Text Classification using Naive Bayes

Name: Lin Sun
Email: lxs5171@mavs.uta.edu

October 20, 2021

Abstract

Naive Bayes classifier might be the most popular and efficient classifier for text classification in the era of traditional machine learning, for its simplicity in concept and high performance. In this project, we step by step built a Naive Bayes classifier for a corpus with messages from 20 newsgroups. An optimal smoothing factor is selected by cross-validation, with which the overall average precision on these 20 categories can reach up to 88.5% without any help of feature selection or any other methods. We also conducted simple noise-removing method and can see up to 17.6% improvement of the precision performance for some categories, which indicating a large room for feature selection research in this area.

1 Naive Bayes Classifier

Naive Bayes Classifier is a very useful tool has a wide range application. It is a kind of generative classifier that performs supervised machine learning in comparison with the discriminative and unsupervised ones. It makes prediction according to the combination of prior probabilities. In the era of traditional machine learning, it could be the best classifier against several common classifiers (such as decision tree, neural network, and support vector machines) in term of accuracy and computational efficiency [13] when tuned carefully. Naive Bayes Classifier is derived from Bayes Rules [17] and constrained by an independence assumption:

- **Bayes Rule** is a theorem in probability and statistics, describes the probability of an event based on prior knowledge of conditions that might be related to the event [17].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

  The equation 1 of Bayes Rule indicates that we can make predictions according to certain properties of the event, if we extend event B to a set of events $\{X_1, X_2, \ldots, X_n\}$ related to event A.

$$P(A_i|X_1 \ldots X_n) = \frac{P(A_i)P(X_1...X_n|A_i)}{\sum_j P(A_j)P(X_1...X_n|A_j)} \tag{2}$$

  From equation 2 we can see the prediction of posterior probabilities can be transformed to the evaluation of a set prior probabilities. However, it's impractical for the unbiased learning of Bayes Classifier by counting every possible combination of the related events, because there would have $2(2^n - 1)$ parameters to evaluate.

- **Independence Assumption** is necessary to reduce the evaluation space to only $2n$ parameters, where we assume every related event X_i is conditionally independent of any other related event X_j given any possible value of event A. With this assumption, we can represent the combination of related events $P(X_1 \ldots X_n|A)$ as $P(X_1|A) \cdot P(X_2|A) \cdots P(X_n|A)$. i.e.,

$$P(X_1 \ldots X_n|A) = \prod_{i=1}^{n} P(X_i|A) \tag{3}$$

Then, the equation of Bayes Rule can be simplified to:

$$P(A_i|X_1 \dots X_n) = \frac{P(A_i)\prod_k P(X_k|A_i)}{\sum_j P(A_j)\prod_k P(X_k|A_j)} \tag{4}$$

And for every possible value A_i the denominator is the same, so to predict which A_i is the most possible value, is to estimate the maximum likelihood of A:

$$A \leftarrow \arg\max_j P(A_j)\prod_i P(X_i|A_j) \tag{5}$$

Therefore, we can simplify the parameter evaluation and only use the conditional probabilities of individually related event instead of their combinations. However, we need to keep in mind of the assumption which is the basis of Naive Bayes, and try to minimize the conditional dependence between the related events.

2 Text Classification

Text classification is an important topic among traditional Text Retrieval [8], which is a branch of Natural Language Processing (NLP) [3]. It's to classify each document into pre-learned classes, usually by the Bag-of-words feature model [16] or/and semantic model [12] using Ontology [1]. When there are only two classes it becomes a filtering task, e.g., filter out spam emails from incoming message streams. As figure 1 shown, a typical text classification system is usually layered into three layers [19]. The learning and classification modules on the top and share the underlying layers, by which the original documents (usually semi-structured or unstructured) in the corpus are transformed to structured representations.

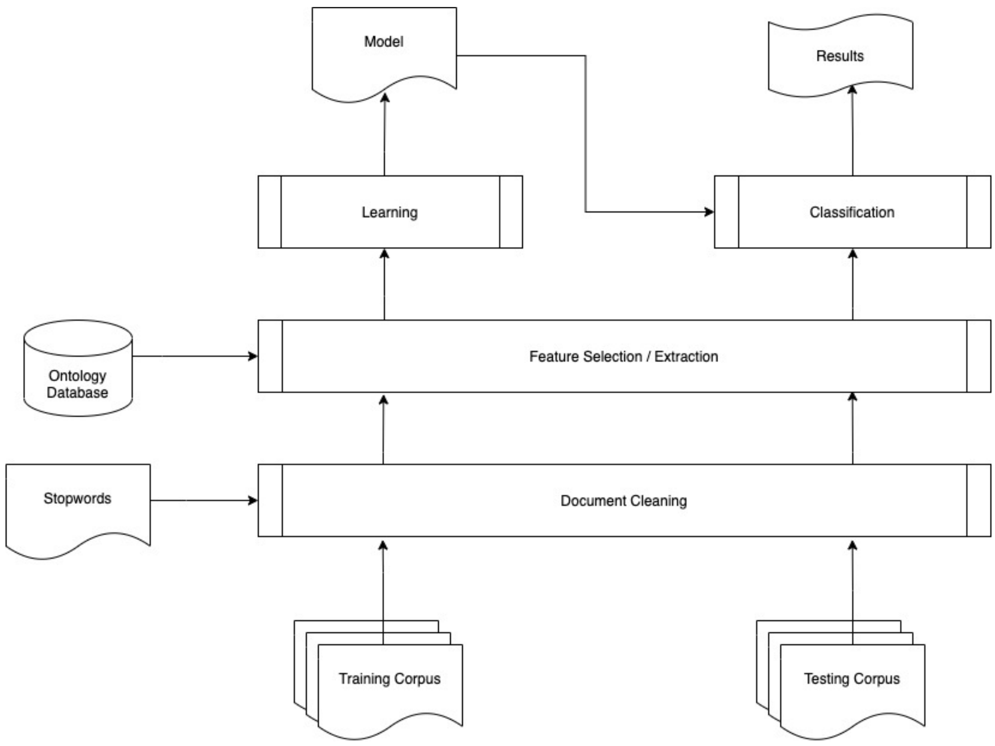

Figure 1: Typical Text Classification system

- the *Document Cleaning* layer [19] is to parse the document into "bag-of-words" and try to eliminate obvious noises, such as bad format, words containing characters in unrecognizable codes, case sensitivity, hyphens, and so on.

- the *Feature Selection* layer, also known as *Feature Extraction*, is to transform the "bag-of-words" into features [6]. This is a critical step for text classification which could dramatically influence the performance of learning and classification. For example, as evaluated in TREC2005, when domain-specific features are selected for biomedical articles, the classification performance outperformed its contemporary competitors a lot [9]. The ontology based and other kinds of semantic analysis can be applied here to improve feature quality [14, 15].

3 Text Classification using Naive Bayes Classifier

Naive Bayes could be the most popular classifier for text classification [5, 7, 2, 4], for its simplicity in concept and high performance in both accuracy and efficiency [13]. As discussed in section 1, in order to use Naive Bayes classifier, the prediction needs to be made according to a collection of related events. In other words, the features of each samples are considered as related events of whether a sample belongs to a class or not. When applied in text classification, we need to evaluate the probability of each feature (known as **term**, or word) related with each class. And according to the independence assumption, it's better to make all the features conditional independent with each other, given the classes.

3.1 Learning: Parameter Evaluation

Basically there are three ways to evaluate the probability for each term against each class, using Maximum Likelihood Estimation (MLE)

- **Binary**: whether a term appears in a class of documents. No matter how many times it appears, only count once.

$$P^b_{ij} = P(term_i|class_j)_{binary} = \begin{cases} 1, & \text{if } term_i \text{ appears in any documents of } class_j \\ 0, & \text{if } term_i \text{ does not appear in } class_j \end{cases} \tag{6}$$

- **Document Frequency**: how many documents a term appears in a class. Specifically,

$$P^{df}_{ij} = P(term_i|class_j)_{df} = \frac{\#D\{\text{documents contain } term_i \text{ in } class_j\}}{\#D\{\text{documents in } class_j\}} \tag{7}$$

- **Term Frequency**: how many times a term appears in all the documents of a class, i.e., regarding all the documents in a class as a single document:

$$P^{tf}_{ij} = P(term_i|class_j)_{tf} = \frac{\#W\{term_i \text{ in } class_j\}}{\#W\{\text{all terms in } class_j\}} \tag{8}$$

We can easily see equation 6 and equation 8 are two extreme cases of calculating the conditional probability. A good model usually comes from somewhere between them. Essentially, equation 8 can be derived from equation 7 multiply the ratio weight of the $term_i$:

$$P^{tf}_{ij} = P^{df}_{ij} \times \frac{\text{average of } term_i \text{ in } class_j \text{ documents that contains it}}{\text{average of terms in all documents of } class_j} \tag{9}$$

And equation 7 and equation 8 are the Maximum Likelihood Estimation of the conditional probability of each $term_i$ in $class_j$. However, it would be totally different for the prior probability of each class when applying these two MLE equations.

- **Prior probability of each class when using document frequence** should count for the document number among the corpus

$$P^{df}_{A=j} = \frac{\#D\{\text{documents in } class_j\}}{\#D\{\text{documents in the corpus}\}} \tag{10}$$

- **Prior probability of each class when using term frequence** should count for the amount of terms in each class among the corpus

$$P^{tf}_{A=j} = \frac{\#W\{\text{terms in } class_j\}}{\#W\{\text{terms in the corpus}\}} \tag{11}$$

3.2 Prediction

After representing an test sample as a set of terms. The classifier is to estimate the most possible class using the prior conditional probabilities of each term and class:

$$class \leftarrow \arg\max_j P(class_j) \prod_i P(term_i|class_j) \tag{12}$$

Where the conditional probability $P(term_k|class_i)$ comes from the results of parameter evaluation. As aforementioned, when applying *document frequency* or *term frequency*, the prior probability of each class $P(class_i)$ should change accordingly.

However, when performing prediction based very large dimension feature space, the product of $P(class_i) \prod_k P(term_k|cla$ could be too small to have any difference against each other. Instead, *log* function is usually introduced to get comparable results which is equivalent for the MLE equation 12:

$$class \leftarrow \arg\max_j \ \log(P(class_j)) + \sum_i \log\left(P(term_i|class_j)\right) \tag{13}$$

4 Naive Bayes Classification on 20 Categories Newsgroup

The *20 Categories Newsgroup* corpus [11] contain around 20,000 newsgroup messages drawn from 20 newsgroups, as table 1 shown, each of which contain around 1,000 messages.

Class	**Amount of messages**
Italk.politics.mideast	1,000
rec.autos	1,000
comp.sys.mac.hardware	1,000
alt.atheism	1,000
rec.sport.baseball	1,000
comp.os.ms-windows.misc	1,000
rec.sport.hockey	1,000
sci.crypt	1,000
sci.med	1,000
talk.politics.misc	1,000
rec.motorcycles	1,000
comp.windows.x	1,000
comp.graphics	1,000
comp.sys.ibm.pc.hardware	1,000
sci.electronics	1,000
talk.politics.guns	1,000
sci.space	1,000
soc.religion.christian	997
misc.forsale	1,000
talk.religion.misc	1,000

Table 1: Corpus sample distribution

According to the architecture illustrated in figure 1, we build a text classification system using Naive Bayes classifier. Common stopwords list [10] is introduced to improve the quality of features. After *document cleaning* and removing the stopwords, each single word is regard as a feature which is usually called as **term** in NLP. Although there are plenty of method to improve feature quality, here we adopt this most simple and intuitive way because the objective of this project is to evaluate the performance of *Naive Bayes Classifier*. We even would not bother to use large dimension feature space directly without any reduction.

During the learning phase, the prior conditional probability of each term is evaluated using aforementioned equations, and then stored in a matrix that we call it as *Prior Probability Matrix* mark as **PPM**:

$$PPM = \left[x_{ij}\right] \tag{14}$$

where i is the *ith* term among all distinct terms in the entire corpus, j is the *jth* class. The shape of PPM is $(\#\{classes\} \times \#\{termsinthecorpus\})$

Then for the prediction, we apply equation 13 for each test document. However, there is a problem when applying this equation directly. For any new words in the test document that appeared in the corpus but not appeared in the training set for a particular class, their conditional probabilities are zero, which is illegal for *log* calculation and would make zero prediction if applying equation 12 directly by multiply the conditional probability of each term that appears in the test document.

To solve this problem, a **smoothing factor** is introduced for *parameter evaluation*:

$$P_{ij}^{df} = P(term_i|class_j)_{df} = \frac{\#D\{\text{documents contain } term_i \text{ in } class_j\} + l}{\#D\{\text{documents in } class_j\} + lJ} \tag{15}$$

where l is the *smoothing factor* which usually takes a value in the range of $(0,\ 1]$, and J is the possible values that *whether a document contains term$_i$* which obviously is $J = 2$. And the equation 15 corresponds to *MAP* estimation. When $l = 1$ it is called *Laplace smoothing*, and when $l = 0$ it is degenerating to the *MLE* estimation which is identical with equation 7. And it is the same if we are going to adopt *term frequency* using equation 8.

Since there is a undecided parameter in the evaluation, i.e., the *smoothing factor* l, it is natural to use *Cross Validation* to select an optimal value.

5 Cross Validation by Rolling

Cross Validation is a useful tool to evaluate the performance of a model, especially useful for comparing different models or different settings of a model [18]. In general, it is leveraging disjoint subsets of the randomly shuffled sample set to in turn evaluate the precision, and output an overall score upon all these subsets. There are many strategies when applying cross-validation, which mostly by changing the amount and combination of subsets for testing and using the rest for training during each time of evaluation.

5.1 Corpus separation

However, in this project we are required to use half of the corpus for training and the other half for testing, i.e.,

$$ratio_{test/training} = \frac{\#\{\text{samples for testing}\}}{\#\{\text{samples for training}\}} = 1 \tag{16}$$

So, we designed a cross-validation method by rolling across each samples in the class. For each round of training and testing, we split the corpus using a window rolling by a shift:

$$sample_i \text{ belongs to} \begin{cases} \text{training set, if } ((i+s) \mod w) < \frac{1}{2}w \\ \text{testing set, otherwise} \end{cases} \tag{17}$$

where w is the *window size* and s is the *shift* for each round. In the project, we set $w = 10$. So, for each round of evaluation there would be $\frac{1}{10}$ different samples in the training set and testing set.

5.2 Performance Evaluation

For each round of evaluation, we calculate the *MSE (Mean Square Error)* as the performance index:

$$MSE = \frac{\sum_j (1 - \text{precision}_j)^2}{\#\{classes\}} \tag{18}$$

where,

$$precision_j = \frac{\#\{\text{correctly classified samples in } class_j\}}{\#\{\text{samples in } class_j\}} \tag{19}$$

After rolling over the corpus, the mean of MSEs for different rounds is used as the performance index for a particular parameter value, i.e., the smoothing factor. Then, the value with minimum $\text{perf}_{\text{smooth factor}} = \frac{\sum MSE_s}{\#\{\text{rounds of rolling}\}}$ is selected as the optimal value of the smoothing factor.

f	**avg**	shift 0	shift 1	shift 2	shift 3	shift 4	shift 5	shift 6	shift 7	shift 8	shift 9
0.1	0.025	0.023	0.022	0.022	0.023	0.024	0.026	0.028	0.027	0.029	0.026
0.2	0.026	0.024	0.023	0.023	0.023	0.025	0.028	0.029	0.028	0.029	0.027
0.3	0.027	0.025	0.024	0.023	0.024	0.026	0.029	0.03	0.028	0.03	0.028
0.4	0.028	0.026	0.025	0.024	0.025	0.026	0.029	0.031	0.029	0.031	0.029
0.5	0.028	0.027	0.025	0.025	0.025	0.027	0.03	0.032	0.029	0.032	0.029
0.6	0.029	0.027	0.026	0.025	0.026	0.027	0.03	0.032	0.03	0.033	0.03
0.7	0.029	0.028	0.026	0.026	0.027	0.027	0.031	0.033	0.03	0.033	0.03
0.8	0.03	0.029	0.027	0.027	0.027	0.028	0.031	0.034	0.031	0.034	0.031
0.9	0.03	0.029	0.027	0.027	0.028	0.028	0.031	0.034	0.032	0.034	0.031
1.0	0.031	0.03	0.028	0.028	0.029	0.029	0.032	0.035	0.033	0.035	0.032

Table 2: MSE score of Cross Validation by Rolling

6 Feature Selection

The only feature selection method we applied is the *stopwords.* And the raw words are used directly as features with minimum document cleaning, that only trimmed words to eliminate special characters on both sides of each word, and turned words into lower cases to uniform each term.

The feature space is sparse. Table 3 shows the feature space distribution for the training set with shift = 0 as an example. Each category only occupies a little portion with the overall feature space, which varies from 6.8% to 12.9%. What for sure is, the sparse feature space with hundreds of thousands of features, introduces large of computation difficulties in precision and efficiency, while Naive Bayes classifier is able to doing well on both of them.

class	**terms**	**portion**
overall	171,422	100%
Italk.politics.mideast	20,222	11.8%
rec.autos	13,352	7.8%
comp.sys.mac.hardware	11,723	6.8%
alt.atheism	13,276	7.7%
rec.sport.baseball	12,832	7.5%
comp.os.ms-windows.misc	22,054	12.9%
rec.sport.hockey	16,275	9.5%
sci.crypt	16,706	9.7%
sci.med	18,053	10.5%
talk.politics.misc	18,624	10.9%
rec.motorcycles	13,145	7.7%
comp.windows.x	19,428	11.3%
comp.graphics	17,430	10.2%
comp.sys.ibm.pc.hardware	12,477	7.3%
sci.electronics	13,327	7.8%
talk.politics.guns	16,476	9.6%
sci.space	16,324	9.5%
soc.religion.christian	15,322	8.9%
misc.forsale	13,605	7.9%
talk.religion.misc	16,937	9.9%

Table 3: Feature space distribution

However, whether the portion a category takes is small or big, does not necessarily lead to poor or good precision performance. Let's take the categories with highest and lowest precision as an example, as table 4 shows, the portion they take in the feature space is similar to each other, and both are close to the average level.

class	**precision**	**portion**
soc.religion.christian	0.993	8.9%
talk.religion.misc	0.542	9.9%

Table 4: Feature space vs precision performance

7 Results

To simplify the model and to focus on the performance of the classifier, we only adopt the parameter evaluation method of *document frequency*. A set of cross-validation is applied against the smoothing factor in the range of $[0.1, 1.0]$. As table 2 shown, the optimal value of smoothing factor is 0.1 which comes out with the minimum average MSE score.

Then, we use the optimal smoothing factor to train on the corpus using half of the samples and to test with the other half. In the end, we take the average precision as the performance index for each class by the aforementioned cross-validation method. As table 5 shows, the precision performance for these 20 classes varies from 54.2% for the class *talk.religion.misc* to 99.3% for the class *soc.religion.christian*. The overall precision for all these 20 classes is 88.5%.

We also find a phenomenon that the lowest precision class *talk.religion.misc* is usually the biggest error source for other classes. Once it is removed from the corpus, as table 6 shows, the performance for some other classes can be improved greatly. For the class *alt.atheism*, its precision improved from 77.7% to 91.4% with 17.6% improvement. This may indicates some degrees of correlation between these two classes regarding the selected terms. As we know, the Naive Bayes classifier assumes conditional independent for the features. So, the simple result of table 6 intuitively indicates that the performance of Naive Bayes classifier can be improved greatly by sophisticated feature selection, and particularly, there is a large room for domain-specific feature selection research in this area.

class	**avg**	shift 0	1	2	3	4	5	6	7	8	9
alt.atheism	0.777	0.71	0.72	0.79	0.82	0.81	0.84	0.83	0.74	0.74	0.76
comp.graphics	0.883	0.94	0.91	0.91	0.89	0.84	0.83	0.84	0.86	0.88	0.92
comp.os.ms-windows.misc	0.844	0.82	0.84	0.83	0.86	0.87	0.87	0.86	0.85	0.83	0.8
comp.sys.ibm.pc.hardware	0.866	0.86	0.87	0.84	0.83	0.82	0.85	0.88	0.9	0.91	0.9
comp.sys.mac.hardware	0.899	0.89	0.9	0.93	0.93	0.91	0.9	0.92	0.86	0.88	0.87
comp.windows.x	0.936	0.92	0.93	0.93	0.95	0.94	0.95	0.96	0.95	0.92	0.92
misc.forsale	0.731	0.75	0.75	0.74	0.72	0.71	0.71	0.71	0.75	0.73	0.74
rec.autos	0.915	0.9	0.92	0.93	0.92	0.92	0.9	0.92	0.91	0.91	0.92
rec.motorcycles	0.953	0.95	0.94	0.94	0.95	0.95	0.95	0.98	0.96	0.96	0.96
rec.sport.baseball	0.964	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.96	0.97	0.96
rec.sport.hockey	0.981	0.98	0.98	0.97	0.97	0.97	0.98	0.98	0.99	0.99	0.99
sci.crypt	0.975	0.96	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.96	0.95
sci.electronics	0.869	0.87	0.87	0.83	0.83	0.86	0.86	0.88	0.9	0.88	0.91
sci.med	0.939	0.96	0.95	0.93	0.93	0.94	0.92	0.93	0.94	0.94	0.94
sci.space	0.957	0.94	0.96	0.96	0.95	0.97	0.97	0.96	0.95	0.96	0.95
soc.religion.christian	0.993	0.99	0.99	1.0	0.99	0.99	0.98	1.0	1.0	1.0	0.99
talk.politics.guns	0.898	0.9	0.91	0.9	0.87	0.89	0.89	0.89	0.89	0.92	0.91
talk.politics.mideast	0.975	0.97	0.97	0.96	0.96	0.98	0.98	0.98	0.98	0.98	0.98
talk.politics.misc	0.799	0.79	0.77	0.8	0.79	0.78	0.8	0.81	0.82	0.81	0.81
talk.religion.misc	0.542	0.61	0.63	0.59	0.58	0.58	0.52	0.45	0.49	0.47	0.51

Table 5: Precisions of each class with the optimal smoothing factor 0.1

class	after	before
alt.atheism	0.914	0.777
comp.graphics	0.91	0.883
comp.os.ms-windows.misc	0.84	0.844
comp.sys.ibm.pc.hardware	0.872	0.866
comp.sys.mac.hardware	0.904	0.899
comp.windows.x	0.932	0.936
misc.forsale	0.752	0.731
rec.autos	0.926	0.915
rec.motorcycles	0.938	0.953
rec.sport.baseball	0.956	0.964
rec.sport.hockey	0.976	0.981
sci.crypt	0.98	0.975
sci.electronics	0.87	0.869
sci.med	0.954	0.939
sci.space	0.964	0.957
soc.religion.christian	0.992	0.993
talk.politics.guns	0.936	0.898
talk.politics.mideast	0.974	0.975
talk.politics.misc	0.852	0.799

Table 6: Comparison of precision before and after noise removing

References

[1] Maedche A. and Staab S. *Handbook on Ontologies*, chapter Ontology Learning. International Handbooks on Information Systems, Springer, Berlin, Heidelberg, 2004. URL: `https://doi.org/10.1007/978-3-540-24750-0_9`.

[2] Jingnian Chen, Houkuan Huang, Shengfeng Tian, and Youli Qu. Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3):5432–5435, 2009.

[3] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[4] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545, 2007.

[5] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, 18(11):1457–1466, 2006.

[6] David D Lewis. Feature selection and feature extraction for text categorization. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[7] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[8] NIST. Text Retrival Conference (TREC). URL: `https://trec.nist.gov/`.

[9] Junyu Niu, Lin Sun, and etc. Wim at trec 2005. In *Text Retrieval Conference (TREC)*. NIST, 2005. URL: `https://trec.nist.gov/pubs/trec14/papers/fudanu-sun.geo.ent.pdf`.

[10] ranks.nl. Stopwords list. URL: `https://www.ranks.nl/stopwords`.

[11] Jason Rennie. 20 newsgroup messages. URL: `https://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html`.

[12] Bloehdorn S. and Hotho A. *Advances in Web Mining and Web Usage Analysis*, chapter Boosting for Text Classification with Semantic Features. WebKDD 2004. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2006. URL: `https://doi.org/10.1007/11899402_10`.

[13] S.L. Ting, W.H. Ip, and Albert H.C. Tsang. Is naïve bayes a good classifier for document classification? *International journal of software engineering and its applications*, 5(3):37–46, 2011.

[14] Carlos Vicient, David Sánchez, and Antonio Moreno. An automatic approach for ontology-based feature extraction from heterogeneous textualresources. *Engineering Applications of Artificial Intelligence*, 26(3):1092–1106, 2013.

[15] Bill B Wang, RI Bob Mckay, Hussein A Abbass, and Michael Barlow. A comparative study for domain ontology guided feature extraction. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 69–78, 2003.

[16] Wikipedia. Bag-of-words. URL: `https://en.wikipedia.org/wiki/Bag-of-words_model`.

[17] Wikipedia. Bayes Rule. URL: `https://en.wikipedia.org/wiki/Bayes%27_theorem`.

[18] Wikipedia. Cross Validation. URL: `https://en.wikipedia.org/wiki/Cross-validation_(statistics)`.

[19] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54, 1998.