accelerator Physics
- Atmospheric and Oceanic Physics
- Atomic and Molecular Clusters
- Atomic Physics
- Biological Physics
- Chemical Physics
- Classical Physics
- Computational Physics
- Data Analysis
- Statistics and Probability
- Fluid Dynamics
- Geophysics
- Instrumentation and Detectors
- Medical Physics
- Optics
- Plasma Physics
- Space Physics

5.1. Data Pre-Processing

The arXiv recently released their papers for bulk downloads for research purposes through Google Cloud storage. Although the papers are organized by subject on their web page, the Popular Physics and the General Physics Subjects are not presorted into folders. Since many papers have the same formatting, which includes a clearly listed subject, only documents in that format with a clearly listed subject were included in the data set. The PDFs were processed using the PDFMiner tool which extracts text from PDFs. Additional functions used to process the PDFs were provided by data scientist Mate Pocs. The text is mined from the PDF and then stored in text files with unique IDs into the category listed on the documents. Some documents have multiple versions which have minor changes they are saved with the same ID number but with a version number greater than one. Only the version one document were stored to prevent duplicates.

Each document is transcribed to a text file and then loaded into the technical or non-technical data frame. The raw data is represented using the utf-8 encoding. Elements in the technical data frame have their categories relabeled as "technical" and elements in the non-technical categories have their categories relabled as "non-technical". This allows for a binary analysis since there are only two options for the category. This will result in a higher accuracy of the sorting algorithm and is additionally less computationally expensive. To ensure there is an equal amount of documents in the "technical" and "non-technical" data sets a random assortment of "technical" data points are selected until both data sets have an equal number of data points. For this experiment there were 4,806 total articles half of which were technical and the other half of which were non technical. All of the data is then collected into a single data frame so that the data can be randomly split into training and testing sets. In all experiments described, unless otherwise specified, 75% of the data is used for testing and 25% of the data is used for testing. For the given data set this means that 3,604 documents were used for training the algorithms and 1,202 where used to test the categorization methods. After the data was split into the training and testing sets the text from each document was processed. In all of the processing, the text is tokenized using NLTK's tokenizer which allows for custom sequencing. The tokens do not include non-alphabetic characters or stop words [4]. Using custom stop words such as 'phys', 'pop', and other words that are specifically connected to the categories are removed to prevent influencing the model. Further removing non-alphabetic characters prevents documents in the same category with similar dates or ID numbers from effecting the test results. The next step in cleaning the data before running the machine learning algorithms is lemmatization.

5.2. Lemmatization

Lemmatization is a system for grouping together words and forms of words with the same of very similar meaning. Lemmatization of the words in each document was preferred over stemming because it takes into account the part of speech of the word as well as its likely meaning in its lemmatized form.

The following are examples of lemmatization using the same nltk lemmatizer used to process the tokens of each document [4]. For example a document contains the following sentence, "They recorded the masses of many stars." The lemmatization and stems of the words in the example sentence are given in Table 1. The word stems displayed in table one are generated by NLTK's Porter Stemmer [4].

Token	Stem	Lemma
they	they	they
recorded	record	recorded
the	the	the
masses	mass	mass
of	of	of
many	mani	many
stars	star	star

Table 1: This table shows the difference between word stems and word lemmas for a given token

5.3. Standardization

To make this data easier to process the data is standardized by setting the mean to zero and scaling the one standard deviation to a z value of to one. This is represented by the equation:

$$z = \frac{(x - \mu)}{\sigma} \tag{3}$$

Where x is the raw value, μ is the mean of the data used for training, and σ is that standard deviation of the data used for training [2]. Then the training data is fed into a classification algorithm which is then used to score the testing data.

5.4. Naive Bayes

The Naive Bayes algorithm for categorization applied to the data set was from the sklearn library [5]. For the data since there are multiple training and testing documents with unique vectors. The Bayes equation for multiple inputs is given by the following:

$$P(c|x_1, ..., x_n) = \frac{P(c)\prod_{i=1}^{n} P(x_i|c)}{P(x_1, ...x_n)} \tag{4}$$

For the analysis of the data using the Naive Bayes two approaches were taken. The standard Naive Bayes approach and the Gaussian Naive Bayes approach. Since the number of classes is fixed at two and the number of features is much greater, the Multinomial, Complement, and Bernoulli version of the Naive Bayes algorithm were not applied [2].

5.4.1. Gaussian Naive Bayes. A Gaussian Naive Bayes algorithm was applied to the data set. A Gaussian Naive Bayes algorithm assumes that the likelihood of features has a Gaussian distribution.

$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} exp(-\frac{(x_i - \mu_c)^2}{2\sigma_c^2}) \tag{5}$$

Where the σ_c and μ_c where μ_c is the mean of the distribution and σ_c is the standard deviation of the distribution [5].

5.5. Logistic Regression

Additionally, the sklearn's Logistic Regression algorithm was applied to the text. The logic regression algorithm can be modified directly by the parameter, C, which determines the strength of the regularization of the fit. High values of the C parameter correspond to fitting the data set as best as possible, which can lead to over fitting. In contrast, lower values of C do not emphasize an exact fit as much as finding a near zero coefficient vector. Raising the value of C prioritizes correctly classifying document in the training set, whereas a low value of C adjusts the algorithm to accommodate the data points.

There are two main functions for implementing Logistic Regression through sklearn, LogisticRegression() and LogisticRegressionCV() [5]. The LogisticRegression() function was used whenever the C parameter was specified. If the C parameter was not specified the LogisticRegressionCV() algorithm was used which tests many C values to find the optimal C value.

5.6. TF-IDF Weighting

Term Frequency - Inverse Document Frequency (TF-IDF) weighting re-weights features so that common words that may not be important for text classification are weighted less than less frequent words that are more meaningful for text classification. Inverse document frequency is defined in the Scikit Learn algorithm by Equation 6 [5].

$$IDF(t) = log\frac{1+n}{1+DF(t)} + 1 \tag{6}$$

Where n is the total number of documents in the training set, and DF(t) is the number of documents which include the token, t. The document frequency is then multiplied by the inverse document frequency resulting in the TF-IDF value. The Inverse Document Frequency values are stored in vector form and are then normalized using the Euclidean normalization function, Equation 7.

$$v_{norm} = \frac{v}{|v|} = \frac{v}{\sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots + v_n^2}} \tag{7}$$

For example let:

Document 1 = ”Coulomb's Law (1738-1806):
The force between the two electric charges reduces to a quarter of its former value when the distance between them is doubled.”

Document 2 = ”Gravity is a force.”

Bag of Words = ['between', 'charges', 'coulomb', 'distance', 'doubled', 'electric', 'force', 'former', 'gravity', 'is', 'its', 'law', 'of', 'quarter', 'reduces', 'the', 'them', 'to', 'two', 'value', 'when']

Number of Documents (n) = 2

TF('between') = 2

DF('between') = 1

IDF('between') = $log\frac{n+1}{DF('between')+1} + 1 = 1.176$

TF-IDF('between') = TF('between')$\times$ IDF('between') = 2.352

Repeating this for the following tokens in the set results in the following vector. Then normalizing the vector results in the following vectors.

TF-IDF (Document 1) = [0.35348442 0.17674221

0.17674221 0.1767422 0.17674221 0.17674221 0.12575354 0.17674221 0. 0.12575354]

TF-IDF (Document 2) = [0. 0. 0. 0. 0. 0. 0.50154891 0. 0.70490949 0.50154891 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

6. Results

The accuracy of the Gaussian Naive Bayes, Logistic Regression, and the combination of Logistic Regression and TF-IDF for a combination of n-gram ranges are shown if Figure 2. The Logistic Regression algorithms used in this experiment had calibrated C values using the LogisticRegressionCV function.

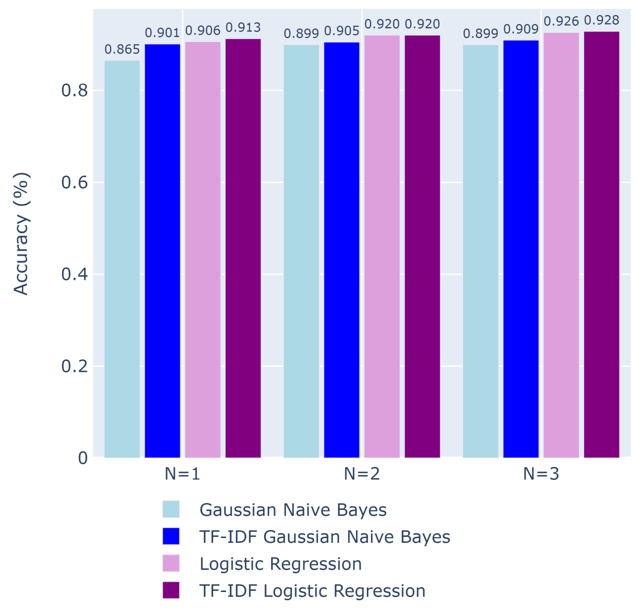

Figure 2. Bar graph of the accuracy of Gaussian Naive Bayes and Logistic Regression algorithms on the data set with and without TF-IDF weighting for n-gram values ranging from one to three. For this experiment the only features included were those that appeared in at minimum thirty documents. Plotting tools provided by Plotly [6]

As Figure 2 illustrates, the combination of the TF-IDF and Logistic Regression methods produce the most accurate results across all n-gram ranges. Further the highest accuracy of all n-gram ranges and categorization methods is the collection of uni-grams, bi-grams, and tri-grams with the combined TF-IDF and Logistic Regression algorithms. Taking into account the results of Figure 2, the next experiment analyzes the relationship between C value, n-gram range, and accuracy. The results of which are shown in the heat map in Figure 3.

Figure 3 shows that the highest accuracy by varying the C parameter and n-gram range is the collection of uni-grams, bi-grams, and tri-grams, with a C parameter with the value of 100. Relatively high values of C corresponds to maximizing the fit of the algorithm to the data. The most effective value of the C parameter is 21.54434690031882.

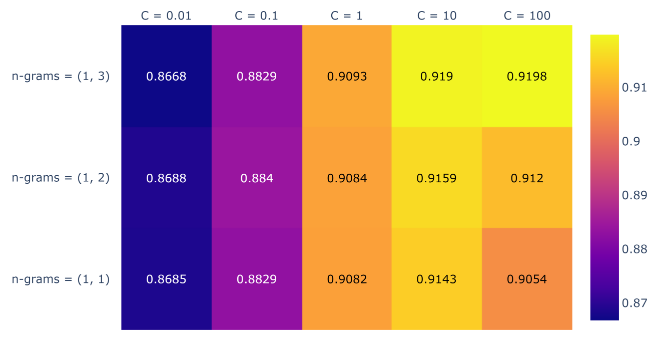

Figure 3. This heat map shows the relationship between the n-gram value and the C value on the accuracy of the Logistic Regression algorithm. The color scale corresponds to the accuracy of the algorithm. For this experiment the only features included were those that appeared in at minimum thirty documents. Plotting tools provided by Plotly. [6]

This value will be labeled as C-Effective for future use. The accuracy of the the Logistic Regression algorithm with the C-Effective value and the n-gram range set to three is 92.845%. For Figure 3 the features only include phrases that are in at least thirty documents. This setting allows for the values to be computed without causing a memory error.

The twenty most impactful features for determining

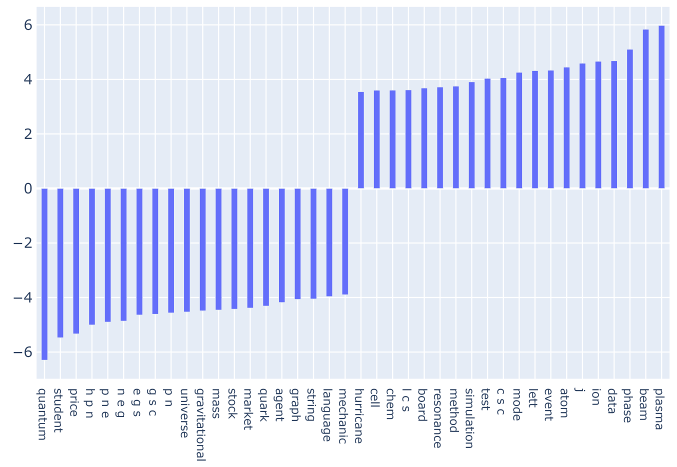

Figure 4. This is a bar graph of the most influential terms and their relative weights. The bar graphs with values in the negative direction correspond to non-technical categorization. The bar graphs with values in the positive direction correspond to technical categorization. For this experiment the only features included were those that appeared in at minimum thirty documents. Plotting tools provided by Plotly [6].

whether a document non-technical or technical is listed in Figure 4.

As shown in figure for there are multiple strongly correlated features that are single characters or trigrams of single characters. The accuracy of the C-Effective valued tri-gram Logistic Regression after all tokens with length less than one are removed is 91.681% compared to that of the algorithm without tokens of all lengths, 92.845%.

7. Conclusion

In conclusion, the most accurate of the tested methods was the Logistic Regression algorithm with TF-IDF weighting, n-grams ranging from one to three, lemmatization and a C parameter of 21.5. The accuracy of that method was 92.845%.. This method was closely followed by the same method without TF-IDF weighting. This method had an accuracy of 92.6%. The best results of the three primary methods of categorization ranked in the following order: TF-IDF weighted Logistic Regression, Logistic Regression, and Naive Bayes. Figure 2 shows that there are small differences in the accuracy of both algorithms when applying the TF-IDF weighting once the n-gram range is larger than two. Further, Figure 2 illustrates that the most significant parameter, of those tested, that improves accuracy is using a Logistic Regression model over a Naive Bayes model for a given n-gram range. Figure 3 illustrates that larger n-gram ranges and relatively C values correspond to higher values of accuracy. The trends in Figure 2 indicate that there may be a local maximum between C = 10 and C = 100. This was verified using the sklearn calibrated C value which showed that the optimal value of C is 21.5. Figure 4 highlights the key terms that contribute most to the categorization of documents. It is noticeable that a large number are n-gram forms of strings of length one. It is interesting that including tokens of length one is important to the accuracy of the algorithm. Including tokens of length one increases the accuracy of the algorithm by approximately 1.2%.

8. Discussion

The experiment successfully categorized physics documents as technical or non-technical with 93% accuracy. The results show that Logistic Regression with TF-IDF weighting of tokens was the most accurate method. The notion that Logistic Regression is more accurate than Naive Bayes is consistent with previously published works. Although it is important to note that the requirement on the computer's memory is much greater for Logistic Regression that for Naive Bayes particularly for larger n-gram ranges. This is important because it more valuable for some users to have a faster run time and less strain on their memory for only a 3% deduction in run time.

Additionally it is interesting to note that a large range in accuracies can occur in the Logistic Regression algorithm with the attainment of both the n-gram range and the C value of the algorithm. An optimized C value can result in a 5% percent increase in accuracy for a given n-gram range. This is a significant parameter to consider when calibrating Logistic Regression algorithms to optimize accuracy. It was expected that independent of parameter settings that the Logistic Regression algorithm would yield more accurate results the the Gaussian Naive Bayes approach. This expectation was incorrect, but the results due show that under optimized calibration the Logistic regression algorithm has higher accuracy.

It was also expected that removing tokens of length one would improve the accuracy of the model. The results show that this is not the case. This is significant because it indicates that their may be significant meaning in the single character strings. This may be the case with the string ”j”, which to some would not appear to have any significant meaning, but within physics it is a primary unit vector used in many calculations. In contrast, it could have easily been predicted that ”student” would be a keyword that indicates that an article is non-technical.

The implications of this experiment include the possibility this categorize or a categorizer of a similar structure to successfully sort physics articles. Additionally, this experiment shows that a range of methods can be used, some less computationally expensive than others, to achieve relatively high accuracy.

Threats to the validity of this experiment include the arbitrary selection of sub-fields as technical or non-technical. In addition, the sub-field categories may have had a large effect on the categories and it is possible that the method would not be as effective on documents that are non covered by the categories listed as ”technical” or ”non-technical.” Further it is important to know that the data size was limited and the amount of document features that could be taken into account was limited by memory capabilities.

8.1. Future Work

Important future work could include: applying the methods to a larger data set, looking at sentence structure and writing style as a document feature, and preforming multinomial classification to classify documents into smaller subgroups. Using a larger and more diverse data set would be an interesting next step because it could shed light on the impact of the sub-fields on the categorization. Additionally a larger data set may be able to minimize the effect of certain subject being more often technical or non-technical which would allow for more precise categorization based on writing level rather than subject.

The second avenue for future work would be to focus on sentence structure and writing style. Using a combination of part of speech taggers and other tools for quantifying the relationships between words in a sentence researchers could start to quantify the writing style differences that separate technical and non-technical articles. Quantifying this difference could be useful for not only categorizing documents but also for quantifying how well suited a document is for its intended audience. This has the potential to be a helpful tool for physicists that are working to communicate their research to an audience with a specific background.

The third possibility for productive future work is to expand from binomial classification to multinomial classification. Multinomial classification has the potential to classify physics articles into their given sub-field or into one of many reading levels. This would be very useful for journals that manage many articles in a wide array of subjects. Additionally this could be useful for making it easier to search for physics articles based on reading level. There is a wide range of background and technical knowledge that readers have and part of the struggle for many readers is to find resources that are appropriate for their reading level. Any tool that is able to help readers find articles that are appropriately matched with the readers prior knowledge has the potential to be very useful.

References

[1] T. Pranckevičius and V. Marcinkevičius, "Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification," *Baltic Journal of Modern Computing*, vol. 5, no. 2, p. 221, 2017.

[2] A. Müller and S. Guido, "Introduction to machine learning with python: A guide for data scientists," 2016.

[3] M.-R. Amini and P. Gallinari, "Semi-supervised logistic regression," 2002.

[4] E. L. Bird, Steven and E. Klein, "Natural language processing with python." O'Reilly Media Inc, 2009.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly