Naïve Bayes Text Classification: A Statistical Natural Language Processing Project

Chris Monson

1 Abstract

Naïve Bayes is surprisingly good at document classification. The fact that good accuracy can be obtained from a simple bag of words model is interesting, and seems to be a good fit for the filtering of adult content on the web. That web filtering is interesting in general is well known [1]. Documents with "adult" content (really not even fit for adults) often contain terms which are good indicators by themselves, making Naïve Bayes an obvious choice. This document explores the performance of Naïve Bayes as such a classifier, using a large set of volunteer-provided data.

2 Introduction

Much work has been done in the field of statistical text classification. Among the methods explored are Naïve Bayes[2–5], Maximum Entropy [6], Support Vector Machines (and variants) [7,8], Mean Frequency Discriminant [8], K-Nearest Neighbor, and others [9–11]. While all of these methods are interesting in their own right, none has received so much attention as the conceptually simple and surprisingly powerful Naïve Bayes classifier, which is the focus of this work.

The task of text classification is to assign a document to a particular *class*, that is, to categorize it. Assuming each document that one may encounter belongs to some category (potentially including a "catch all" or "don't know" category), supervised text classifiers attempt to learn their trade from a set of training examples [9, 10].

Several problems immediately appear when using a linear text classifier. The accuracy, while high, is still not high enough (at around 80% in many cases) to be used to protect children from adult content, for example. Techniques that can boost the accuracy (like Support Vector Machines) often take longer to train [7], though this has been addressed to some extent in [8].

Rare data problems also abound, since new documents almost always have terms that the classifier has never seen before. At the opposite end of the spectrum, many suffer from data overload, giving the most weight to words that are not very interesting, like stop words or structural information common to all HTML documents.

Despite these difficulties, it is possible, especially when dealing with a very domain-specific application like adult content filtering, to do much better than average even with a simple classifier like Naïve Bayes.

3 Bayes' Law and Classification

The classification problem is one of optimization: given the potential categories for a particular document, which one is most correct? In this setting, it is natural to ask the question, "Given the current document, what is the probability that it belongs to class c?" Given a set of such probabilities, the class assignment algorithm becomes one of selecting the category whose probability is highest for the document in question. These probabilities, denoted $p(C_i|d_t)$, are generally not available and must be estimated. Enter Bayes' Law:

$$p(C_i|d_t) = \frac{p(d_t|C_i)p(C_i)}{p(d_t)} = \frac{p(d_t|C_i)p(C_i)}{\sum_{j=1}^{N} p(d_t|C_j)p(C_j)}. \tag{1}$$

Bayes' Law allows us to calculate the desired probability using parameters that may be derived from the training data. Typically, when using this approach to classify documents, some simplifying assumptions are made about the document and the class in question, since even calculating the right side of (1) can be intractable due to complicated dependency structures. Naïve Bayes assumes several key things:

- A document is defined by the words it contains, independent of its structure.
- The probability of each word is independent of all others in the same category.

These assumptions serve to simplify the formula. Thus, (1) becomes

$$p(C_i|d_t) = \frac{p(C_i)\prod_{j=1}^{W_t} p(w_j|C_i)}{\sum_{k=1}^{N} p(C_k)\prod_{j=1}^{W_t} p(w_j|C_k)}. \tag{2}$$

The probabilities on the right side are trivially calculated using word frequency counts, explained in more detail below.

It is worth noting that the above represents the "conditional independence" Naïve Bayes model, which is only one way of approaching it. Other methods include different statistical distribution models, like *multinomial* and *negative binomial*, among others [2, 5]. Each has its strengths and weaknesses, but the formulation given in (2) will be the chief focus of this work, mainly because of its simplicity.

Substituting word frequencies for probabilities gives us

$$p(C_i|d_t) = \frac{\frac{W_{ci}}{W}\prod_{j=1}^{W_t}\frac{W_{j|ci}}{W_{ci}}}{\sum_{k=1}^{N}\frac{W_{ck}}{W}\prod_{j=1}^{W_t}\frac{W_{j|ck}}{W_{ck}}}. \tag{3}$$

W_t is the number of words in document d_t, W_{ci} is the number of words in category C_i, and $W_{j|ck}$ is the number of times word w_j appears in category C_k. The implementation of (3) is relatively simple (in comparison with some other approaches), and the training time is linear in the number of documents, two characteristics that add to the appeal of the approach [8].

The independence assumption, while severe, does not hurt the classifier as much as might be expected. Much work has been done both to clarify the reasons for this and to make use of such information to improve the method. Some improvements include using additional structural information, attempting different models (multinomial, negative binomial, and others), and noting similarities between Naïve Bayes and Support Vector Machines [2–4,8,12]. Additionally, some abandon Naïve Bayes altogether in favor of other algorithms [7–10].

4 Other Methods and Issues

A particularly interesting problem is that of feature extraction, i.e., finding the appropriate words in the document for use in classification. Some words will be better indicators than others, and if care is not taken, the wrong words can tend to skew the results (e.g., stop words like "the" and "and" which usually have a high frequency in all documents). Some work has been done by [2] in this area, among others.

As has been suggested, other text classification methods exist. One popular approach is to treat words as dimensions in a vector space, making each document a vector in that space [7, 8, 10, 12, 13]. Document classification is done by searching for nearby training vectors and assuming that proximity in the word space is the same as similarity of class. This approach will not be discussed in any detail here, though the idea has some merit. For one thing, it loosens the severe independence assumptions of other linear classifiers by allowing multiple words to contribute to vector similarity.

Much more could certainly be said of current and past work that is relevant to the topic at hand. It is, in fact, difficult *not* to find information on Naïve Bayes and other text classification algorithms. The topic of text classification has been of interest for several decades, and interest appears to be accelerating. (For an excellent survey of available methods, see [9, 10].)

5 Implementation

This section describes some of the implementation choices and issues that were encountered along the way. Implementing Naïve Bayes was a good way to learn of its strengths and weaknesses, and of the issues encountered when dealing with any classifier based on word frequencies.

5.1 The Corpus

The corpus used is the Open Directory Project (*http://www.dmoz.org*), which is a very large human-edited directory of web sites. The link and category information is downloadable in a convenient format, and a large number of the pages are still available online. Of the available URIs, 516,278 were used as training and test data. Of those, 98,446 are considered to be in "adult" categories, and the rest in various others (news, mail, sports, etc.).

The corpus, while large and human reviewed, contains a substantial number of errors. Many sites that are marked as benign actually contain explicit language and material[1]. This is due to the fact that this particular corpus is edited by volunteers, and the barrier of entry is not very high. Thus, some volunteers appear to be gaming the system by surreptitiously inserting incorrect data.

Another shortcoming of the corpus is the fact that only one category is assigned to each web page. Many sites fall within multiple categories and thus need to be assigned to multiple categories. It makes sense that there should be some degree of belief associated with each assignment, as well. Such a corpus does not appear to exist.

Despite its shortcomings, the sheer volume of data afforded by the corpus is a great advantage. Even with the noise, there is enough good information to make it useful as a training set.

5.2 Tokenization

Tokenization is central to statistical classification and greatly affects its performance. This was a hard problem to solve for several reasons, and the ultimate solution is not perfect.

One problem is that words may contain punctuation, like the word "don't" or the number "$50,000.00". Determining whether to strip or use the punctuation as part of a word is a thorny problem, and I discovered that it was much easier to just throw a few mostly-accurate heuristics at it than to solve it outright.

Also, in this application, text is embedded inside of HTML structure. Web browsers are notoriously forgiving of this structure, making it somewhat difficult to extract the text correctly. For example, to make a `<` sign in a document, you are supposed to enter the code `&lt;`. Otherwise it is interpreted as a start of tag character. Even so, browsers are forgiving if it appears outside of a tag and is followed by a space, among other exceptions. Similar issues crop up with the use of JavaScript and other meta information in the document.

Throwing this information away, however, is not a good idea, because often the meta information (specifically that relating to keywords for search engines, alternate names of images, etc.) provides some of the best indication of what the document is [1]. It is possible to have a document with very little human-readable text that still has a lot of giveaways as to what it is (like the title of the page).

These and other issues that were raised highlighted an important point: tokenization is a difficult problem. A substantial amount of time was spent attempting to find a good tokenization algorithm. Tokenization is somewhat critical, since it provides the classifier with the features it will use.

[1]Only web site text was used in this project, and the vast majority of it was never viewed by human eyes. The debugging of tokenizers and other tools was done with benign files, and other classification bugs were resolved through randomized lists of words. This corpus was chosen because I have access to a great deal of data on it from my outside job, allowing me to focus on algorithms rather than data collection.

In the end, a fairly simple approach served well enough. All words are converted to lower case (avoiding some simple rare-data problems), and anything that is not a digit, letter, a $ or # is treated as whitespace. The scheme is not perfect, but it appears to work fairly well for this particular application. Perhaps a better approach would be to try to extract tokens from sentences in the same way that a statistical parser would, but that seemed to be overkill for a class project.

5.3 Naïve Bayes Implementation

Revisiting (3), and making use of the fact that only two categories (adult and not adult) are of interest, we have:

$$\begin{aligned} p(C|d_t) &= \frac{\frac{W_c}{W}\prod_{j=1}^{W_t}\frac{W_{j|c}}{W_c}}{\frac{W_c}{W}\prod_{j=1}^{W_t}\frac{W_{j|c}}{W_c} + \frac{W_{\neg c}}{W}\prod_{j=1}^{W_t}\frac{W_{j|\neg c}}{W_{\neg c}}} && (4)\\ &= \frac{W_c^{(1-W_t)}\prod_{j=1}^{W_t}W_{j|c}}{W_c^{(1-W_t)}\prod_{j=1}^{W_t}W_{j|c} + W_{\neg c}^{(1-W_t)}\prod_{j=1}^{W_t}W_{j|\neg c}} && (5)\\ &= \frac{1}{1+\left(\frac{W_{\neg c}}{W_c}\right)^{(1-W_t)}\prod_{j=1}^{W_t}\frac{W_{j|\neg c}}{W_{j|c}}}. && (6)\end{aligned}$$

5.3.1 Computer Precision

While (6) is easy to implement directly, several problems arose in the implementation that made it impractical. For example, the products in the denominator and the powers of word frequency ratios can cause serious floating point precision errors in a computer. In fact, for documents containing more than a handful of words, the numbers become too big (or small) to get accurate results of any kind. Consider that W_t is the total number of words in a test document, W_c and $W_{\neg c}$ represent the total words in two categories. The ratio of these taken to the indicated power must be very accurate for the final outcome to truly be a useful probability. In practice, implementing this equation directly caused precision errors that tended to transform the entire product into 0.0 or $+\infty$ with very little in between.

One may ask why the derivation was taken all the way to (6) in the first place. It turns out that this form is extremely convenient for another transformation, this time to help the computer with accuracy. If we let $e^{\lambda} = \left(\frac{W_{\neg c}}{W_c}\right)^{1-W_t}\prod_{j=1}^{W_t}\frac{W_{j|\neg c}}{W_{j|c}}$, then

$$\begin{aligned} \lambda &= \ln\left[\left(\frac{W_{\neg c}}{W_c}\right)^{(1-W_t)}\prod_{j=1}^{W_t}\frac{W_{j|\neg c}}{W_{j|c}}\right] && (7)\\ &= (1-W_t)\big(\ln W_{\neg c} - \ln W_c\big) + \sum_{j=1}^{W_t}\big(\ln W_{j|\neg c} - \ln W_{j|c}\big). && (8)\end{aligned}$$

In practice, the logarithms yield much more reasonable intermediate numbers than we would otherwise have. Equation (6) thus becomes

$$p(C_i|d_t) = \frac{1}{1 + e^{\lambda}} \tag{9}$$

where λ is as given in (8). The greater intermediate accuracy of the calculations lends itself to more accurate probabilities, and therefore a more reliable classifier.

While implementing (9) and (8), several other issues cropped up that perhaps should not have been surprising, but that were just the same. First, it is possible for some of the word counts to be 0. This causes the logarithms to approach $-\infty$, which is not terribly useful. Even so, it is necessary to represent words that appear in one place but not in the other, as these are likely to be good indicators. To fix this, all word counts were incremented by a small value: 0.01. This kept the logarithm routines happy and did not affect the accuracy of non-zero counts in any measurable way.

5.3.2 Thresholding

Since we are dealing with only two categories we need only calculate the probability that a particular document belongs to one of them, since $p(\neg C|d_t) = 1 - p(C|d_t)$. In other words, we compute $p(C|d_t)$ and then compare it against a fixed standard. If it passes, d_t is deemed to belong to class C. The threshold is easily determined by noting that the probability for one class must be greater than the other in order to be selected:

$$\begin{aligned} p(C|d_t) &> p(\neg C|d_t) = 1 - p(C|d_t) && (10) \\ 2p(C|d_t) &> 1 && (11) \\ p(C|d_t) &> 0.5. && (12) \end{aligned}$$

Thus, we should be able to simply compute (9) for one category (adult) and test against a fixed threshold of 0.5. This fits intuition and is easy to implement.

As will be discussed later, this formulation also allows us to set a dial on how conservative we wish the algorithm to be. While the mathematically correct threshold is shown to be 0.5, there is no reason that this threshold could not become a tunable parameter in the algorithm. Setting it higher would indicate that we are more conservative (require more evidence to make a decision) about placing documents into class C, while a lower value would indicate a greater readiness to consider less evidence before deciding on class C.

5.3.3 Multiple Classes

A note is in order regarding problems with more than two classes. The simplifications and precision optimizations made above only apply to a two-class problem. This does not destroy generality, however. It is possible to use the same approach for more than two classes by allowing documents to belong to more than one class at a time, with differing probabilities. Thus, the decision as to whether a document belongs in a particular class becomes a binary one: does this document belong to class C_i or class $\neg C_i$?

Figure 1: Distribution of documents for each confidence level. The non-adult documents should have a large peak near 0.0, and the adult documents near 1.0.

Looking at the problem in this way allows classification to become more flexible. Not only can a document belong to multiple classes, it is also possible for it to belong to *no* class, indicating that we truly do not know what to do with it. In this way, the simplifying assumptions and precision optimizations actually *add* flexibility to the approach.

6 Experiments

After implementing a Naïve Bayes classifier, several experiments were performed as a way of exploring its parameters and behaviors. The corpus used has already been discussed, and consists of several hundred thousand documents, some classified as adult, some classified in other (more benign) ways.

In all experiments here, the training set consisted of only 9,845 adult documents and 54,947 other documents. The uneven distribution of adult to non-adult documents made for some interesting behavior and will figure into the analysis of the results.

The above numbers represent only 10% of the entire corpus. While this seems like a small amount, the data was representative enough to perform meaningful experiments, and had the added benefit of not consuming insane amounts of RAM and taking a long time to load. Accuracy did not change noticably with extra data, so all results here will focus on this 10% of the corpus used for training.

6.1 Accuracy Over the Training Data

The accuracy of Naïve Bayes in this domain is surprisingly high at 98.9%, but this is not representative of what is really happening. The following table shows the relationship between class and accuracy.

	Classifier Output	
True Class	ADULT	NOT
ADULT	0.170	0.009
NOT	0.002	0.819

The diagonals represent correct classifications, and the off-diagonals represent false negatives and positives (top and bottom). Taking the accuracy over positive examples and negative examples separately, we have:

True Class	Accuracy
ADULT	0.94972
NOT	0.97564

Figure 2: Distribution of documents for each confidence level. The non-adult documents should have a large peak near 0.0, and the adult documents near 1.0.

As you can see, the accuracy for documents belonging to the adult category is lower than for those that do not. A distribution graph for the two categories is shown in Figure 1. Note that for the most part, the categories are well separated. Documents belonging to the adult class show up on the far right, and documents not belonging to it show up on the far left. Zooming in, however, we see that the classification is not perfect. There is a bit of overlap in the classification, especially at the extremes of the confidence space. This is where the false negative and false positive numbers come from.

Some spot checking of the data confirmed that the picture is not as bleak as it seems. Some of the data was simply mislabeled, and the classifier labeled it correctly. This is one of the features of Naïve Bayes and other similar methods: it is robust in the face of noisy training data.

6.2 Accuracy Over the Test Data

The accuracy over the chosen test data was still fairly high, coming in at 97.2%, but lower than that for the training set. The breakdown of class and accuracy is given below.

	Classifier Output	
True Class	ADULT	NOT
ADULT	0.158	0.023
NOT	0.005	0.814

Again, the diagonals represent correct classifications, and the off-diagonals represent false negatives and positives (top and bottom). Taking the accuracy over positive examples and negative examples separately, we have:

True Class	Accuracy
ADULT	0.87293
NOT	0.99389

The accuracy suffered more for the test set than it did for the training set, which is to be expected. The accuracy for the non-adult documents went up, however, which was somewhat surprising. Some spot checking of words in each document revealed some interesting information. The documents in the adult class contained many words that had not been seen before, and each of these contributed a low probability for the adult class by default. Thus, the false negative rate was higher in the test data than in the training data.

As in the previous section, the confidence overlap is shown in Figure 2. The overall distribution of documents looks roughly the same. Nothing jumps out as being extremely far off.

Figure 3: The word counts for each confidence level are shown for the adult class. Note that the graph is cut off at 10,000 words even though the leftmost bar extends to nearly 1,000,000 and the rightmost bar extends to nearly 100,000.

(a) Overall accuracy

(b) False positives

(c) False negatives

Figure 4: Accuracy information for different values of k

6.3 Words That Are Interesting

That some features of a document are more useful than others is well known in the statistical classification community [9, 10, 12, 13]. In fact, for some algorithms it is essential that some features be removed to perform dimensionality reduction (kNN, ANN). It seems reasonable that some feature selection might help Naïve Bayes.

Consider Figure 3. It shows the confidence level indicated by each possible word taken alone. This is somewhat different than the actual Naïve Bayes approach, where word probabilities are combined to generate a classification confidence. The graph shows that some words are more "interesting" than others. The left and right sides have bars that extend far beyond the range of the graph (1 million on the left and 100,000 on the right), representing the words that tend to be the best at distinguishing between categories.

The peaks near the middle of the graph, where the confidence level is 0.5, represent words sitting right at the threshold, or words that are not very good at class disambiguation. For example, words like "#ee00ee" (an HTML color), "abstract", "blather", "the", "and", and many others tend to appear equally frequently in either kind of document. This makes them poor candidates for disambiguation.

To test this theory, I tried running the classifier using only k words of each document that had the highest "utility", that is, that maximized $u = |p(C|w) - 0.5|$. The results of that test are shown in Figure 4(a). The figure, unfortunately, does not tell the entire story. While the overall accuracy improved, the true performance is better viewed by looking at false positives and false negatives separately, as in Figures 4(b) and 4(c).

(a) Over-
all
ac-
cu-
racy

(b) False
pos-
i-
tives

(c) False
neg-
a-
tives

Figure 5: Accuracy for different threshold values

It is interesting the the false positives went down, but the false negatives went up while the accuracy increased overall. The fact that the false negatives increased is partly due to the nature of adult web pages. They generally contain a lot of links and image tags, and very little text. The text that would tip off a Naïve Bayes classifier is nowhere near as plentiful as the benign text in the document. As more words are considered, more of the benign words will weigh in against the heavy indicators.

Several things are of note here. One is that the accuracy was surprisingly high when using just one word to classify the document. It appears that this occurs because benign words tend to appear in both categories, but other indicator words generally appear in only one category. Thus, the word that best classifies a document (furthest from neutral) is usually an indicator for the adult class, rather than for another class. In this particular application, the choice of Naïve Bayes is a very good one.

Another item of note is the fact that the accuracy reaches very high levels with just a handful of words from the document. This indicates to me that many of the words in the training set are not usually much use. It may be that a large portion of the training data could be culled as an optimization, though more research is probably needed to both determine how to do this and verify that it is a valid idea.

Finally, the results presented in Figure 4 were obtained with a threshold of 0.5. It is possible that a more conservative threshold be used for this application, since false positives are generally more acceptable than false negatives when an algorithm of this nature is, for example, protecting children at school. Lowering the threshold for classification represents a more conservative view of what is adult and what is not, and the results of that exercise are shown in Figure 5.

It is of note that the only metric to suffer as a result of an increasing threshold was "false negatives". Indeed, the fact that the confidences are so well separated (Figures 1 and 2) is probably responsible for the surprising robustness of the algorithm in the face of a changing threshold.

7 Conclusions and Future Research

It appears that Naïve Bayes is an excellent classification approach for adult content filtering on the web, at least with things the way they currently stand. Its inherent simplicity and ability to classify with very little information make it a good choice. Certainly as classifiers of this kind become more popular, the content on nefarious sites will adapt to thwart it, but for now this approach works very well.

Several results obtained in this project were surprising to me, among which were the very small number of words needed to classify a document and the difficulty in implementing an algorithm that requires high precision. Actually implementing the algorithm and using it on real data has been a very interesting exercise.

It was driven home that obtaining training data is often the hardest part of an undertaking of this nature. The data, while plentiful, is not perfect, and working to make it more perfect would be not only a major undertaking, but one that would probably leave me feeling rather dirty. Using a statistical method that is robust in the face of noise is therefore essential to a project of this kind.

Several things jump out as needing some attention. First, the independence assumption of Naïve Bayes is very strong, and though it works pretty well, is sometimes too strong to catch subtleties and inferences in a document. It would be good to explore some statistical approaches to adding dependencies, ideally some that would do so without unduly complicating the model or exploding the data (like n-grams would).

Another interesting area to pursue is that of including document structure in the training data [1]. Currently all words are treated equally, be they keywords, image names, or parts of the legible text. That some words have different locations than others may be useful as part of the feature set.

Finally, an exploration of other methods like Support Vector Machines [7] and Neural Networks [9, 10, 12, 13] as classifiers is bound to be interesting.

References

[1] K. K. R. Arul Prakash Asirvatham, "Web page classification based on document structure." http://citeseer.nj.nec.com/asirvatham01web.html.

[2] J. D. M. Rennie, "Improving multi-class text classification with naive bayes," Master's thesis, Massachusetts Institute of Technology, 2001.

[3] S. E. Department, "On the naive bayes model for text categorization." http://citeseer.nj.nec.com/544463.html.

[4] L. Jensen and T. Martinez, "Improving text classification by using conceptual and contextual features." http://citeseer.nj.nec.com/jensen00improving.html.

[5] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[6] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, pp. 61–67, 1999.

[7] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proceedings of ECML-98, 10th European Conference on Machine Learning* (C. Nédellec and C. Rouveirol, eds.), no. 1398, (Chemnitz, DE), pp. 137–142, Springer Verlag, Heidelberg, DE, 1998.

[8] K. Shih, Y. han Chang, J. Rennie, and D. Karger, "Not too hot, not too cold: The bundled-svm is just right!." http://citeseer.nj.nec.com/shih02not.html.

[9] R. A. Calvo and H. A. Ceccatto, "Intelligent document classification." http://citeseer.nj.nec.com/calvo00intelligent.html.

[10] K. Aas and L. Eikvil, "Text categorisation: A survey," tech. rep., Norwegian Computing Center, June 1999.

[11] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka, "Training algorithms for linear text classifiers," in *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval* (H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, eds.), (Zürich, CH), pp. 298–306, ACM Press, New York, US, 1996.

[12] Y. Yang and X. Lui, "A reexamination of text categorization methods," in *Proceedings of the 22 nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, (USA), pp. 42–49, 1999.

[13] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1/2, pp. 69–90, 1999.