Text Classification

Team name on Kaggle: If P=NP then N=1 or P=0
Nissan Pow, Alireza Saberi, Gauravdeep Singh Shami
McGill University, Montreal, Canada

I. INTRODUCTION

The goal of this project was to automatically categorize abstracts of scientific papers into four categories: statistics, math, physics and computer science. We had a set of labeled training data (i.e. abstract text and category) consisting of approximately 100,000 example abstracts with labeled categories as well as a set of unlabeled test data (i.e. abstract text only) with 30,000 abstracts without labeled categories, which needed to be predicted. We used Naive Bayes (NB) as our baseline algorithm, K-Nearest Neighbor (K-NN) as our standard algorithm, and an existing package [1] for SVM as an additional algorithm. To select the optimal parameters for each method, we performed 4-fold cross-validation. Overall we found that SVM produced the most accurate result as measured by the F1 score.
To access to the source code, go to:
https://bitbucket.org/npow/arxiv

II. RELATED WORK

Text classification for document categorization is a problem widely studied in library science, information science and computer science, where the task is to categorize the document into one or more classes or categories. While library science mostly deals with manual classification or labelling, a more advanced algorithmic approach is employed in the information and computer sciences. The documents are categorized based on certain features or attributes [2] contained in them and their relative weight, such as term frequency, in the document.

Automatic document classification tasks can be divided into three types: supervised classification, where some external mechanism (e.g. human feedback) provides information on the correct classification for documents; unsupervised document classification or document clustering, where the classification is done entirely without reference to external information; and semi-supervised document classification, where parts of the documents are labeled by the external mechanism.

The following techniques are commonly used in literature for text classification: Expectation maximization(EM) [3] , Naive Bayes classifier [4], tf–idf (term frequency–inverse document frequency) [5], Latent Semantic Indexing [6], Support vector machines (SVM) [7], neural network [8], K-nearest neighbor algorithms [9], Decision trees [10] such as ID3 or C4.5, Concept Mining [11], Rough set based classifier, Soft set based classifier, Multiple-instance learning [12], Natural language processing approaches.

III. ALGORITHMS

The baseline algorithm employed by us for text classification was Naive Bayes. It has been widely acclaimed [13] in the literature for being a fast and light-weight classifier with results pretty close to other state-of-the-art algorithms such as SVM. The reason for the success of the Naive Bayes classifier lies in the fact that it 'decouples' multiple pieces of features (evidence) and treats each piece of evidence as independent. In simple words, this classifier looks at the evidence (prior probability of features) while deciding which category to put the new sample in. The intuition behind this is to give high probabilities to common occurrences and lower to uncommon ones. The resulting probability for classifying a new sample is found by applying the Bayes rule over P (occurrence /feature). The highest probability class in the likelihood function is the chosen class. If the data-set is pre-processed adequately and a rich feature set selected, Naive Bayes can result in excellent accuracy. We also tried the complement Naïve Bayes [13] variant to increase the accuracy of NB, which will be discussed later on in the paper.

We chose the K-NN classifier as our standard algorithm for classification. This is a classifier whose implementation can be justified intuitively. The algorithm tries to "fit" the new data point using similar data points in the training set. Training this classifier is effortless as the learning phase only consists of storing the multidimensional vectors. Upon arrival of a new vector, the K-NN algorithm simply considers the k-nearest data points to the input vector, k being a user-specified parameter. The outcome can be decided as the mean/median of the outcomes of the k-neighbors in a regression case or simply the distance metric is Hamming distance in case of the discrete variables. [14]

Finally, for the advanced classifier, we employed Support Vector Machines. We used the SciKit Python package to run the code. SVM has been widely considered as the state-of-the-art algorithm in the area of text classification [15]. The basic operational principle of SVM is to reuse the concept of linear classification but to maximize the gap between the hyperplane and the support vectors. New examples are then plotted in the same space as the training sample and their outcome is predicted based on which side of the hyperplane the sample ends-up. Further, "The Kernel Trick"[16] can be used to perform a non-linear classification, mapping inputs in higher-dimensional spaces.

IV. DATA PRE-PROCESSING METHODS

Tokenization [17] is used as a basic pre-processing technique in text classification applications. In this approach, given a defined document unit, the character sequence is chopped into pieces called tokens. Tokens are an instance of a sequence of characters that are grouped together as a useful semantic unit for processing. Tokenization takes care to avoid special characters such as punctuation in the tokens generated.

In our implementation, in addition to splitting on whitespace, we lowercased the words and removed special characters.

V. FEATURE SELECTION METHOD

Feature selection is paramount to any classification problem as the accuracy of the results and the computational time of the algorithm chosen to classify depends heavily on the dimensionality of the dataset and its richness. The main focus in our feature selection strategies has been to retain the most useful features and to disregard features that are noisy and redundant, which add to the dimensionality of the feature space. After a careful survey on the current trends in feature selection for text classification, we went ahead with the following strategies.

A. *Bag-Of-Words*: The bag-of-words [18] is a simplifying representation used in Machine Learning applications that deal with text classification. In this model, the abstract of the text is represented as the bag of its words, disregarding grammar and even its word order. Thus, this approach creates a dictionary of all the distinct words used in the document, and uses them as features. In our implementation there were about 120,000 unique words, considering only unigram. We also tried including bigrams and trigrams, as well as only using bigrams. We tried reducing the number of features by only using the top k most frequently occurring words in the corpus. Also within this approach, we tried different combinations such as with/without stemming, with/without stopword removal etc. Our results indicate that using the full feature set resulted in the best accuracy with stemming, and stopword removal having little to no effect.

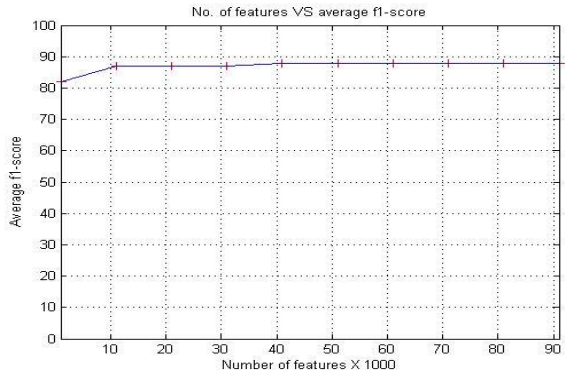

Figure 1: Effect of varying the number of features

#	precision	recall	f1-score	support
CS	*0.88*	*0.85*	*0.86*	*5670*
Math	*0.9*	*0.92*	*0.91*	*4331*
Physics	*0.94*	*0.89*	*0.92*	*5046*
Stat	*0.71*	*0.83*	*0.77*	*2195*
avg/total	*0.88*	*0.83*	*0.88*	*17242*

Table 1: Table for maximum feature 91k

B. *First k-sentences*: In this approach, we used the first k-sentences of the abstract (k=1 to 5) instead of using the whole abstract for the feature selection. The rationale behind this approach being the first few sentences of the abstract are the introduction to the topic and are likely to contain key words that are most relevant to the topic which the abstract belongs to

C. *Non-word features*: Articles from different categories may potentially differ in their styles of writing. Some may be abstract and others more descriptive, even though the vocabulary is the same. In this context, we tried using the following features:

- total number of nouns, verbs, adjectives, adverbs in the abstract
- total number of words in the abstract
- total number of LaTeX formulas in the abstract
- average sentence length
- total number of non-alphabetic characters in the abstract

D. Part-of-speech filtering[19]: In this scheme of features extraction, we used only nouns as features from the corpus in the feature set for training the model

E. *Character n-grams* [20]: We tried the subsequence string kernel SVM[21] method using Shogun[22], but it turned out too slow for practical purposes. Using only sub-sequences of length 1 with the first 100 characters in each abstract failed to finish computation after 12 hours, so we abandoned this approach. However, inspired by this idea we also looked at using character n-grams for 1-5 characters from the first sentence in each abstract. We were able to achieve f-scores of 90% during cross-validation with this.

During the course of the project, we realized that addition of new features onto the feature space hardly affects the results as much as changing the weighting factor for that particular feature. Keeping this is mind experimented with the following feature scaling methods:

A. Term-Frequency (TF): This is simply the raw frequency of the term used in the document. Variations such as Boolean, logarithmically scaled and augmented frequency can be made depending on the data.

B. Term-Frequency Inverse Document Frequency (TF-IDF): This weighting factor is the product of two terms, TF and IDF. The term TF is the same as discussed above. The factor IDF is a measure of how much information the word provides. It is factor to determine whether the term is common or rare across all the documents. It is the logarithmically scaled fraction obtained by dividing the total number of documents by the number of documents containing the term.

C. BM25: The BM25[23] or more popularly known as the Okapi BM25 is a retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the

query terms in the document. It is a family of scoring functions.

D. *Delta TF-IDF*: This scheme of weighting the features calculates the difference ("Delta") of that word's TF-IDF score in the positive and negative corpora. This term frequency transformation is supposed to boost the importance of words that are unevenly distributed between positive and negative classes and discounts evenly distributed words.

E. *Bi-Normal Separation (BNS)[24]:* This method is also an improvement over the TF-IDF method which scales some feature inappropriately. This method was invented specifically for text classification using SVM. The BNS score for each value is computed and the TF-IDF score is replaced by TF-BNS. The BNS score is implemented using the difference between the inverse Normal cumulative distribution function of the true positive rate and the false positive rate.

Apart from the five methods listed above, we also used a combination of Delta-TFIDF and BM25. Computing Delta-IDF and BNS was quite slow since we had to create 4 different vectorizations (one for each class). During cross-validation, Delta-TFIDF with BM25 turned out to be the best by a very small margin, closely followed by BM25 and TFIDF (all achieving f-scores around 89%). TF and BNS performed significantly worse, with f-scores around 87%.

VI. Optimization & parameter selection

Although Naive Bayes is a very widely used classifier, it has its own systemic issues. Naive Bayes tends to select poor decision boundary weights when the distribution of classes is skewed. In the training set, we found that "statistics" made up about 12.7% of the examples, while "cs" made up 32.7%, with "physics" and "math" being 29.3% and 25.3% respectively. Thus we see that "cs" occurs almost 3 times more frequently than "stat". Naive Bayes also assumes the features to be independent of each other. To mitigate these issues, we tried an alternate version of the multinomial Naïve Bayes algorithm, the Complement Naive Bayes (CNB) [13] .The CNB classifier estimates feature probabilities for a class, Y, based on the samples in the complement class, Y'.
The following formula is used in CNB:

$$\hat{\theta}_{\tilde{c}i} = \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha}$$

Where $\hat{\theta}_{\tilde{c}i}$ is the probability that word i occurs in that class. $N_{\tilde{c}i}$ is the number of times word i occurred in documents in classes other than c and $N_{\tilde{c}}$ is the total number of word occurrences in classes other than c, and αi and α are smoothing parameters. We have found that CNB performs only slightly worse than the normal version of multinomial Naïve Bayes, while taking significantly longer to train. These results were also obtained by [25].

We found k-NN to be extremely slow to create the predictions, since for each test case it needs to find the k nearest neighbors within the entire training set. Furthermore, since we had over 100,000 features it was computationally challenging to fit our feature vector within memory. To overcome these features, we decided to limit our training set to 10,000 examples and perform feature reduction using Latent Semantic Analysis [26]. We also used a ball-tree [27] to store our training examples in order to facilitate quick neighbor queries. With these modifications, we were able to achieve about 79% accuracy using 4-fold cross-validation.

For k-NN: The choice of k heavily impacts the results. A lower k causes very non-linear functions to be approximated, but noise in the data is also captured. Thus bias is low and variance of the output is high. On the other hand, a higher k causes the output to be much less sensitive to variations and makes the output smoother. Thus bias is high, but the variance is low.

A lot of research has gone in the selection of proper value of k. A popular approach is to use evolutionary algorithms to optimize feature scaling. Similarly, feature scaling by the mutual information of training data with the training classes is also used widely. Some basic tricks can also help sometimes such as choosing an odd numbered k in binary classification problems. Also the bootstrap method can be used for an empirically optimal k.

From the results of our cross-validation, we have found that k=2 with 26 features produced the best results (see the results in tables 2 to 7)
For SVM, The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C. We tried using non-linear kernels but found them to slow for practical purposes on this dataset. To tune the parameters of our SVM, we used GridSearchCV for values of C = [0.1, 1 ,10, 100, 1000, 10000], class_weight = [None or 'auto'], loss = ['l1','l2'], penalty = ['l1','l2']. GridSearchCV cross-validates with every combination of the parameter to choose the best parameter.
We used 4-fold cross-validation on our training set before submitting to the leaderboard on Kaggle. The leaderboard acted as a hold-out test set, which enabled us to see how well our model would generalize to new test cases. We found that there was very little difference between our cross-validation results and the leaderboard results, which was a good sign that we were not over-fitting to our training set.

#	*precision*	*recall*	*f1-score*	*support*
CS	*0.88*	*0.85*	*0.87*	*7056*
Math	*0.91*	*0.93*	*0.92*	*5449*
Physics	*0.94*	*0.90*	*0.92*	*6317*
Stat	*0.72*	*0.84*	*0.77*	*2729*
avg/total	*0.89*	*0.88*	*0.88*	*21551*

Table 2: Naive Bayes Classification Report

	CS	Math	Physics	Stat
CS	5994	351	188	523
Math	154	5053	101	141
Physics	337	79	5677	224
Stat	288	82	72	2287

Table 3: Confusion Matrix for Naive Bayes

#	precision	recall	f1-score	support
CS	0.87	0.89	0.88	5670
Math	0.91	0.94	0.92	4331
Physics	0.93	0.92	0.92	5046
Stat	0.83	0.74	0.78	2195
avg/total	0.88	0.83	0.88	17242

Table 4: SVM Classification Report

	CS	Math	Physics	Stat
CS	5054	237	172	207
Math	148	4054	84	45
Physics	245	69	4652	80
Stat	374	100	105	1616

Table 5: Confusion Matrix for SVM

#	precision	recall	f1-score	support
CS	0.73	0.88	0.80	798
Math	0.87	0.86	0.86	652
Physics	0.89	0.84	0.86	741
Stat	0.81	0.53	0.64	308
avg/total	0.82	0.82	0.81	2499

Table 6: K-NN Classification Report

	CS	Math	Physics	Stat
CS	699	33	39	27
Math	73	561	14	4
Physics	86	28	620	7
Stat	97	25	24	162

Table 7: Confusion Matrix for K-NN

.

VII. DISCUSSION

Naive Bayes can perform surprisingly well despite its simplifying assumptions, and we were able to achieve results that were competitive with state of- the-art text classification algorithms such as SVM.

K-NN performed the worst out of the algorithms that we tried. We were only to achieve f1-scores of 80% during cross-validation. This can probably be explained by the fact that in high dimensions, all examples look alike so that the choice of nearest neighbors is effectively random.[28]

In general, we found that using TF-IDF resulted in improved accuracy compared to just using TF, especially so for K-NN where there was a difference of several percentage points.

From the classification reports, we see that the F1-scores of "cs" and "stat" are consistently lower than that of "math" and "physics". This is probably because "cs" and "stat" are the majority and minority classes in the training set so "cs" would tend to be predicted more often than ideal, and vice-versa for "stat". We tried to mitigate the skewness of the distribution by undersampling "cs" and oversampling "stat" so that the number of training examples in each class would be roughly the same. We also investigated generating synthetic training data by randomly choosing two abstracts within one class and concatenating the first two halves of them. However, we were unable to improve upon the results using either of these techniques.

We hereby state that all the work presented in this report is that of the authors.

.

VIII. .”REFERENCES

[1] http://scikit-learn.org/stable/modules/svm.html

[2] Library of Congress (2008). The subject headings manual. Washington, DC.: Library of Congress, Policy and Standards Division. (Sheet H 180: "Assign headings only for topics that comprise at least 20% of the work.")

[3] Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of the Royal Statistical Society, Series B 39 (1): 1–38

[4] Rish, Irina (2001). "An empirical study of the naive Bayes classifier". IJCAI Workshop on Empirical Methods in AI

[5] Robertson, Stephen. "Understanding inverse document frequency: On theoretical arguments for IDF". Journal of Documentation 60 (5): 503–520

[6] Dumais, S.; Platt, J.; Heckerman, D.; Sahami, M. (1998). "Inductive learning algorithms and representations for text categorization". "Proceedings of the seventh international conference on Information and knowledge management - CIKM '98". p. 148

[7] Cortes, C.; Vapnik, V. (1995). "Support-vector networks".Machine Learning 20 (3): 273

[8] Russell, Ingrid. "Neural Networks Module". Retrieved 2012

[9] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician 46 (3): 175–185

[10] Rokach, Lior; Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc. ISBN 978-9812771711

[11] Yuen-Hsien Tseng, Chun-Yen Chang, Shu-Nu Chang Rundgren, and Carl-Johan Rundgren, " Mining Concept Maps from News Stories for Measuring Civic Scientific Literacy in Media", Computers and Education, Vol. 55, No. 1, August 2010, pp. 165-177.

[12] Christopher D. Manning, Hinrich Schütze: Foundations of Statistical Natural Language Processing, MIT Press (1999), ISBN 978-0-262-13360-9, p. xxxi

[13] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, David R. Karger "Tackling the Poor Assumptions of Naive Bayes Text Classifiers" Artificial Intelligence Laboratory; Massachusetts Institute of Technology; Cambridge, MA 02139

[14] Prof. Pineau's class notes, Comp 598. http://www.cs.mcgill.ca/~jpineau/comp598/Lectures/09InstanceLearning-publish.pdf

[15] Burges, Christopher J. C.; A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2:121–167, 1998

[16] Prof. Pineau's class notes, Comp 598. http://www.cs.mcgill.ca/~jpineau/comp598/Lectures/13SVM2-publish.pdf

[17] Lita et al. (2003) present a method for truecasing . Natural language processing work on computational morphology is presented in (Sproat, 1992, Beesley and Karttunen, 2003).

[18] George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* 3 (March 2003), 1289-1305.

[19] Google code from: https://code.google.com/p/cleartk/wiki/TutorialPartOfSpeechClassifier

[20] William B. Cavnar and John M. Trenkle, "N-Gram-Based Text Categorization", In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval

[21] Lodhi, Huma; Saunders, Craig; Shawe-Taylor, John; Cristianini, Nello; Watkins, Chris (2002). "Text classification using string kernels".Journal of Machine Learning Research: 419–444

[22] http://www.shogun-toolbox.org

[23] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In Proceedings of the Third Text REtrieval Conference (TREC 1994). Gaithersburg, USA, November 1994.

[24] George Forman. 2008. BNS Feature Scaling: An Improved Representation over TF·IDF for SVM Text Classification. 17th ACM Conference on Information and Knowledge Management, August 2008.

[25] Antti Puurula. Scalable Text Classification with Sparse Generative Modeling: http://www.cs.waikato.ac.nz/~asp12/publications/Puurula_12.pdf

[26] Susan T. Dumais (2005). "Latent Semantic Analysis". Annual Review of Information Science and Technology 38: 188.doi:10.1002/aris.1440380105

[27] Omohundro, Stephen M. (1989) "Five Balltree Construction Algorithms"

[28] Pedro Domingos. 2012. A few useful things to know about machine learning. *Commun. ACM* 55, 10 (October 2012), 78-87. DOI=10.1145/2347736.2347755 http://doi.acm.org/10.1145/2347736.2347755