KDD Assignment 2

Rogério Pontes

University of Minho,
rapontes@inesctec.pt

Abstract. Data is being generated at a faster pace than ever before and the existing tools used to extract relevant information must be accurate and efficient. The data generated are stored on textual documents that contain information to be latter retrieved trough search queries. Some of these documents can have field specific information such as technical patents or biomedical terms. These documents make it harder for simple indexing engines to accurately answer a query. To process and retrieve relevant documents according to the search terms a text mining process is required.
Text mining uses classification algorithms to assign class labels to text documents. This process trains a classifier which is able to latter retrieve documents given a search query. To achieve an effective classification, multiple steps have to be taken, and different classification algorithms evaluated to find the most appropriate to a given dataset.
In this report we present the outcome of classifying two categories of the Reuters dataset. These solution proposed takes into consideration the format of the documents on the dataset and does multiple evaluation of four different classification algorithms on different scenarios. From our evaluation we found that a decision tree classifier is the best fit has it provides a perfect score in all of the metrics considered.

Keywords: Textual data mining, Knowledge discovery databases, Classification, Unstructured data, decission trees, k nearest neighbors, Naive Bayes, support vector classification, Reuters dataset

1 Introduction

In todays day in age, technology has leaped boundaries by connecting a vast amount of human population over the Internet. This global interconnection has lead to a growth of data that is immediately accessible. Data can be naively divided into meta-data generated by computers and textual documents composed by humans. The latter, is generated in the process of human interaction trough emails, publications in social media platforms, scientific publications or media outlets.

Most data generated by humans have information with the potential to be commercially valuable. The general opinion of a product can affect its demand, and an understanding of the reception of a product can help a business choose the best option to increase its revenue. Aside from the commercial aspect, there

is also a need to sort and search textual documents in an efficient manner. One common example of this need comes from a simple query search on a search engine to a more demanding application. A demanding field is the search for scientific documents. To extract relevant information from a large collection of documents it becomes unreliable to use a manual method as it not only prone to errors but also incapable of providing information in a timely manner. Furthermore, the existing methods of information access and search engines indexing process are only able to find documents that are directly related with a given query but are not able to extract additional information [1]. An additional problem of searching textual documents it is their lack of structure and the use of a natural language. These two factors increase the difficulty of having an automatic process, capable of extracting meaning from textual publications, emails, messages, comments and reviews of products.

The analyses of textual data by automatic processes of a machine is known as textual data mining, first introduced by *Feldman and Dagan* [2]. While multiple definitions have been put forwards to define what consists the process of textual data mining, we are considering data mining as the process of applying classification algorithms and statistical analyses to a collection of individual document. The objective of this process is to extract relevant information in order to aggregate and classify documents.

The main goal of document classification is to search for a relevant document in a large collection of documents given a search query. Closely related to document classification is document clustering, that aims to group together documents that are closely related. Both of these techniques can be applied to a search engine application to provide its users a better experience and faster access to relevant documents according to their search terms. Sentimental analyses, on the other hand, has the goal to identify the general opinion of a users in relation to a topic. Consider an online shopping application that sells multiple products, where the users of this application can write reviews of any product. Given a product with multiple reviews, sentiment analysis can be useful to understand if the product has a general positive or negative opinion from the users.

The objective of this report, is to describe the steps taken to create a text mining application capable of classifying small finite dataset. The work presented is just a sample of the engineering process that is undertaken by production ready textual data mining systems such as Mathematica [3] or Semantic Scholar [4]. As such, the report describes the application of different classification algorithms in textual data mining to a well known dataset, Reuters dataset. The following four classification algorithms will be used : support vector machines, Bayes classifier, k-Nearest neighbor and decision trees. From these classifiers only the best will be selected according to following metrics, such as the precision, recall and the harmonic mean. The source code used for the assignment is available in the following link `https://github.com/roger62/classify-text`.

1.1 Outline

In section 2 common classifiers used for text mining are introduced, and a brief comparison between them is provided. The following section, section 3 introduces the Reuters dataset and explains the relevant details that will influence the classification process. On section 4, the steps taken by a text mining application and the system architecture used to classify the dataset is explained. After the system architecture, section 5 presents the results of classifying the Reuters dataset with the four different classifiers. The final section 6 concludes the report, by summarizing the results and presenting future work.

2 Background

The existing work on textual data mining aims to improve the accuracy of classification algorithms when working with textual data. Traditionally, these methods work on data mining by deriving relationships from structured data. One crucial step for text mining and data mining is the feature selection. In both cases the data processed by the classification algorithm consists of features that must be filtered and preprocessed in order to have better results. The work of *George Forman* [5] evaluates the impact of twelve feature selection methods over a range of different datasets, including the Reuters dataset. However this study is limited to a single classification algorithm, Support Vector Machines (SVM). SVM are considered a recent classification algorithm [6,7] that is capable of categorizing new datasets by defining hyperplanes from the training dataset. These algorithms stand out as their classification accuracy is independent from the number of existing features [7].

While SVM are considered state of the art in text classification algorithms, the Naive Bayes algorithms are often discarded as they are outperformed by other classifiers [8]. The naive version of the algorithm assumes that the features of the datasets are independent. Furthermore the algorithms rely on the Bayes' theorem that assumes that the probability of an event is dependent from previous observations. According to *Jasson Rennie et al.* [9], these algorithms have two main problems. First the Naive Bayes choices is strongly affected by the amount of training that exists for each class. Secondly, in text mining, the assumption that features are independent which is not a good assumption when words are related and are a strong indicator for a document classification.

Another simple machine learning algorithm is the k-Nearest Neighbors algorithm (k-NN). This algorithm works by finding for a given input, the closest points stored from a training dataset. The distance is calculated with the euclidean distance and the number of close points is parameterized by k. The majority of points that belong to same class determine the class of the given input [10]. One major problem with the k-NN algorithm is that it stores all of the training information. This is information is always processed for every query. This can be a major problem, when the training dataset is large, causing the classification process to be slow [10].

The last, of the most common classification algorithms are the Decision Trees algorithms. These algorithms assume that every feature has a domain of finite discrete values and there is a single target feature. In a decision tree, every node of the three is labeled with an input feature. The arcs from every node is one of the possible values of the domain. On the leafs of the tree, there is the classification label that depend on the feature values. This algorithms have numerous advantages, such as being simple to understand, have a good accuracy even with noisy data and do not require any assumptions about the data distribution [11]. Nonetheless, they have problems to learn certain concepts such as XOR and require additional algorithms to prevent the constructed tree from being over-fitted to the training dataset.

3 Reuters Dataset

The reuters dataset is a collection of 22 Documents that were published on the Reuters newswire in 1987. The documents have been cleaned, and duplicates were removed by different researchers over the following years to create a standard dataset. Such dataset, enables different classification algorithms to be compared across studies without being influenced by discrepancies. The original 22 files are in a Standard Generalized Markup Language (SGML) format, with the first 21 having 1000 independent documents each, and the last file having 578. Each document on a file is divided by a pair of *tags* with the name *REUTERS* that mark the beginning and ending of a document. In every case, these tags are the only content in a line. While these tags have additional information they are discarded as they are not relevant for the classification process. Inside these tags, there are additional tags that contain metadata about the document and tags that limit the actual body of the document. The metadata tags hold information such as the date of origin, the general *topics* of the documents, relevant places and people. From these tags, the most relevant are the topics, which are used as the *class* for the classification algorithms. There are several topics trough out the documents, and some documents can be part of different topics. The topics range from countries, money and food, but in this report we are restricting to just two topics, coffee and interest.

4 System architecture

Every text mining application follow a series of common steps in order to process and retrieve information [12]. These steps are composed to create a workflow for the data analyses task and range from processing the raw documents, organize the information in a structured format and classifying the results. As such, in this report we sort the several steps in three different groups and explain in detail each part. The system architecture will later be presented and describe how these steps have been implemented.

4.1 Components workflow

The first group is the preprocessing of a raw dataset and the creation of documents that can be used to create a bag of words. The group is depicted in Figure 1. This group starts with the information retrieval step. This step consists of extracting a set of documents, a corpus, that contains textual data that might be useful to extract information from it. As in this report, the reuters dataset is being used, this step consists of a simple download of a tar archive [13].

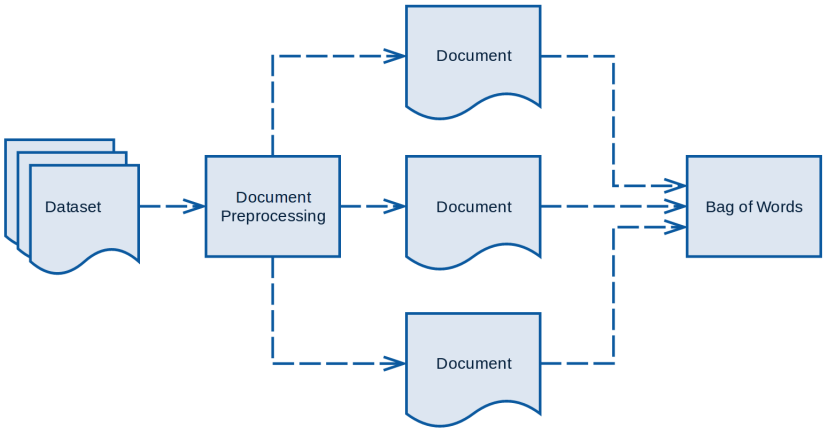

Fig. 1. Document preprocessing

Following the information retrieval, the documents need to go through some preprocessing in order to create a corpus of documents that can be used to create a document term matrix. As presented in Section 3, the Reuters original dataset has multiple documents in several files. These files must be processed to divide the documents into individual files and sort them in groups of topics. This requirement comes from the existing software applications that work with the assumption that each document is in a different folder and documents classified by one label are in the same folder. As such, in our architecture we have a component that goes through every file of the original Reuters dataset, divides each file in multiple documents and stores them in a new folder according to its topic. As a document can have multiple topics, the same document is duplicated and stored on the corresponding for folder for each topic.

The following step is the creation of a bag of words from every document. As in this report we are only considering a smaller subset of topics, the bag of words is created from documents with the topics of coffe and interest. A bag of words is a common data structure used in text data mining applications to store which words appear in each document and how many times they are encountered. With this goal, the bag of words tokenizes strings and gives it a unique id, counts the

number of occurrences in the document and normalizes the tokens in the overall corpus.

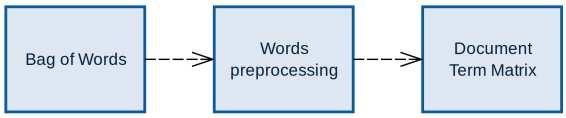

Fig. 2. Creation of Document Term Matrix

The bag of words has all the possible terms that exists on the corpus, however some of these terms may not be useful for the classification process. As such following the creation of the bag of words, the next step is often the preprocessing of the terms. The goal of this task is to clean the natural language text from ambiguities, irrelevant information and improve the process of extracting relevant information from the document. The normal tasks performed in this step are: removing punctuation, remove small words, remove stop words, convert to lower case and remove numbers. Additionally, taking into consideration the format of the original documents, it is possible to remove additional text, such as dates or formatting tags. Thus, the second group, depicted in Figure 2, consists on the steps of taking the bag of words, filtering unwanted terms and creating a document term matrix.

A document term matrix describes the frequency that the terms occur in a given document. In this matrix every row entry represents a single document of the corpus being analyzed and each column represents a term obtained from the corpus. Two types of representations for storing the frequency of the terms in the documents. A binary representation, simply counts if a term is present or not in a document. This simple representation can be used for document retrieval and identify which documents are similar through a proximity measure. Another possible representation is the term frequency–inverse document frequency (td-if). This scheme counts how many times a term shows up in a document and has the following two assumptions: a term with a high frequency might be more important than a term with low frequency; if a term is present in many documents it is probably less relevant than a term that is present in fewer documents. In our system, the selected representation for the matrix is the td-if as it is the most common technique to measure the relevance of a term [14].

The final group of steps is in Figure 3 and consists of training and evaluating the classifiers. The training and testing of the classifiers are two independent phases but start from the document-term matrix. However, the whole matrix is not used for training and for testing as this would over-fit the classifier. To prevent over-fitting, a cross validation step is used. A cross validation step, takes the document-term matrix and divides into k subsets. After the division the classifier is trained with $k - 1$ subsets, and the last subset is used to test the accuracy of the classifier [15].

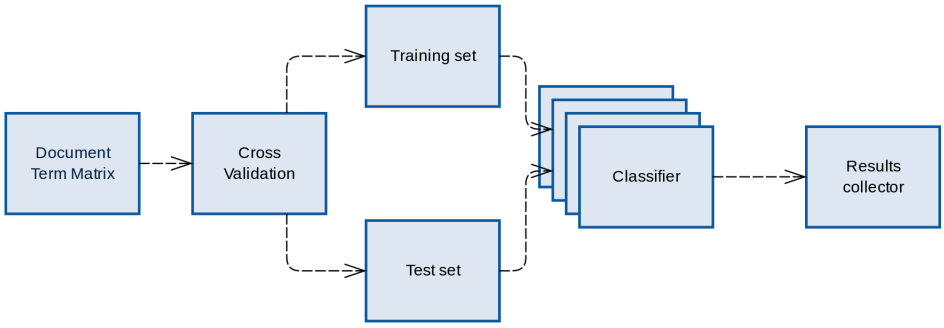

Fig. 3. Creation and evaluation of classifiers

To evaluate the quality of a classification model, different metrics can be used. The most basic metric is the precision, that measure the ratio of the retrieved document given a query that are actually relevant to the query. The recall metric is also frequently used when assessing classifier. This metric is the percentage of relevant documents that are returned as an answer to a given query. The last metric that is commonly used is the harmonic mean F_1 calculated from the ratios of the accuracy and recall [16].

4.2 Components implementation

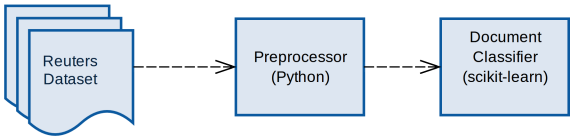

Fig. 4. System architecture

The implemented system architecture is depicted in Figure 4 and has two main components that handle every step of the work-flow. The first component is a python script thats handles the corpus preprocessing and every step associated to the first group of steps. The remain two groups of steps are handled by another component that loads the preprocessed corpus, trains and evaluates the classifiers. This final component was developed in python with the library scikit-learn [17]. The scikit-learn is a machine learning library that eases the development of a text mining classification application by providing high-end routines to load and process data. Furthermore it also contains multiple classifiers and handles most of the complexity of testing, training and evaluating the performance. This library can be compared to the R text mining (tm) library. Another possible option is the Knime framework. However, python and

scikit-learn were used as they offer a complete set of tools, use less resources than Knime and python is a general purpose language unlike R. Furthermore the library has all of the core functionality with a well written documentation. For more in depth comparison of these tools was published by *A. Jovic* [18].

5 Classifiers Evaluation

The classification process follows the steps described in section 4 architecture for four classifiers. The classifiers used to classify the dataset were the naive Bayes for multinomial models, Linear support vector, k-nearest neighbors vote and a Decision tree classifier. Each of the previous classifiers are labeled as follows on the plots: bayes, svc, knn, dtree. Furthermore all of the classifiers, except the k-nearest neighbors used the default parameters specified in the scikit learn documentation [19]. For the k-nearest neighbors it was used a voting with 11 neighbors, where each one had a weight inverse of their distance to the point.

The evaluation process of the classification algorithms assessed the algorithms behavior under different conditions. The algorithms were evaluated with and without an additional preprocessing of the documents. This preprocessing simply removed all of the metadata tags from the documents. The goal of this change was to see how different was the classifiers behaviors with just the body of the documents and without the metadata that contains the topics of the document. In the plots the, the first bar (darker blue) is without preprocessing and the second bar (light blue) has preprocessing. Another variable on these evaluation was the percentage of the document term matrix that was used in the cross validation phase to train the classifier. Four percentage of classifiers were used, 20%, 40%, 60% and 80%. For each experiment, three metrics were measured: precision, recall and harmonic mean.

Figure 5 contains the results of the precision evaluation. In this plot, it is possible to verify that the naive Bayes for multinomial models has the worst precision when compared with the remaining classifiers. Despite having a worst precision, it is always higher than 0.8. However the remaining classifiers have near perfect precision. Regarding the percentage of document term matrix percentage used for training, there is no significant variation in the results. However, the preprocessing step has an impact on the results even though small. On the naive Bayes for multinomial models, the preprocessing step improves the precision results most of the times. However on the k nearest neighbors classifier, the opposite effect can be verified. The preprocessing step decreases the classifier accuracy. The support vector classifier and the decision tree classifier remain mostly consistent across the tests.

The results of the recall metric are plotted in Figure 6. As with the precision results, the naive Bayes for multinomial models have the worst results. In fact, the recall metric is above 0.5 and bellow 0.7. This results are similar for test with or without preprocessing and across the different percentages of matrix-term used for training. Moving to the support vector classifier, the recall metric is also slightly worse than the accuracy results. Despite this, these results are

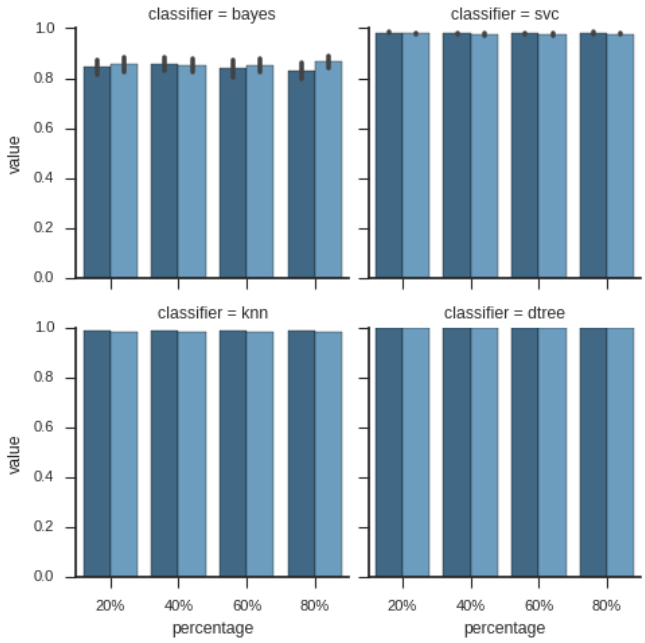

Fig. 5. Precision results for each classifier

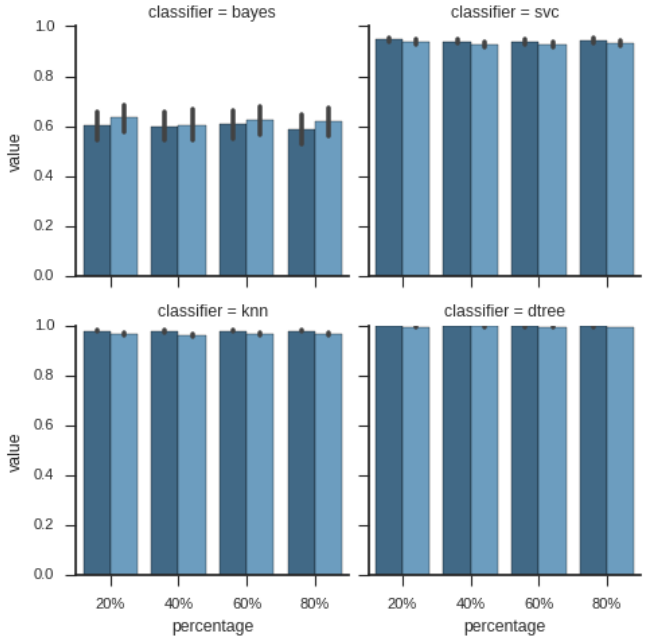

Fig. 6. Recall results for each classifier

far better than the naive Bayes as the results are all above the 0.9 mark. The k nearest neighbor has similar recall results to the support vector classifier. The decision tree classifier has the best recall results, with every result having a 1.0. For different matrix training percentage, the results remain equal across every classifier, however the use of the preprocessing phase has a visible impact on the first three classifiers. Once again, similar to the precision results, the use of the preprocessing has a positive impact on the recall of the naive Bayes results. However, the same is not verified in the support vector classifier and on the k nearest neighbor as there is a slight decrease on the recall results.

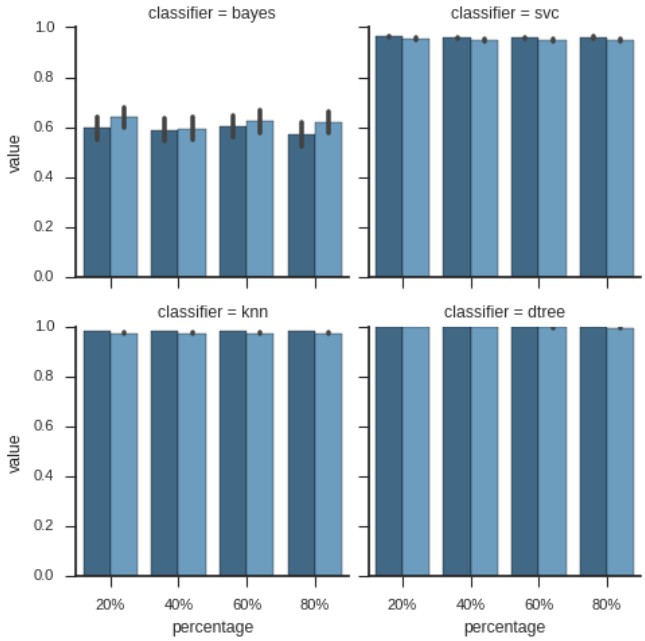

Fig. 7. Harmonic mean for each classifier

Finally, moving on to the harmonic mean results, the same patterns are found as in the recall results. Starting on the naive Bayes, the harmonic results range from 0.5 to 0.7 as in the recall measure. These results are also worse than the results obtained on the precision metric. The support vector classifier and the k nearest neighbor are once again slight worse results than the precision results, however this decrease is just one point of percentage. The decision tree classifier remains with the best results as it has a perfect score on every test. The training percentage used from the term matrix does have a significant impact on the classifier. However, as usual, the preprocessing has a slight impact, consistent with previous observations.

Across every metric, the decision tree is clearly the best classifier, as it has a perfect score in every test variable. However, one important aspect is that,the precision, recall and harmonic mean are strongly related. Taking a look at the naive Bayes, which is the one with the most variation, the precision results show that the documents retrieved by the classifier are mostly relevant, however little more than half of the relevant documents are retrieved. As the harmonic mean is calculated from these two metrics, it is normal to verify that the harmonic mean results do not differ significantly from the first results obtained.

6 Conclusion

This report described the steps taken to select the best classifiers for two categories of the Reuters dataset. The system has three main groups of steps that takes the raw dataset and restructure it in a document term matrix used to train and test four classification algorithms. These algorithms have been evaluated by three metrics, precision, recall and harmonic mean. Each metric provided an insight to the algorithms behavior on different conditions. The classifiers were tested with and without a preprocessing that took into consideration the documents metadata. Furthermore, the classifiers were also tested with different sizes of training data.

From our evaluation, the decision tree classier is the best fit for the Reuters dataset. Besides the results obtained, the decision tree classifiers are a good choice as data analysts find them easy to understand and visualize the decision made by the classier. Furthermore, they are known to have a good performance for numerical and categorical data, even if some assumptions of the data model are violated.

As future work this study can be improved in multiple ways. First and foremost it can study the classifiers results when classifying a dataset with more documents and more classes. As this study is restricted to just two classes, it is not a comprehensive study of the Reuters dataset. Besides the size of documents and classes the classifiers used could also be expanded and even the same classifiers could be tested with different parameters. The classifiers used can also be expanded beyond the classifiers available in scikit, however the library provides additional classifiers, such as: Gaussian process, Random Forest, Neural Net or AdaBoost.

Overall, this report fits to the assignment proposed for the Knowledge discovery databases class, by describing the steps taken to create a text mining classification application capable of processing two classifying two types of reuters documents. The steps taken were also based on the slides indicated on the assignment. This work goes slightly further by evaluating multiple classifiers and presenting their results under different conditions.

References

1. M. A. Hearst, "Untangling text data mining," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Lin-*

guistics, ACL '99, (Stroudsburg, PA, USA), pp. 3–10, Association for Computational Linguistics, 1999.
2. R. Feldman and I. Dagan, "Knowledge discovery in textual databases (kdt)," in *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95*, pp. 112–117, AAAI Press, 1995.
3. Wolfram Research, Inc., "Mathematica 8.0."
4. Semantic Scholar, "Semantic scholar."
5. G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1289–1305, 2003.
6. Y. Chen, G. Wang, and S. Dong, "Learning with progressive transductive support vector machine," *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1845–1855, 2003.
7. T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, pp. 137–142, Springer, 1998.
8. Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 42–49, ACM, 1999.
9. J. D. Rennie, L. Shih, J. Teevan, D. R. Karger, *et al.*, "Tackling the poor assumptions of naive bayes text classifiers," in *ICML*, vol. 3, pp. 616–623, Washington DC), 2003.
10. C. Y. Wang, K. Zhang, Y. G. Yan, and J.-G. Li, "A k-nearest neighbor algorithm based on cluster in text classification," in *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, vol. 1, pp. 225–228, Aug 2010.
11. P. Tan, M. Steinbach, and V. Kumar, "Classification: Basic concepts, decision trees, and model evaluation in introduction to data mining. ed," *AddisonWesley*, p. 769, 2005.
12. S. M. Weiss, N. Indurkhya, T. Zhang, and F. Damerau, *Text mining: predictive methods for analyzing unstructured information*. Springer Science & Business Media, 2010.
13. I. Carnegie Group and L. Reuters, "Reuters-21578."
14. A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing and Management*, vol. 39, no. 1, pp. 45 – 65, 2003.
15. G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *SIGKDD Explor. Newsl.*, vol. 12, pp. 49–57, Nov. 2010.
16. D. M. Powers, "What the f-measure doesn't measure: Features, flaws, fallacies and fixes," *arXiv preprint arXiv:1503.06410*, 2015.
17. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
18. A. Jovic, K. Brkic, and N. Bogunovic, "An overview of free software tools for general data mining," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1112–1117, May 2014.
19. "scikit-learn.org."