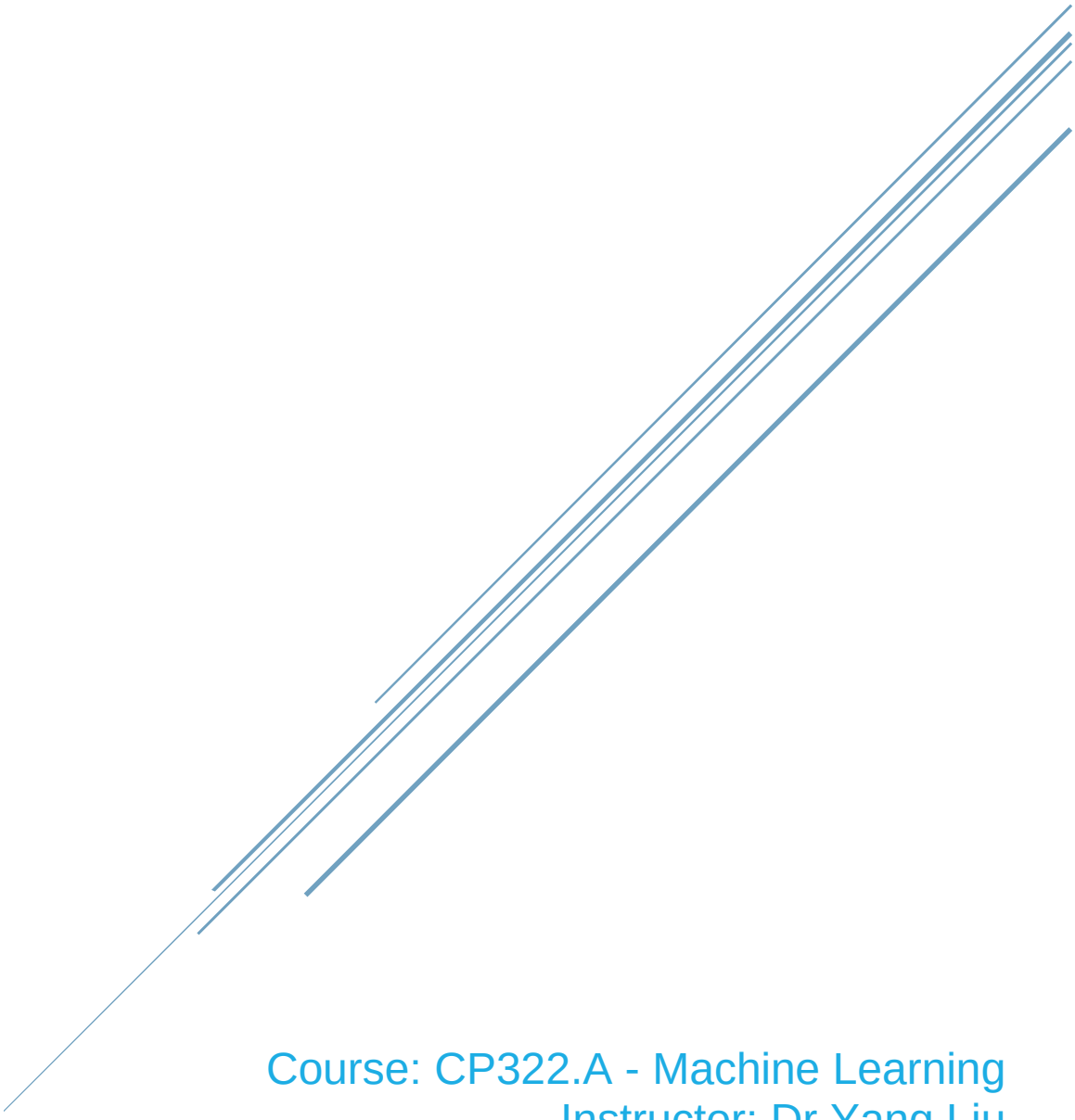
FINAL PROJECT REPORT

Authors:

AMARDEEP SARANG - 160112080

DANIEL BEREZOVSKI – 160173040



Course: CP322.A - Machine Learning
Instructor: Dr Yang Liu
Institution: Wilfrid Laurier University

Introduction

Journalism is important. It is what keeps people informed, aware, and excited about politics, business, food, technology, and many more topics. And while millions of people love reading journalists' work, many don't have the time to browse through dozens of articles to find one or two that are on a subject matter relevant to them. Thus the problem of filtering by subject or document identification is created. Nowadays people expect extremely personal, and tailored recommendations for content and the first step in the process of tuning articles to readers personal interests is being able to dynamically interpret the subject of the articles they read… whether it's for a basic system that filters articles based on subject matter, for a slightly more advanced search function, or as previously mentioned, a first step to a completely personalized article recommendation system. To solve these problems with machine learning, we explored 2 different algorithms that we suspected will perform this task well, with an amplitude of preprocessing techniques. This allowed us to use a dataset of 40 articles (10 from each category listed above), to build extremely accurate models to classify articles into one of the aforementioned classes. Our goal was to determine which preprocessing technique, combined with which algorithm would give us the best results, and pave the way for a larger investigation with more data, and more classes.

Investigation procedure

Dataset collection

To explore the topic of document identification we required a collection of articles that all pertained to a small number of topics, where each document contained some word that would help identify which topic it was from. In addition, for model training and evaluation purposes we required a way to identify which topic each article belonged to. Through our research, we were unable to find a dataset that met these characteristics. So we decided to build our own dataset by collecting articles from lifestyle and online news websites. All the articles collected were from one of four topics: technology, politics, business, and food. Ten articles from each topic were collected for a total of forty articles in the dataset. Each article was placed in its own txt file and the first row of each file would contain a single digit that would identify the topic of the article according to the following key: technology=1, politics=2, business=3, food=4.

Once we had our raw dataset we could begin our investigation. During our investigation we would be exploring two machine learning algorithms, k-nearest neighbors and Naïve Bayes, with three different stop word lists, random sampling, and TF-IDF values for each word. Stop word list 1 would have 100 common words which were to be removed, list 2 would have 1000 and list 3 would have 3000.

Procedure

The investigation procedure consisted of two main parts; preprocessing which entailed converting the collection of articles into a feature set usable by the machine learning algorithms and implementing the machine learning algorithms to evaluate their performance. The following preprocessing procedure was followed:

1. The article would be parsed into an array (list) of words.

2. The words for each article would be added to a “feature set word list” which will be referred to as the feature list it would only hold one instance of every word with no repeats.
3. Steps 1-2 would be repeated for all articles in the collection. So by the end, the feature list would contain all words from every article.
4. A stop word list would be applied to the feature list. If a word was in the stop word list as well as the feature list it would be removed from the feature list. One of three different stop word lists would have been applied. It is also an option the skip this step and not apply the stop words.
5. N number of words from the feature set would be randomly selected to make the feature set smaller. This step is optional.
6. The file would be scanned and a count of how many times each word from the feature list appears in the article would be recorded
7. For each word in the feature list the term frequency: $Tf = \frac{w_d}{N_d}$ where: w_d = the number of times that word appears in document d, N_d=the total number of words in document d. This ratio is then printed to a txt file.
8. As an alternative to step 7 the TF-IDF may be calculated instead of term frequency. For each word in the feature list TF-IDF=$\frac{w_d}{N_d} * \log_e\left(\frac{T}{D_w}\right)$ where: w_d = the number of times that word appears in document d, N_d=the total number of words in document d, T= the total number of documents in the set, D_w= the number of documents in the set that contain the word. This ratio is then printed to a txt file.
9. Once step 7 or 8 is repeated for each word in the feature list, the last number printed on the line will be the topic digit for the article.
10. Steps 6-9 would be repeated for each article.

Table 1: Datasets, their names, number of features, and the preprocessing techniques used to create them

File (dataset) name	# of feature	Preprocessing used
dataset_1.txt	7612	no stop words term frequency ratio used
dataset_2.txt	7248	removed stopwords using stopwords_1.txt, term frequency ratio used
dataset_3.txt	6704	removed stopwords using stopwords_2.txt, term frequency ratio used
dataset_4.txt	5546	removed stopwords using stopwords_3.txt, term frequency ratio used
dataset_5.txt	1000	removed stopwords using stopwords_3.txt + randomly select 1000 features, term frequency ratio used
dataset_6.txt	7612	TF-IDF ratio used
dataset_7.txt	7248	removed stopwords using stopwords_1.txt, TF-IDF ratio used
dataset_8.txt	6704	removed stopwords using stopwords_2.txt, TF-IDF ratio used
dataset_9.txt	5546	removed stopwords using stopwords_3.txt, TF-IDF ratio used
dataset_10.txt	1000	removed stopwords using stopwords_3.txt + randomly select 1000 features, TF-IDF ratio used

After completing this procedure we can see that a dataset file will be created with each row containing a ratio entry (TF or TF-IDF) for each word in the feature list, which will be comma delaminated plus the topic digit at the end of the row. Each row will correspond to a new article. As can be seen in the preprocessing procedure several preprocessing steps are optional and interchangeable, different combinations of preprocessing techniques were combined to make 10 different datasets. The combination of preprocessing techniques used to make each dataset can be seen in table 1. Each of these datasets would be used with each machine learning algorithm where they would be split into test and training sets which would be used the train and test the algorithms.

Algorithms and individual performance

K-nearest neighbors

K-nearest neighbors is a similarity-based algorithm. It calculates the Euclidean distance from the target input to each available training article, and makes a classification decision based on the k nearest training articles and a majority voting policy. K-nearest neighbors was chosen for this investigation because it is primarily designed to handle continuous descriptive features and discrete target features, this made it an easy choice when working with term frequency or TF-IDF ratios and 4 discrete topics as the target outputs. In addition, K-nearest neighbors is also fairly robust to the curse of dimensionality which was needed since this problem involved working with large feature sets as can be seen in table 1.

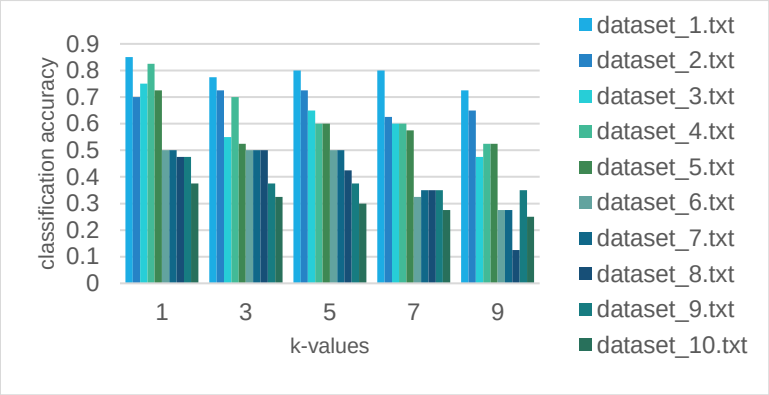
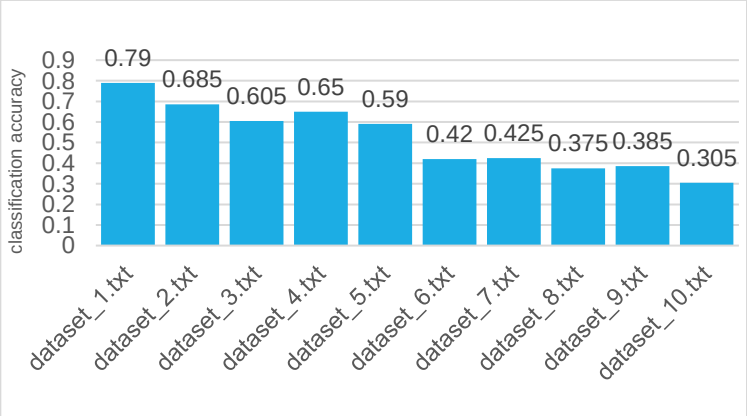
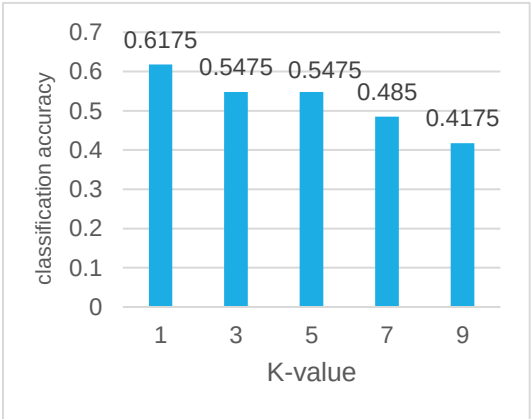


Figure 1a (above): *shows the classification accuracy from each combination of dataset and k value.*

Figure 1b (below left): *shows the classification accuracy from each dataset while averaging the results from each k value.*

Figure 1c (below right): *shows the classification accuracy from each k-value while averaging the results from each dataset.*





During this investigation, we explored what effect preprocessing had on the dataset and which value of k would give the best performance.

As can be seen from figure 1a the results from the K-nearest neighbors varied greatly from a classification rate of 85% down to 25% depending on the k value and the type of preprocessing that had been used to create the dataset. From figures 1a to 1c, the effects of preprocessing become apparent, however perhaps even counter-intuitive, as the algorithm performed the best with dataset 1 (79%); which had no preprocessing. This is further reinforced by the fact that 3 of the top 4 results use dataset 1. Dataset 10 consistently gave the worst performance with every k value, thus only getting an average classification rate of 30.5%. This was despite dataset 10 having the most preprocessing in the form of using the largest stop word list, having random selection and using the more complicated TF-IDF ratio. In addition, we can see the effect the k value had on the performance in figure 2b. When used with all datasets, k=1 had the best overall performance with an

average classification rate of 61.75%. The lowest performance was when k=9 with an average classification rate of 41.75%. From the five k values tested k=1,3,5,7,9 we can see that the relation between the k values is roughly linear with a gentle negative slope, that is to say, as the k value increases the performance generally decreases. Taking both of these observations into account, it is easy to see that the best performing instance of K-nearest neighbors was using dataset 1, with k=1, achieving an accuracy rate of 85%. The second-best performance was using dataset 4, with k=1, achieving an accuracy rate of 82.5%. There was a tie for third place since using dataset 1 and k=5 or k=7 both had an accuracy rate of 80%.

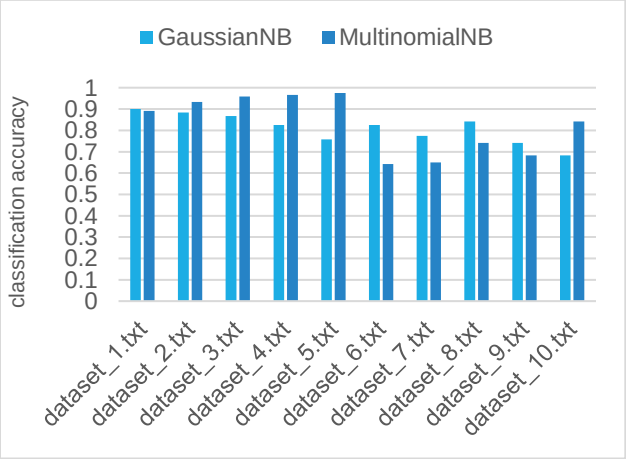


Figure 1a (above): *shows the classification accuracy from each dataset using both GaussianNB and MultinomialNB*

Naïve Bayes

The Naïve Bayes machine learning algorithm was a very natural choice for this task. Historically, this probabilistic model has been often used for language related classification tasks. An age-old example that is always brought up when talking about the Naïve Bayes algorithm is spam email detection, or in other words, classifying emails as spam or not spam. Our problem is just an extension of the spam problem, with the main differences being generally longer text sequences and more classes. Naturally, since Naïve Bayes was such a popular and powerful choice for spam detection, we decided to apply it to our problem as well. After implementing our solution, we found that all models stayed between a 55% and 97.5% accuracy range, with the most consistent and accurate model being the Multinomial Naïve Bayes algorithm, with preprocessing technique #5, which in our case was removing stop words using the 3rd set, and selecting only 1000 features to use, out of the ~7550 that we had. In our case each feature was a word, essentially meaning we selected only 1000 of the most impactful words out of the ~7550 available to us. This model produced an accuracy of approximately 97.5% over many iterations. It is good to note here that even though performance does seem relatively strong in this case, we only had 10 examples from each dataset, and out of those 4 randomly selected as testing features. The worst accuracy rate of ~55% was obtained with preprocessing technique #7, where we construct a TF-IDF table and use the first set of stop words.

Conclusion

Preprocessing results

When we began this investigation one of our major goals was to determine which preprocessing techniques or combination of techniques gave the best overall performance. However, as the data above shows, this is not a straightforward question and depends greatly on the algorithm used. For example, when looking at Naïve Bayes it seems that using more preprocessing in the form of a

larger stop word list and random selection, made the algorithm perform better. On the contrary, K-nearest neighbors performed better when little or no preprocessing was used. The one pattern that did emerge when using both algorithms is that they both performed better on datasets 1-5, which used the simpler term frequency ratio, than they did with datasets 6-10 which used the more complicated TF-IDF ratio. Thus, one could conclude, at least in the context of this investigation, that using term frequency gives a better performance than TF-IDF.

Algorithm results

The second goal of our investigation was to find which algorithm among the two had the best overall performance. Looking at the data it is evident the Naïve Bayes did, in fact, perform better then K-nearest neighbors. The best classification rate of Naïve Bayes was 97.5% which was significantly better than K-nearest neighbors best at 85%. Furthermore, Naïve Bayes' top four results were all in the low to high 90s whereas K-nearest neighbors' top four results were in the low-mid 80s. Even when considering the wide range of results with both algorithms Naïve Bayes had a better overall average classification rate at 81.91% when compared to the K-nearest neighbors average classification rate of 52.3%.

Overall results and evaluation

The final goal of this investigation was to determine which, if any, combination of machine learning algorithms and preprocessing techniques performed well enough to solve the problem of document classification. The best performing combination was that of the Multinomial Naïve Bayes algorithm, with preprocessing technique #5 which had a classification accuracy rate of 97.5% which could be considered a fairly strong result given the fact that it was trained on a limited dataset with only 40 articles. Similarly, the next four strongest results also used the Multinomial Naïve Bayes algorithm and they all had strong classification rates which were higher than 90%. Taking this into consideration, as well as the limited scope of this investigation it can be concluded that the Multinomial Naïve Bayes algorithm can solve the problem of document identification when the correct preprocessing is used. Contrary to Naïve Bayes, K-nearest neighbors may be considered a somewhat weak classifier, but in its best performance, it had a classification rate of 85% which is still a significant improvement over the 25% classification rate of a random classifier. In summary, Multinomial Naïve Bayes can be recommended as a strong solution to the problem, but K-nearest neighbors cannot be recommended at all.

Comparing with other papers

When comparing our investigation to those found in the research papers of Thorsten Joachims from the University of Dortmund, Germany and of a research team from Amrita University, India, we find that both of the algorithms performed better in our investigation. It is hard to say why this is since our investigation used a different dataset and preprocessing techniques, making it hard to induce a direct comparison. The difference is likely attributed to differences in the datasets.

Future research

Although the results from above seem promising, this investigation was limited in scale due to time and complexity constraints. One of the main limitations was the relatively small dataset which only consisted of 40 articles with 10 from each topic. If this topic was investigated further, the next investigation should strongly consider expanding the dataset substantially. Future investigations should also include more complex algorithms such as Support vector machines and neural

networks. In addition, this investigation could not explore all variations of algorithms used such as the use of Manhattan and Minkowski distance in K-nearest neighbors, this should be investigated further.

References

"1000 MOST COMMON WORDS IN ENGLISH." *Education First*, www.ef.com/english-resources/english-vocabulary/top-1000-word. The source for stop word list 2

"3000 MOST COMMON WORDS IN ENGLISH." *Education First*, www.ef.com/english-resources/english-vocabulary/top-3000-words/. The source for stop word list 3

Goble, Clark, and Art Pollard. "Onix Text Retrieval Toolkit API Reference." *Stopword List 1*, www.lextek.com/manuals/onix/stopwords1.html. The source for stop word list 1

Joachims, Thorsten. "Text Categorization with Support Vector Machines: Learning with Many Relevant Features." *Machine Learning: ECML-98 Lecture Notes in Computer Science*, 1998, pp. 137–142., doi:10.1007/bfb0026683.

Kumar, S, and Vairaprakash Gurusamy. "Preprocessing Techniques for Text Mining."

Rajasundari, T, et al. "Performance Analysis of Topic Modeling Algorithms for News Articles." *ResearchGate*, www.researchgate.net/publication/320431243_Performance_analysis_of_topic_modeling_algorithms_for_news_articles.