Multilabel Text Categorization of Research Articles

Rachel Chiang

August 2023

1. Introduction

There are a lot of scientific articles, especially online, so it can be hard to find relevant or related articles. Accurately tagging articles can help with recommendation systems and search engines and thus help people find new information more conveniently, so people can spend more time learning and reading than searching. The goal of this project is to correctly classify at least 70-80% of the articles within the next month to help quickly automate article tagging.

2. Data

The data used for this project are scientific research article titles and abstracts, with at least one of six labels: Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, and Quantitative Finance. This dataset was found on Kaggle, and it was scraped by Analytics Vidhya, which is a community of data professionals [1]. The dataset "train.csv" contains 20,972 article samples with nine columns: a unique ID, the title, the abstract, and six topic columns corresponding to the different science subjects, populated by a 1 or 0 to indicate if the article is or is not tagged with the category (Appendix A). A single article may belong to multiple topics; there may be more than one topic category, so the category vector may be 011000, which would indicate that it belongs to physics and mathematics. (Further mentions of a specific label encoding example will take the form of a binary-appearing number like just prior or in an array form, like [0, 1, 1, 0, 0, 0].) The dataset "test.csv" contains 8989 articles, and the first three columns are the same as those from the "train.csv", but the articles are unlabeled.

3. Data Wrangling and Exploratory Data Analysis

A. Data Wrangling

Verification and a broad understanding of the data was sought at this step. There were no missing values and no problems in the overall statistics of the values, and the inputs that would be expected to be unique were unique. There was one suspicious sample, which contained only five words total between its title and abstract, and these five words did not seem very indicative of its topic or important, so it was dropped. The titles and abstracts were concatenated with one another, and some light text cleaning and preprocessing were conducted. Using genism and NLTK, the texts were converted to lowercase tokens, the stop words were removed, and the resulting pruned lists of words were lemmatized. The process took about 15-20 minutes over both input files.

B. Exploratory Data Analysis

First, the distributions of the labels and words were examined. The dataset was imbalanced, with Quantitative Biology and Quantitative Finance having very few samples, and Statistics also had relatively few samples compared to the remaining three. In the "train.csv", 8594 articles were tagged with Computer Science (40.9%), 6013 with Physics (28.7%), 5618 with Mathematics (26.7%), 5206 with Statistics (24.8%), 587 with Quantitative Biology (2.8%), and 249 with Quantitative Finance (1.2%). This means that our datasets are imbalanced. In particular, Computer Science was overrepresented, and the two Quantitative categories were underrepresented.

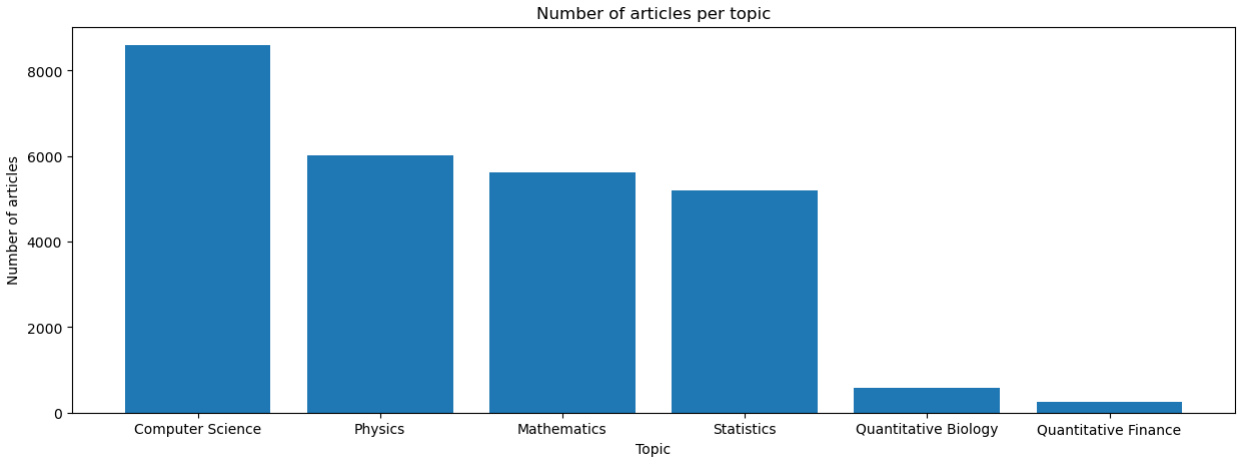


Figure 1. Number of articles per topic.

There was a total of 26267 labels, so there were 5295 more labels than there were articles. 15928 of the articles were marked with one topic (76%), 4793 with two topics (23%), and 251 with three topics (1%).

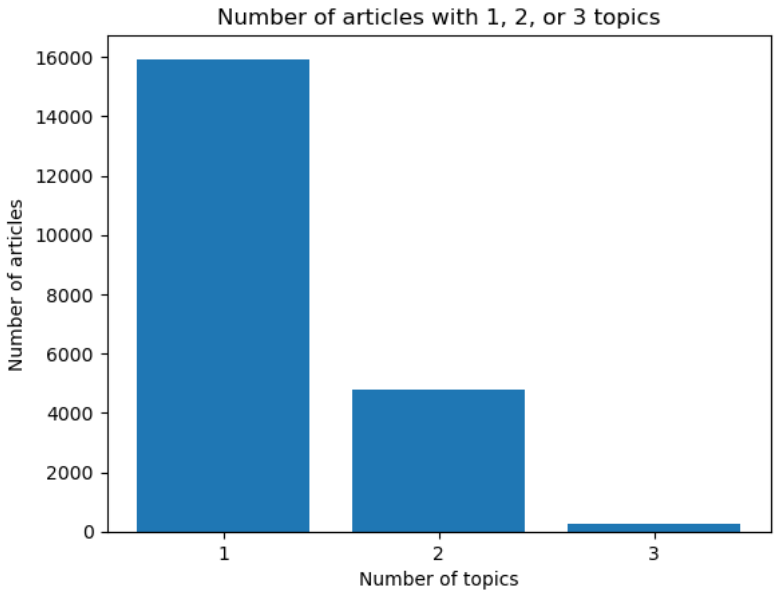


Figure 2. Number of articles tagged with exactly 1, 2, or 3 topics.

Overall, there was an average of 166 total words per sample between the title and abstract per article, with a minimum of 12 and a maximum of 467 (Appendix B). There were slight differences in the distributions of quantity of words per topic, with Mathematics having the lowest range and Quantitative Biology having the highest range.

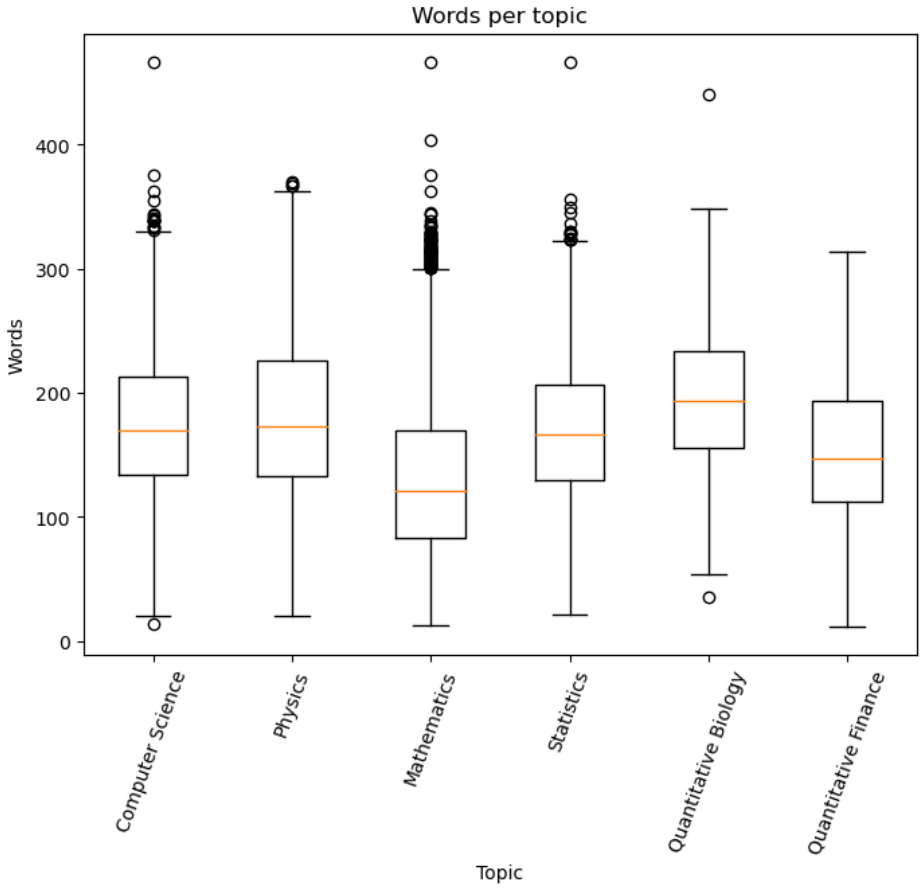


Figure 3. Box plot of number of words in a sample for each topic.

The ten most common tokens in all the articles in the "train.csv" were model, data, method, network, system, result, problem, based, time, and show, and "test.csv" had a similar top ten results. These tokens related to science articles, which at the very least verified that the articles were mostly about science. Per topic, the same words appeared a lot, as would be expected, so I filtered the universally most common tokens so that the top *n* tokens per category would be more expressive. Please view Appendix C to see the changes that occur before and after filtering. The word clouds below, in Figures 4-9, show the top 200 words per category, with the 30 most common global words filtered out. The Computer Science articles looked like they pertained to some new technologies, since it contained the tokens "neural" (likely networks) and "image," and they have some tokens with fundamental concepts, such as "performance" and "graph." The Physics articles looked like they have to do with "energy" and astrophysics. The Physics, Mathematics, Quantitative Biology, and Quantitative Finance articles seemed to express the most different-looking top tokens, with the latter two expressing the most specific words pertaining to their topics. Meanwhile, Statistics and Computer Science seemed to share a common theme of relating to Data Science concepts, such as machine learning, deep learning, and neural networks.

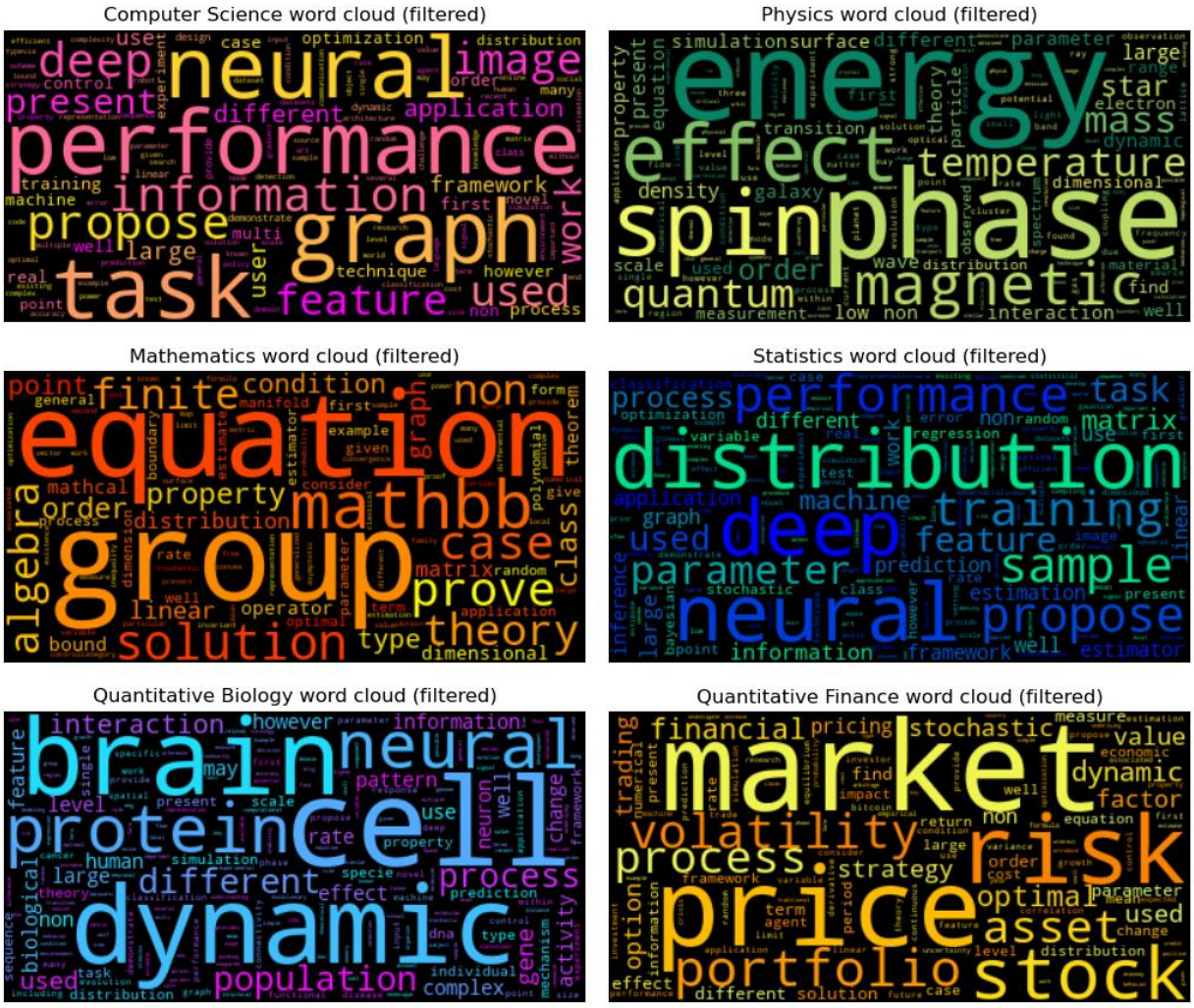


Figure 4-9. Word clouds of 200 tokens with top 30 global common words filtered.

4. Preprocessing and Modeling

This step can be divided into two broad sections: one dealing with single-label categorization and one dealing with multilabel categorization. The former was done with scikit-learn, using TF-IDF vectorizers and Naïve Bayes classifiers on four and six categories, and the latter was done using spaCY pipelines on four categories.

A. Single-Label Categorization

The first six classifiers used a TF-IDF vectorizer that conducted some basic text preprocessing, such as stripping accents, lowercasing before tokenizing, and removing scikit-learn's default English stop words, and the minimum document frequency was 2, and both unigrams and bigrams were accepted. The prediction labels were encoded to numbers 0 to 5, and articles of multiple categories were dropped, so 15,927 (76%) of the articles were used, and the dataset was split into 80% and 20% training and testing sets. Then, two Naïve Bayes classifiers were used out-of-the-box, MultinomialNB and ComplementNB. The latter performed slightly better, which would be as expected since it is claimed to be better-suited for imbalanced datasets. The class

imbalance was clearly problematic, as the confusion matrices for these showed that there were little to no labels predicted for Statistics, Quantitative Biology, and Quantitative Finance. The feature spaces for these were very large, exceeding 130,000, so SelectKBest reduced the number of features to 20000, which turned out to work well across all four classifiers.

Thus, the two strongly underrepresented classes, Quantitative Biology and Quantitative Finance, were dropped for the next experiment. These two accounted for only 651 articles in the single-label subset, so the remaining dataset contained 15,276 articles. The same TF-IDF vectorizer was used for the first six classifiers, but a slightly different one was used for the final two. Using RandomizedSearchCV, a couple of different parameters for the TfidfVectorizer were found to work well, max_df=0.2 and min_df=5, but the other parameters remained the same.

Overall, the ComplementNB classifiers tended to perform better, and filtering K-best features improved each of the models. Dropping the two underrepresented labels improved the models' recalls slightly, and it even improved precisions slightly for the three majority labels (Computer Science, Physics, and Mathematics). No matter what, every model predicted Statistics relatively poorly, and each ComplementNB variation had precision and recall floating around 0.85 and 0.37 for Statistics. Below is a table of these Naïve Bayes classifiers' balanced accuracy and macro F1, which were chosen due to the imbalance, and Appendix D contains further model details and some visualizations for this section.

Model	Num Classes	K-best?	Balanced Accuracy	Macro F1
Multinomial NB	6	No	0.4576	0.4318
ComplementNB	6	No	0.5546	0.5778
Multinomial NB	6	Yes	0.4689	0.4522
ComplementNB	6	Yes	0.6427	0.6789
Multinomial NB	4	No	0.6928	0.6687
ComplementNB	4	No	0.7585	0.7715
Multinomial NB	4	Yes	0.7450	0.7531
ComplementNB	4	Yes	0.7914	0.8082

Table 1. Metrics for Naïve Bayes Classifiers, using TF-IDF vectorizers on a single-label, multiclass problem.

B. Multilabel Categorization

The multilabel dataset used 20,316 articles because the original size 20,971 was reduced by 655 (3%), which were the articles labeled with only Quantitative Biology, only Quantitative Finance, or exactly both of those. Preprocessing involved the typical text preprocessing, for which I used spaCY's NLP "en_core_web_sm" pipeline. For compatibility with spaCY's multilabel text categorizer, the dataset had to be split into three parts: a training set (75%), a dev set (15%), and a test set (10%), and converted into serialized forms accepted by the categorizer.

Three models were created: a unigram bag-of-words model, a bigram bag-of-words model, and a CNN model. Their results were similar, but the CNN model was the slowest to train, the unigram model was the fastest to train, and the unigram model performed better than the bigram model. Appendix E contains the precision, recall, and F1-score of the three models. Slight variations to

the parameters of the unigram model were tested, but they did not yield different enough results or valuable enough improvements. The only model that attained better scores had a slower learning rate, but naturally, this increased the time of training, and the improvement was very small, so it was deemed not worth the time.

Adjusting the threshold of the scorer improved the model in terms of recall, and mostly it helped the Statistics predictions. The threshold determines whether a category's prediction score is interpreted as true or false. A few thresholds were tested, but the scorer with a threshold of 0.45 struck a desirable balance between precision and recall for all the categories. There were some prediction scores that were not high enough to meet any reasonable threshold, resulting in a prediction of 0000 (or no category), but every article should have at least one category, so the scorer was also amended to allow enforcing a minimum of one category returned—the category with the highest score. This also improved the recall of the model, but the tradeoff was a lower precision. Having a higher recall may be more desirable than a higher precision for most applications of a categorization system. For example, a search engine should return at least somewhat relevant articles rather than no articles, or a section displaying more articles to read should not be empty. However, it is important to strike a balance on precision and recall and consider the applications. Even though an empty search result could be undesirable, returning too many incorrect articles could reduce users' trust in the system.

	Precision	**Recall**	**F1-score**	**Support**
Computer Science	0.85	0.90	0.87	898
Physics	0.91	0.85	0.88	607
Mathematics	0.84	0.84	0.84	558
Statistics	0.74	0.83	0.78	496
Micro Avg	0.84	0.86	0.85	2559
Macro Avg	0.84	0.85	0.84	2559
Weighted Avg	0.84	0.86	0.85	2559
Samples Avg	0.88	0.89	0.86	2559

Table 2. Classification report for the unigram bag-of-words model with a scorer threshold=0.45 and a forced minimum of one label returned as true for the category with the highest score.

The classification report for the unigram model with a score threshold of 0.45 and a forced minimum of one label returned true is displayed above in Table 2. The unigram model still performed the weakest with Statistics, but a precision and recall of 0.74 and 0.83 are not too bad given the class imbalance. As seen in the confusion matrix in Figure 10, the model tended to label more articles as Computer Science than was true. That is not too surprising given most of the articles were indeed Computer Science. Glancing over some problem cases revealed that some of the excessive labeling of Computer Science may also be because terms relating to software, technology, and computers tend to show up in all kinds of articles. Perhaps the tools and approaches may have been mentioned in the abstracts. Also, with recalling the word clouds from the exploratory data analysis section, Computer Science and Statistics seemed to have a lot of overlapping words.

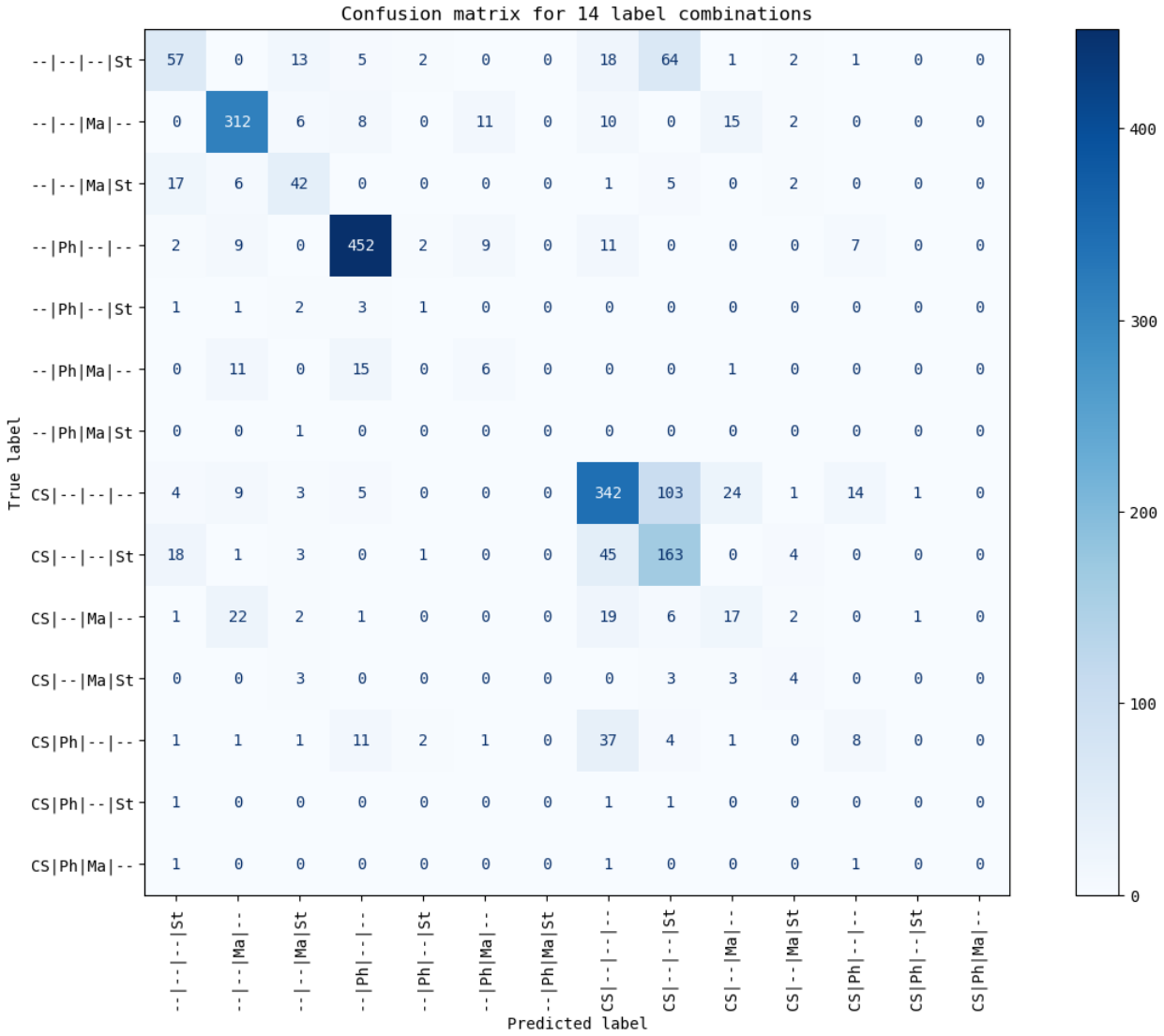


Figure 10. Confusion matrix for the fourteen different combinations of categories.

C. Comparisons and Evaluations

In terms of the preprocessing steps, scikit-learn's TF-IDF vectorizer was very quick, taking only about five seconds to preprocess the documents, whereas spaCY's pipeline took three to four minutes. However, spaCY's pipeline was more sophisticated in that it also considered word meanings and usages, though term frequency and document infrequency were not considered at this point of the process, but in general, more work or different features can be extracted with spaCY. Neither of these methods was as slow as using NLTK and genism, which took 10 to 15 minutes to preprocess the text.

The scikit-learn classifiers and spaCY models attempted to solve different problems, but even so, the spaCY models performed better than the NB classifiers. The NB classifiers had especially poor recall when trying to predict Statistics, whereas the spaCY unigram bag-of-words model had higher recall for Statistics. The four-class ComplementNB had precision, recall, and an F1-

score of 0.84, 0.39, and 0.53 for Statistics, whereas the unigram bag-of-words model had 0.74, 0.83, and 0.78 for Statistics. The scores for the ComplementNB classifier were better for the larger classes, but it could be easier to correctly classify for single labels versus multiple labels because there are not as many things that need to be predicted correctly; the correct prediction for a single label problem is a true value for exactly one category per article, whereas the correct prediction for the multilabel problem can contain a true value for any one to three categories per article. In other words, a single label guess would have a 1 in 4 chance to randomly pick the correct answer, but a multilabel guess would have a 1 in 16 chance to randomly pick the correct answer, and two of those will never be the correct answer. In terms of the time taken to train, training the Naive Bayes classifiers was very fast, whereas training any of the spaCY models took longer.

Overall, the unigram bag-of-words model would be the best model for this problem, especially since it does address the multilabel aspect of the problem. Its accuracy was good, and it can attain decent precision and recall across all categories. The threshold to use may depend on the application because different use cases may prefer more precision or more recall, but using a threshold around 0.45 to 0.50 will land precision and recall across all categories in the range of 0.75 to 0.93. It may be better to force the interpreter to return at least one category if no category met the threshold because every article needs at least one label anyway.

5. Conclusion and Future Work

Text categorization can facilitate organizing and selecting relevant information, which can help people and software find information more quickly and conveniently, so it would be useful to classify new text accurately and automatically. The unigram bag-of-word model can be used to tag new articles that may fit in the Computer Science, Physics, Mathematics, and Statistics categories to acceptably high accuracy, though the potential for overrepresentation of Computer Science and underrepresentation of Statistics should be kept in mind. There were two main issues encountered with the model in its current state, so amendments or alternative approaches can be considered in the future.

From the beginning, it was known that there was category imbalance within this dataset. Three of the six categories were represented decently, but the three underrepresented categories were not given any special treatment, and two of these categories, Quantitative Biology and Quantitative Finance, were dropped altogether. Although Statistics had the smallest presence in the dataset, the final model was not too bad at predicting Statistics, so further attempts to balance the classes were not pursued in this project.

A common approach to balance the classes would be to try resampling the dataset. A more involved approach that expands the project in a different direction could be to consider using different categories. The four categories were broad, and it seemed like Data Science topics appeared often in both Computer Science and Statistics, which was seen both during the EDA section and when looking at samples that were mislabeled by the model. Perhaps this project could be turned into an unsupervised and supervised problem: find similar groups, determine what these categories probably are or should be, and then try to classify them based on the new

categories—it would even make use of the other 9000 articles from "test.csv." One advantage of changing these broad categories into more granular categories could be that it may help people find what they are looking for faster. However, a question comes to mind, is the class imbalance that bad of a problem? Could it depend on the application? If a paper involves some Computer Science technology, is it bad to label it with Computer Science? Could this help people find or consider different applications for their problem domain? Could this help people learn the demand and usages for their tools to help give fresh ideas on what people may need or want out of the tools? Does this encourage exploration of articles that seem unrelated yet may be related by methods? Overall, the question is whether excessive labels can be useful somehow and therefore valuable.

The other issue was that the model sometimes would not predict a high enough score for any category to be determined true given any reasonable threshold (for example, if all the predicted values were less than 0.30). There were two simple solutions implemented in this project, which were to reduce the threshold slightly and to force the model to always predict at least one category as true, where the one category to be set true was the category that had the highest score. Lowering the overall threshold too much would cost the model precision, and simply returning the category corresponding to the max score would mean the prediction never returns more than one category even though our dataset contains articles possibly labeled with multiple categories, so maybe implementing a more sophisticated scorer or scoring with a different approach could be considered.

A more sophisticated scorer may be one that dynamically interprets the scores depending on the distribution of the current scores. For example, if the scores were [0.18, 0.16, 4e-6, 0.03], it would return [1, 1, 0, 0], instead of [0, 0, 0, 0] given a threshold of 0.5, or [1, 0, 0, 0] given the same threshold and a requirement that the maximum category is set to true if no category met the threshold. This would help overcome the issue of an uncertain batch of prediction scores never labeling an article with more than one category, and it may also help alleviate when scores are almost meeting the threshold but failing to return another category as true despite also being relatively highly recommended. For example, [0.51, 0.49, 0.01, 0.01] with a threshold of 0.5 would return [1, 0, 0, 0], even though the predictions were close and not very strong. Otherwise, perhaps scoring could be remade to be less of a classification problem with binary results, in which categories are each only true or false, and more of a recommendation problem, in which the model's predictions are treated more as a ranking of category by relevance, though how useful this is may depend on the application.

References

[1] B. Densil, “Topic Modeling for Research Articles,” *Kaggle*. 2020. <https://www.kaggle.com/datasets/blessondensil294/topic-modeling-for-research-articles>

Appendices

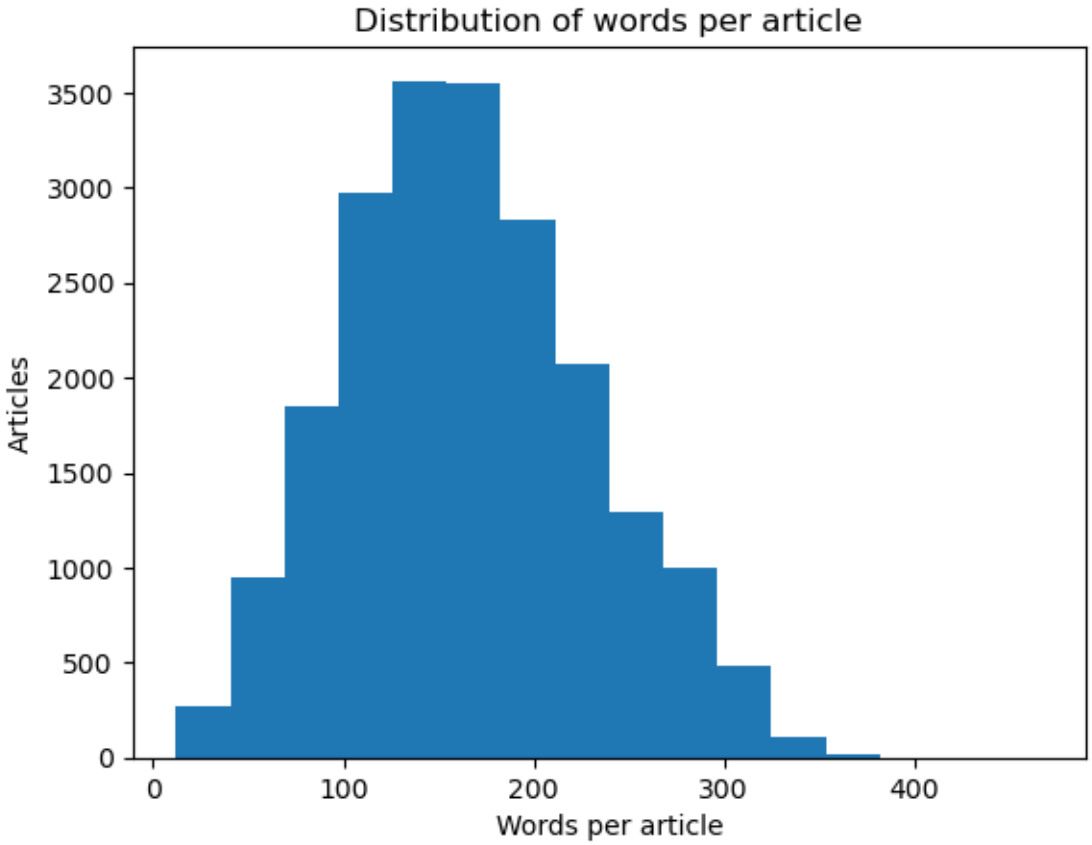
Appendix A

Article Dataset from “train.csv”

Column	Data Type	Description	In test?
ID	int64	Unique number ID	Yes
TITLE	object	Title of the article	Yes
ABSTRACT	object	Abstract of the article	Yes
Computer Science	int64	Binary number indicating if the article belongs to this category. 0 if false, and 1 if true.	No
Physics	int64	Binary number indicating if the article belongs to this category. 0 if false, and 1 if true.	No
Mathematics	int64	Binary number indicating if the article belongs to this category. 0 if false, and 1 if true.	No
Statistics	int64	Binary number indicating if the article belongs to this category. 0 if false, and 1 if true.	No
Quantitative Biology	int64	Binary number indicating if the article belongs to this category. 0 if false, and 1 if true.	No
Quantitative Finance	int64	Binary number indicating if the article belongs to this category. 0 if false, and 1 if true.	No

Appendix B

Words per Sample

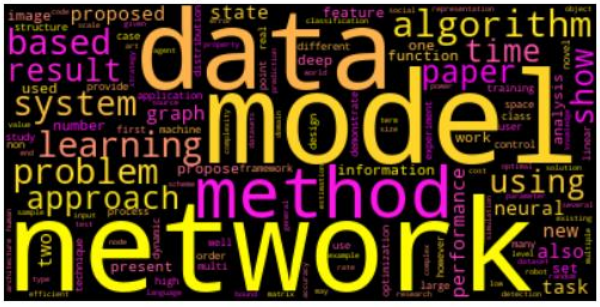


Appendix C

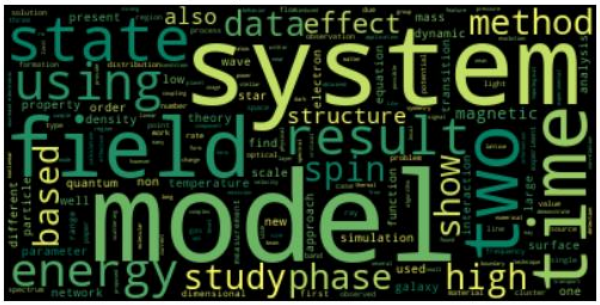
Word Clouds per Topic (200 words)

Unfiltered

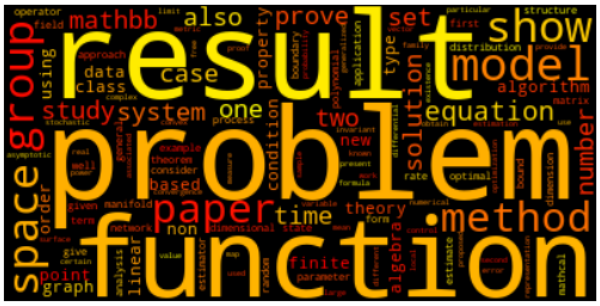
Computer Science word cloud



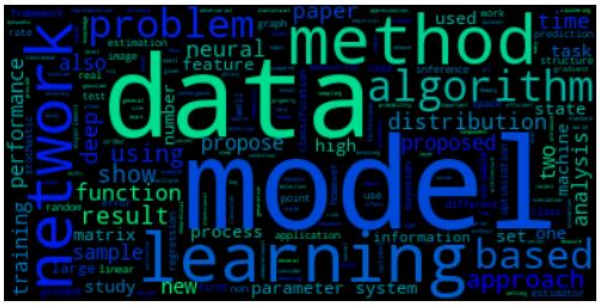
Physics word cloud



Mathematics word cloud

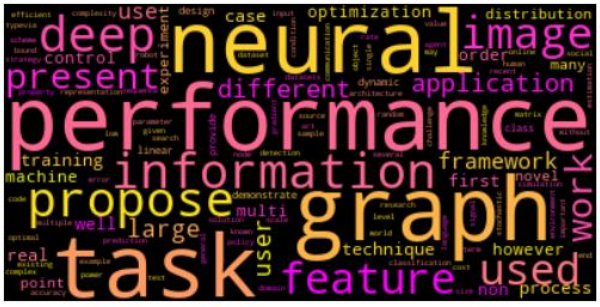


Statistics word cloud

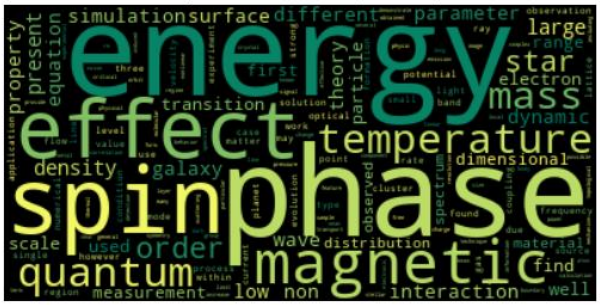


Filtered

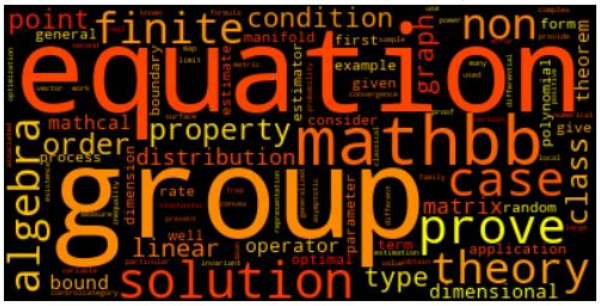
Computer Science word cloud (filtered)



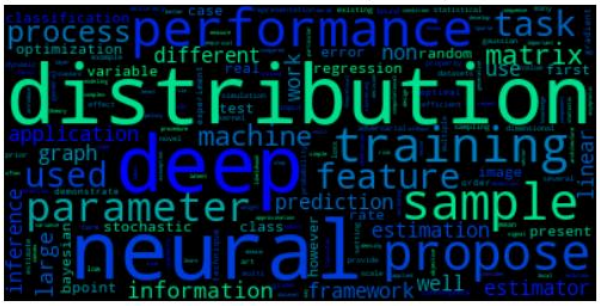
Physics word cloud (filtered)

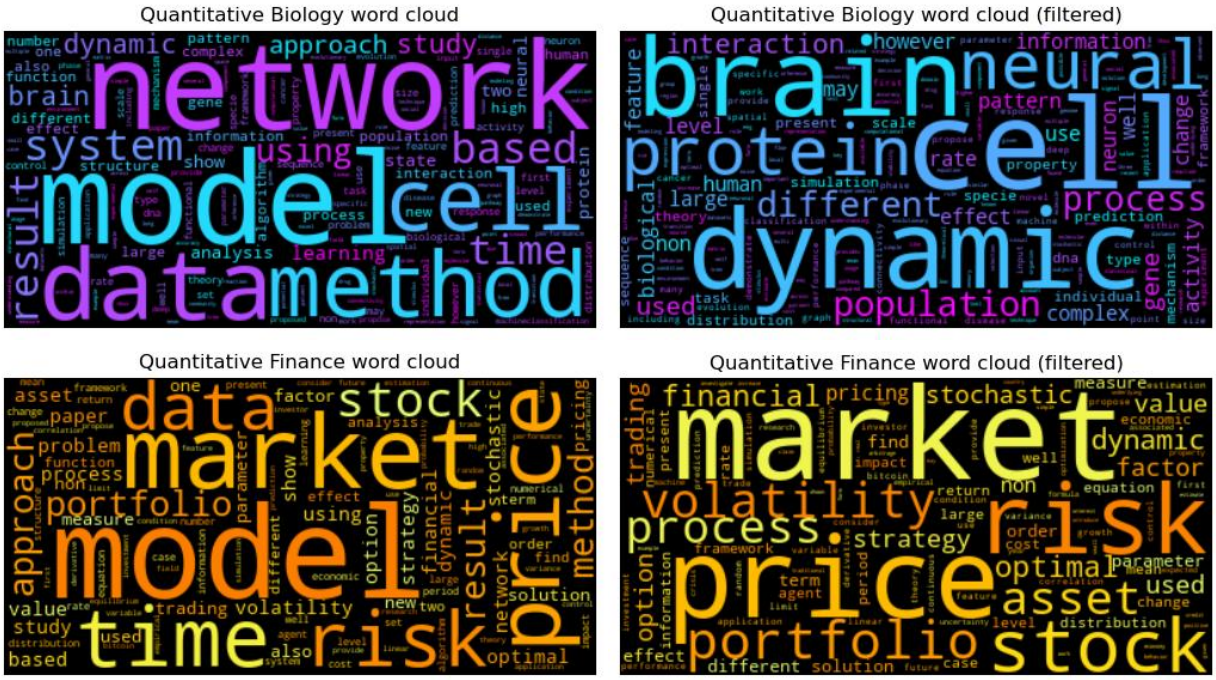


Mathematics word cloud (filtered)

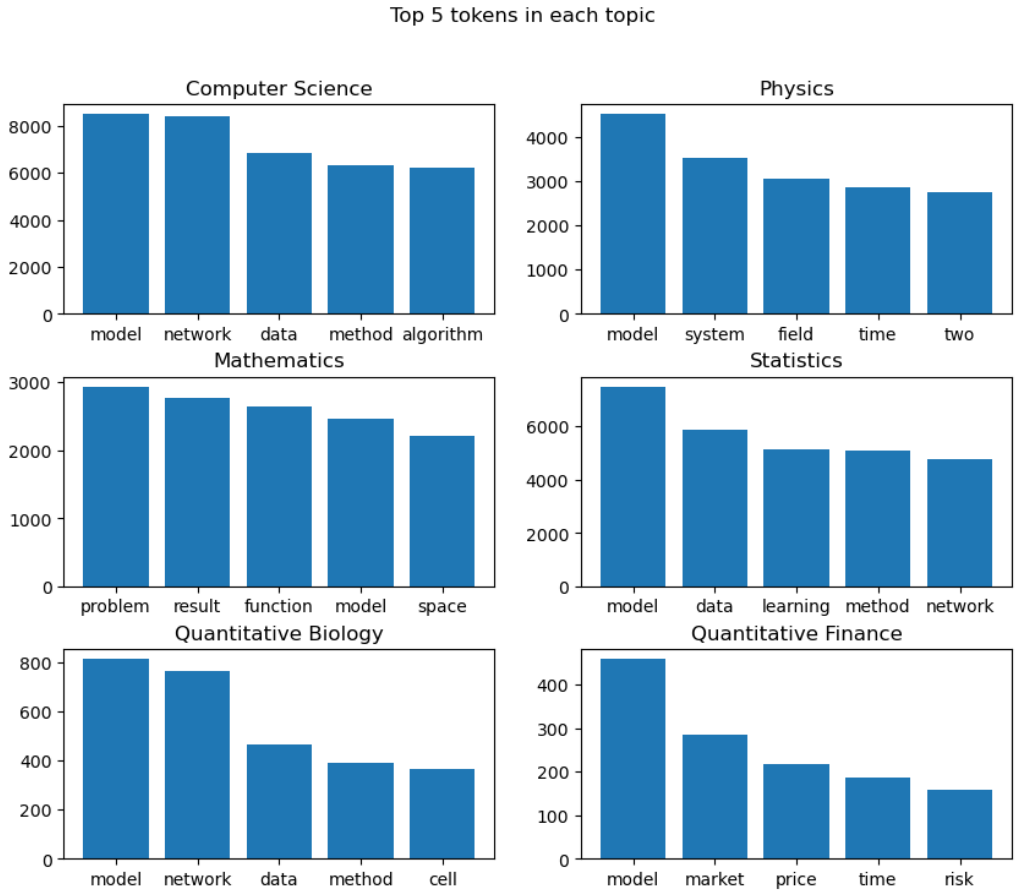


Statistics word cloud (filtered)

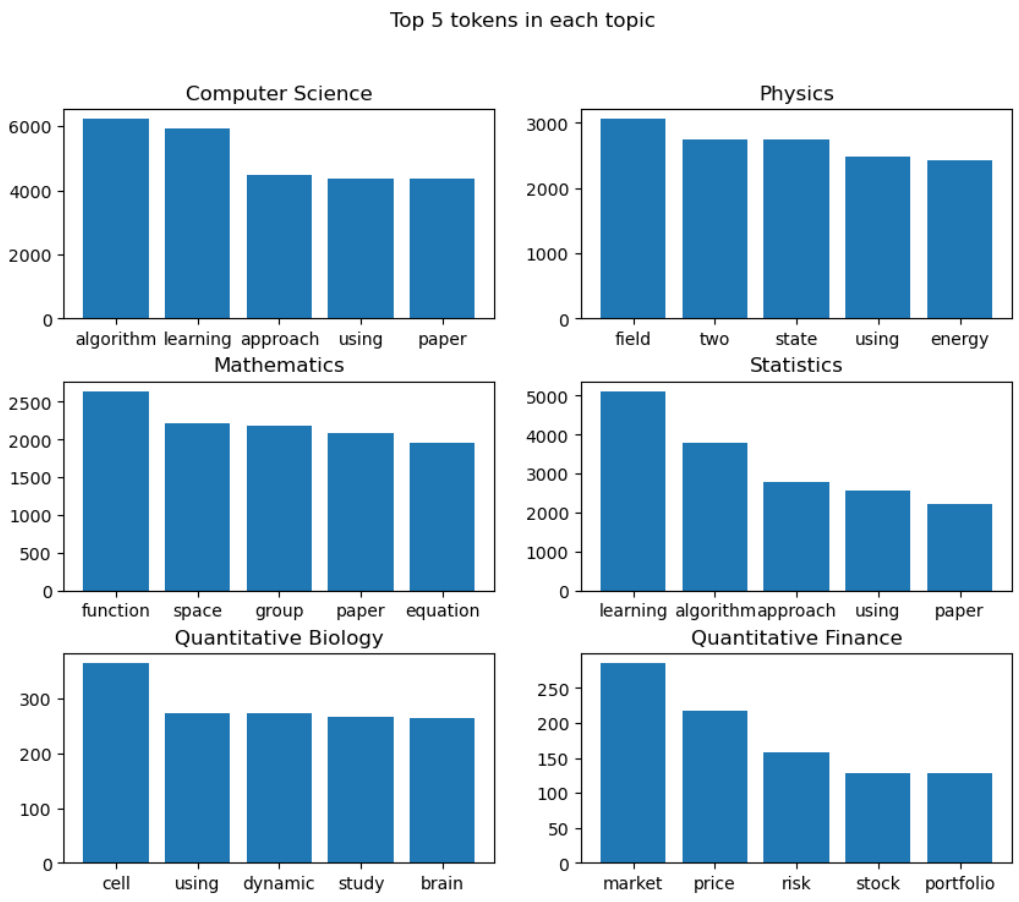




Unfiltered Top 5 Tokens per Topic



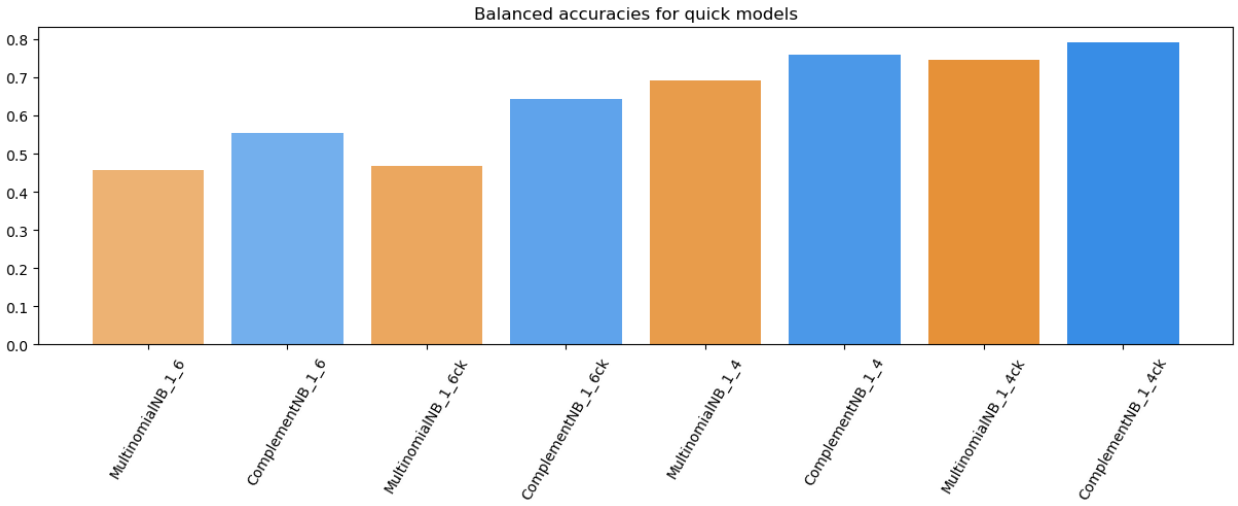
Filtered Top 5 Tokens per Topic



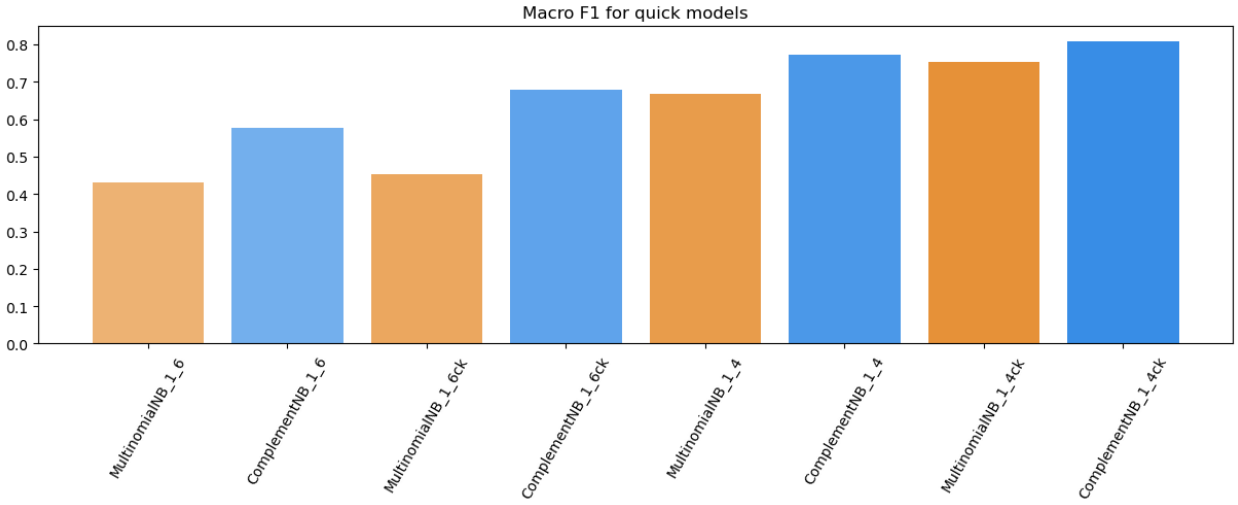
Appendix D

TF-IDF Vectorizer and Naïve Bayes Classifiers for the Single-Label Multiclass Problem

Balanced Accuracies



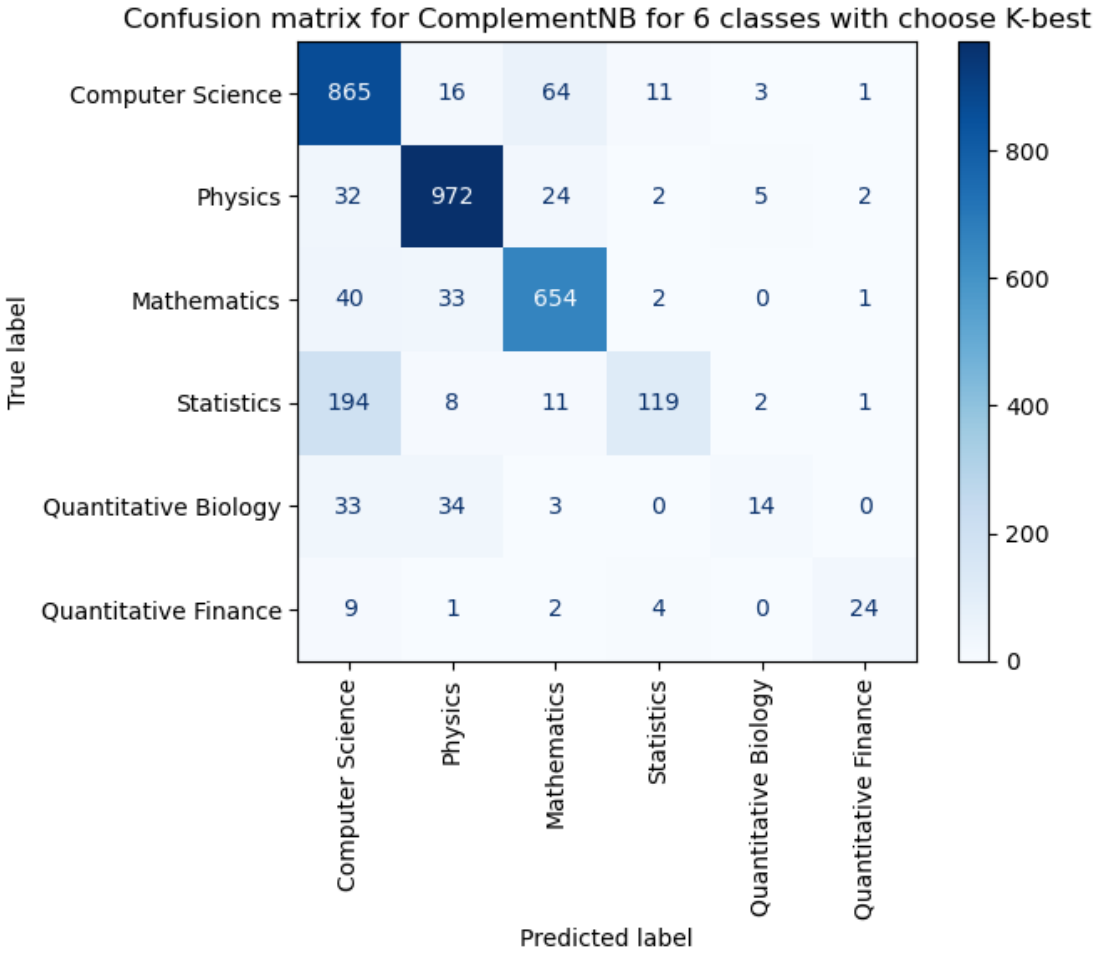
Macro F1



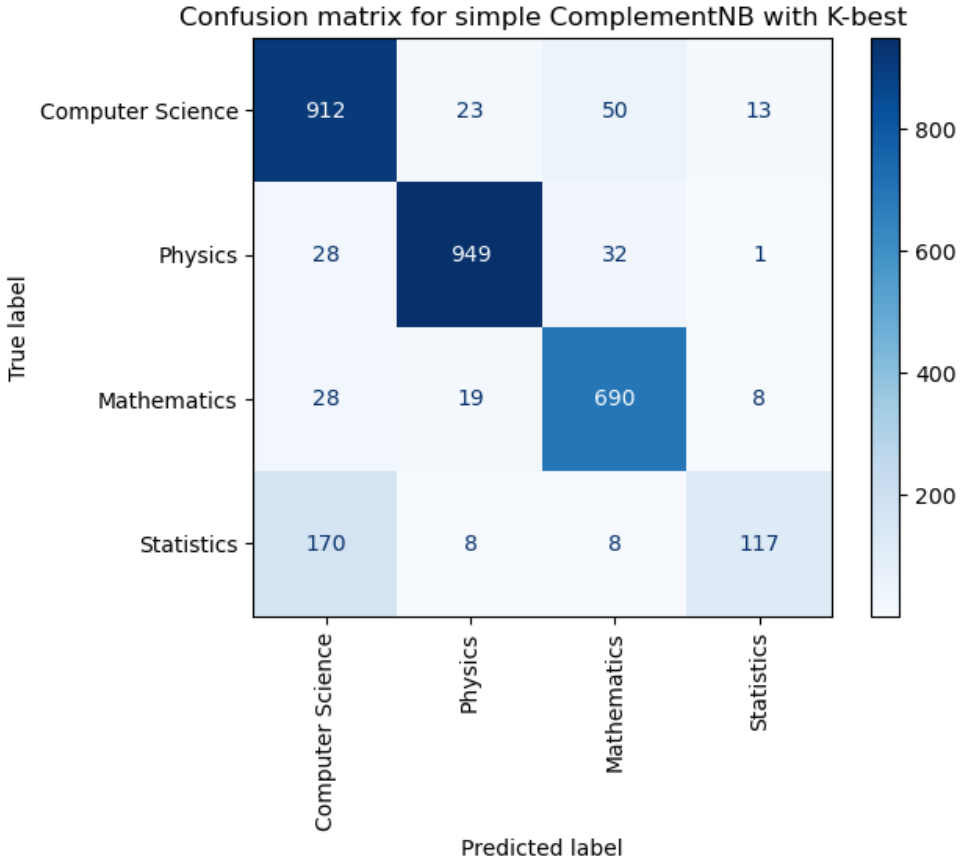
Precision and Recall for ComplementNB on six and four classes with select k-best

	Six Classes		Four Classes	
	Precision	**Recall**	**Precision**	**Recall**
Computer Science	0.74	0.90	**0.80**	**0.91**
Physics	0.91	0.94	**0.95**	**0.94**
Mathematics	0.86	0.90	**0.88**	**0.93**
Statistics	**0.86**	0.36	0.84	**0.39**
Quantitative Biology	0.58	0.17	n/a	n/a
Quantitative Finance	0.83	0.60	n/a	n/a

Confusion matrix for ComplementNB for six classes with select k-best



Confusion matrix for ComplementNB for four classes with select k-best



ComplementNB performed better than MultinomialNB across the board, and the final model, ComplementNB_1_4ck (ComplementNB on four classes with SelectKBest) did the best. The last two models both also used a slightly different TF-IDF vectorizer.

Vectorizer	**strip_accents**	**lowercase**	**stop_words**	**max_df**	**min_df**	**ngram_range**
Version 1	unicode	True	english	1.0	2	(1,2)
Version 2	unicode	True	english	0.2	5	(1,2)

Appendix E

Three spaCY Models' Evaluations with Scorer Threshold=0.5

	Unigram BoW			Bigram BoW			CNN		
	P	R	F	P	R	F	P	R	F
Computer Science	86.73	87.31	**87.01**	83.99	**88.20**	86.04	**87.76**	81.40	84.46
Physics	**92.86**	83.53	**87.94**	92.52	**83.53**	87.79	88.75	84.51	86.58
Mathematics	85.93	**82.08**	**83.96**	86.23	80.82	83.44	**87.85**	79.03	83.21
Statistics	**75.72**	79.23	**77.44**	73.14	77.42	75.22	66.15	**85.89**	74.74