

Document Preprocessing For Naive Bayes Classification and Clustering with Mixture of Multinomials

Dmitry Pavlov, Ramnath Balasubramanian, Byron Dom, Shyam Kapur, Jignashu Parikh
 Yahoo Inc., 701 First Avenue, Sunnyvale, California 94089
 dpavlov,ramnath,bdom,kapur,jignashu@yahoo-inc.com

ABSTRACT

Naive Bayes classifier has long been used for text categorization tasks. Its sibling from the unsupervised world, the mixture of multinomial models, has likewise been successfully applied to text clustering problems. Despite the strong independence assumptions that these models make, their attractiveness come from low computational cost, relatively low memory consumption, as well as ability to handle heterogeneous features and multiple classes. Recently, there has been several attempts to improve the accuracy of Naive Bayes by performing heuristic feature transformations, such as IDF, normalization by the length of the documents and taking the logarithms of the counts. We justify the use of these techniques and apply them to two problems: classification of products in Yahoo! Shopping and clustering the vectors of collocated terms in user queries to Yahoo! Search. The experimental evaluation allows us to draw conclusions about the promise that these transformations carry with regard to alleviating the strong assumptions of the multinomial model.

Categories and Subject Descriptors: I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms, Experimentation.

Keywords: Classification, clustering, data transformations, performance, Naive Bayes, mixture of multinomials.

1. INTRODUCTION

Naive Bayes classifier has been a subject of multiple research studies [2, 5] and almost on every occasion it was noted that the Naive Bayes' performance can be jeopardized by several independent problems. These include the imbalance of training documents in classes, mismatch between the actual text distribution and the multinomial distribution assumption, double counting of evidence. More sophisticated SVMs and maximum entropy classifiers typically outperform Naive Bayes. However, this comes at the expense of high, if at all affordable, computational cost.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'04, August 22–25, 2004, Seattle, Washington, USA.
 Copyright 2004 ACM 1-58113-888-1/04/0008 ...\$5.00.

As an example, consider the data we use in Yahoo! Shopping for categorizing the products. Each product, that is currently active on Yahoo! Shopping site, is represented as a bag of terms. After straightforward frequency-based feature selection, this leads to a total of slightly over 200,000 individual feature terms. The total number of classes equals about 60 at present and is expected to grow. This by itself represents a challenge for SVMs, as they do not have an efficient way of handling the multiclass problems. One-against-others solution is going to “enjoy” even more severe problems with imbalanced classes than Naive Bayes. Error-correcting output coding [1] remains a viable option, but in any case the computational penalty of training a single SVM is going to be multiplied by at least the number of classes, i.e. possibly grow 1 to 2 orders of magnitude in our case. But still by far the main problem for SVMs comes with the number of training data points, which are in our case products, constantly growing in number. The most advanced SMO algorithm [9] has at least quadratic complexity in the number of data points per iteration. There has been work on speeding up SVM training includes various forms of data sampling, boosting [7] and squashing [8]. However, in our experience handling multiclass problems is still an issue while the preprocessing steps above can easily result in either inaccurate classifier (e.g., sampling) or are too computationally expensive (e.g., squashing).

Maximum entropy classifiers are not going to have as big a problem handling multiple classes as SVMs, but they are probably even slower than SVMs if trained with a commonly used generalized iterative scaling [4]. There has been also work on speeding up the training of maximum entropy models. However, even with these speed ups, the algorithms are still far from the simplicity and speed of naive Bayes. Thus, the advertised superiority of SVMs and maximum entropy classifiers over Naive Bayes generally comes with a bouquet of hard-to-deal-with problems, and from this perspective Naive Bayes classifier looks quite attractive, especially if some of its drawbacks can be alleviated.

This was exactly the goal of several recent studies [11, 10]. The first two papers discuss approaches with introducing dependencies between features and performing clustering prior to classification. These methods are quite a bit more complex than the regular Naive Bayes and, although valid, are not considered here. The last paper by Rennie et. al. [10] looks appealing to us as the only manipulations authors perform are on the data itself, and the Naive Bayes classifier can still be learned in one pass through the data. The authors illustrate various problems that Naive Bayes suffers

from and propose heuristic ways of handling them:

1. *Systemic Problem 1*: imbalance between the number of training examples per class, resulting in poor choice of weights for linear decision boundary. Proposed solution consists of using data from and only from classes complementary to class c to estimate model parameters for class c .
2. *Systemic Problem 2*: feature independence assumption, resulting in heavier class weights for classes with stronger dependencies between features in the data. Proposed solution consists of the weight normalization.
3. *Other Problems*: mismatch between multinomial assumption and power law distribution of frequencies. Proposed solution consists of various data transformations, including
 - TF, used to obtain less “power-law-like-looking” distribution by replacing the counts by their logarithms incremented by 1;
 - IDF, used to discount non-discriminative features by multiplying their counts by the inverse frequency in the training examples; and
 - length normalization, used to avoid the domination of the shorter documents in the parameter estimates by putting the documents on the surface of the unit sphere.

In this paper, our main goal is to study the effects of data transformations on text classification with Naive Bayes classifier and text clustering with multinomial mixture models. In Section 2 we briefly define the models. In Section 3, we implement the proposed heuristics on the Yahoo! Shopping data and perform a comparison of the plain Naive Bayes classifier and various combinations of heuristics. We confirm that some of the transformations above are indeed helpful for boosting the accuracy of the Naive Bayes classifier. In Section 4 we turn to the problem of unsupervised learning of the multinomial mixture model on the Yahoo! User query logs, where the problem is to identify similar terms by studying their collocation vectors with other terms. We apply the multinomial mixture model to this problem and illustrate the effects of the data transformations above for this learning task. We find a single data transformation that appears to be highly helpful in identifying semantically similar terms. We propose an explanation to this phenomenon and draw conclusions in Section 5.

2. MODEL DESCRIPTION

We only present brief model descriptions here, for a more detailed discussion please refer to [3] for Naive Bayes classifier and to [6] for multinomial mixture models.

We assume that the data consists of documents, each represented as a bag of words, i.e. as a map from the set of terms occurring to their count in the text. The length of the document is defined as a sum of counts all terms extracted from its text. The set of all terms is referred to as alphabet A , with size $|A|$. For purposes of classification the document is then assigned a unique class label from a finite set of labels C . The assumption then is that the terms in every class (in classification) or cluster (in clustering) were generated by a multinomial mixture model:

$$P(t_1, \dots, t_n | D) = \sum_k \left(\prod_i \theta_{ik}^{n_i} \alpha_k \right), \quad (1)$$

where t_i is the i -th term and n_i is its count in the current document D , α_k is the document-independent prior probability of the class (cluster) k , and θ_{ik} is the probability of the observing term i in the class (cluster) k . Model parameters also satisfy the normalization conditions.

Given the data, i.e. the set of N documents $\{D\}$, one can formulate a maximum likelihood or maximum a-posteriori parameter estimation problem, the optimization of which leads to the following set of equations for classification:

$$\theta_{ik} = \frac{\beta_k + f_{ik}}{\beta_k |A| + \sum_i f_{ik}}; \quad \alpha_k = \frac{\gamma + f_k}{\gamma |C| + N}, \quad (2)$$

where f_{ik} is the frequency of term i in the documents of class k , f_k is the frequency of documents of class k in the data and β_k and γ are two user-defined smoothing parameters (0 value corresponds to no-smoothing maximum likelihood parameter estimates).

Clustering is going to involve similar equations but is going to be iterative (EM algorithm) and involve an additional estimation of the class posterior ($P(k|D)$):

$$\begin{aligned} P(k|D) &\propto \prod_i \theta_{ik}^{n_i, D} \alpha_k; \\ \theta_{ik} &= \frac{\beta_k + \sum_D P(k|D) n_{i, D}}{\beta_k |A| + \sum_{i, D} P(k|D) n_{i, D}}; \\ \alpha_k &= \frac{\gamma + \sum_D P(k|D)}{\gamma |C| + \sum_{k, D} P(k|D)}. \end{aligned} \quad (3)$$

Thus, there is much similarity between the classification and clustering approaches based on the multinomial model, with the main difference being that for clustering class labels needs to be predicted and each document can potentially contribute to the estimates of model parameters for a given class. The reason why mixture of multinomials induces a soft clustering over the documents, is that once the model is fit to the data, for each document one can compute the class-posterior distribution $P(k|D)$ as in Equation 3 and interpret the result as a probability with which a given document D belongs to each of the $|C|$ fit clusters.

3. CLASSIFICATION OF Y! SHOPPING DATA

We ran a series of experiments with different data transformations preceding Naive Bayes fitting on active Yahoo! Shopping data. We used all labeled products in Y! Shopping as of February 7, 2004. We parsed, tokenized and stemmed the title and description from every product, so that after aggregating the counts of same tokens a product is represented as a “bag of tokens”. The total alphabet size for the data at hand was about quarter of a million and the total number of products (retained after straightforward frequency-based feature selection) was a little under 100,000. The total number of categories used for classification was 58.

We ran a 20-fold cross-validation experiment for each of the normalization choices listed in the schema of Table 1 under several heuristic ways of changing the parameters of the decision boundary outlined in [10] and listed in the row

Table 1: Average cross-validated accuracy on the test data for Yahoo! Shopping, various data normalizations (columns) and heuristic changes to the parameters of the decision boundary (rows) are used. Best value in each row is bolded, best overall value is additionally boxed. WR stands for Weight Renormalization, CCF stands for Complementary Class Fitting. All numbers are percents.

	Raw	TF	IDF	norm	TF + IDF	TF + norm	IDF + norm	TF + IDF + norm
As Is	80.66	79.84	80.86	78.06	81.22	78.37	77.73	78.19
WR	79.93	78.73	82.82	76.75	82.17	76.60	76.29	76.18
CCF	79.47	79.22	80.13	78.93	80.06	78.71	78.90	78.79
WR + CCF	79.58	79.28	79.83	78.64	79.94	78.51	78.28	78.17

annotation of Table 1. We analyzed both data transformations and heuristics in the introduction Section 1. In each row of the Table we computed the best accuracy and put it in bold font. The best accuracy from all runs is additionally boxed.

Notice that out of 8 possible ways of transforming the data and across 4 heuristics, the most promising are either doing IDF on its own or in combination with TF transform. The best heuristic appears to be weight renormalization, advertised to alleviate the feature independence assumption. The overall boost in accuracy resulting from the use of IDF and weight renormalization (82.82%) compared to the raw Naive Bayes (80.66%) is a significant 2.2%. Using the complement class heuristic on its own or together with weight renormalization failed to improve the performance, and for certain data transformations even made it worse than for the raw Naive Bayes classifier. The same conclusion applies to length normalization, that masked the effect of doing TF/IDF transforms on our data. We revisit this phenomenon in Section 4, where the length normalization exhibited the opposite behavior, turning out to be the only data transformation that resulted in creating semantically meaningful clusters on terms extracted from Yahoo! user query logs.

Overall, we conclude that some of the basic data transforms are capable of significantly boosting the accuracy of the Naive Bayes classifier, and as such should always be experimented with in real-world applications.

4. CLUSTERING OF Y! USER QUERY LOGS

As was mentioned in Section 2, the classification and clustering approaches based on the multinomial model are very similar, thus before we started off experimenting with the query logs, our intuition was that we would find yet another proof that data transformations are of little help. This, however, proved to be incorrect.

The data we had at hand consisted of all user queries on main Yahoo! Search site accumulated over a period of one week. Each unique query was preprocessed by removing punctuation, special symbols followed by the identification of meaningful n -grams. In what follows, each retained n -gram is referred to as *unit*. Then for each unit U , we identified all other units U_1, \dots, U_k , with which U co-occurred in at least one query in the query corpus and recorded the co-occurrence counts of U and U_i in the data vector corresponding to unit U . We further discarded all units whose total co-occurrence count (similar to the length of the text document) is below 10. This led to a total of 22,276 units in the space of 132,481 collocated units.

The goal was to cluster the units into semantically re-

lated groups by exploiting the similarities between the collocations of units. To see why this is possible, consider the sport brands Adidas, Nike, Puma etc., all of which are likely to co-occur in queries with the same set of units, such as “sports”, “jogging”, “ski”, or “catalogue”. Similarly, the airport names LAX, JFK, Heathrow etc., are likely to co-occur with other distinguishing terms, such as “travel”, “parking”, “flight” and others. Some of the units can have certain affinity to multiple clusters, such as, for instance, Amazon would belong to “Internet Shopping Portal” and “River” clusters.

We started off by fitting a mixture of multinomials to the raw count data and soon discovered that clusters are really strange and non-intuitive, specifically all seemingly similar objects, such as airport names, celebrity names, city names etc., were spread across multiple clusters. To study this effect we concentrated on the following 3 units: “barn”, “barns” and “tack” and tried fitting a two-component mixture model to the vectors of collocations for them. The data for this experiment is given in Table 2. Our intuition was that “barn” and “barns” should have been placed together, and “tack” alone in a separate cluster. Analyzing the data reveals that “barn” and “barns” have at least 7 high-frequency common collocated units (compared to only 2 common unit between “barns” and “tack”) and should indeed be together. However, we consistently obtained a highly confident response that instead “barn” was placed alone in a separate cluster.

The key to a solution lied in understanding that shorter¹ documents or sets of collocations have considerably more influence over parameter estimates than the longer ones. Indeed, the multinomial model prescribes to normalize the components of θ vectors to sum to 1. Thus, the shorter documents take over the longer ones simply because their count profile looks on average higher, and consequently more likely, after a “sum to 1” normalization. When the length of the document is much higher compared to the other documents (as for “barn”), whether due to a single peaking feature, such as “pottery” in “barn”, or due to just many co-occurring features, this document appears essentially to be highly unlikely to have been generated by **any** cluster. Moreover, adding this document to a cluster already containing a shorter document will reduce the likelihood of this shorter document. To support this finding, we evaluated the log-likelihoods of the data consisting of “barn”, “barns” and “tack” under all different ways of clustering them into two clusters with a multinomial mixture. The results of this experiment are reported in Table 3. Notice, that on the raw data, placing “barn” on its own in a separate cluster has more incentive than grouping “barn” and “barns” despite

¹in a sense of total count

Table 2: Data for a 3 term “barn”, “barns” and “tack” example. Shown in bold are the common terms between at least two of the terms. Followed by each collocated term is the co-occurrence frequency, some of the collocated terms with low frequency are omitted.

barn: pottery (18131), the (1343), dress (1152), red (832), pole (617), owl (580), burning (486), plans (451), white (417), horse (356), swallow (267), old (256), door (231), furniture (201), yard (200), sale (199), theater (189), auto (172), boot (171), pictures (150), tire (150), wood (148), designs (144), bike (134), kits (132), homes (117), apple (111), builders (97), house (92), building (88), cycle (87), angus (83), liquor (81), nursery (79), dance (76), puppy (75), bargain (74), car (74), carpet (71), dinner theater (69), flower (66), comedy (65), metal (63), candle (61), booty (61), construction (60), dairy (60), oak (59)
barns: pole (667), horse (358), noble (157), old (147), for sale (108), storage (86), pottery (76), metal (72), pictures of (65), yard (54)
tack: horse (1025), shop (527), stateline (280), tick (152), western (124), stores (110), for sale (93), hard (83), trunk (79), used (72), english (72), pony (53), auction (53)

Table 3: The data log-likelihoods under a multinomial mixture model for all possible ways of hard-partitioning 3 terms “barn”, “barns” and “tack” into 2 clusters. Note that normalization favorably affects the choice of the clustering.

Cluster Partitioning	Log-Likelihood of Raw Data	Log-Likelihood of Length-Normalized Data
barn + barns + tack VS \emptyset	-92407	-21.06
barn + barns VS tack	-83537	-19.58
barn + tack VS barns	-87325	-20.04
barns + tack VS barn	-80645	-19.90

of their high similarity in the unit space. Further notice, that length normalization helped to get things back to the expected behavior.

This domination effect is cited as a primary reason for using length normalization in [10], as it results in equating the contributions of every document. Placing the original vectors of collocations of the surface of the unit sphere did fix the problem, on both the 3 unit data set as well as on the full set of units, thus allowing us to claim that length normalization also makes similar documents look more similar. An apparent disconnect of this claim with the results of our classification experiments can be explained by the fact that in manual categorization some of the similar records are sometimes purposefully placed into different classes, in which case the transformation only harms.

Table 4 contains some of the unit clusters discovered from the full data. Notice that we are able to discover semantically coherent clusters. Several comments are in order here:

1. The length normalization is equivalent to considering a set of weighted documents, where the weight of the document is the inverse of its length. In this case, one can show that optimizing the weighted log-likelihood is indeed equivalent to the normalization that we performed upfront on the data, i.e. dividing the count of each unit inside the document D by its length in L_2 .
2. We tried all other transformations advertised in the Introduction, with absolutely no effect, specifically the 3 unit problem above was only solved by employing the length normalization. A possible reason here is that most of the data transformations listed in the Introduction help with *discriminating* between the classes, and not in finding similar documents. Even though one might think that the tasks should be equivalent, i.e. similarity between the documents should imply

their belonging to the same class and vice versa, they are apparently quite different.

3. Smoothing needs to be applied very cautiously as, if overdone, it will tend to make even highly dissimilar documents look similar. In practice, we never used a value of smoothing parameters greater than 10^{-5} .
4. As Table 3 prompts, it is quite easy to end up having suboptimal solution, with a likelihood close to optimal one. In order to attempt achieving the best (highest) likelihood, we performed multiple restarts of EM procedure and picked the best scoring solution.

5. CONCLUSIONS

In this paper, we advocated the use of Naive Bayes classifier for practical multiclass text categorization, citing its virtues, including low computational cost, relatively low memory consumption, ability to handle heterogeneous features and multiple classes, and often competitiveness with the top of the line models. We further performed the experimental evaluation of several data transformations advertised by [10] to help alleviate the restrictive assumption of the multinomial model made by Naive Bayes. Our evaluation has singled out the length normalization as a measure capable of not only equating the influence of every document on the parameter estimates, but also making the similar documents even more similar. This transformation allowed us to perform clustering of units based on the vectors of collocation with other units into semantically related groups. No other transform, as well as fitting the model to the original data, resulted in even remotely as nicely interpretable clusters as with length normalization. Furthermore, in the application of the transform to the Yahoo! Shopping classification data, we confirmed the finding that there could be a discon-

Table 4: Some of the clusters formed by applying the multinomial mixture model to the unit collocation vectors normalized by the length in L_2 . Each unit is followed by the probability that it belongs to this class. Only units whose highest probability score points to the cluster are shown.

logan (1.000), lax (1.000), sfo (0.998), dulles (0.993), o'hare (0.990), islip (0.989), heathrow (0.981), gatwick (0.973), seatac (0.947), ohare (0.918), dfw (0.598), bwi (0.487)
adidas (1.000), customized (1.000), nike (1.000), custom made (1.000), reebok (1.000), suede (1.000), canvas (1.000), dada (1.000), waterproof (1.000), chanel (1.000), mesh (1.000), discontinued (1.000), puma (1.000), stilettto (1.000), north face (1.000), red wing (0.999), converse (0.998), birkenstock (0.998), high heel (0.991), timberland (0.944), new balance (0.854), addidas (0.804), merrell (0.720), ecco (0.622), clarks (0.585), nine west (0.527), payless (0.359)
ronald reagan (1.000), john f kennedy (1.000), john f. kennedy (1.000), jfk (1.000), abraham lincoln (1.000), george bush (1.000), george washington (0.999), thomas jefferson (0.998), theodore roosevelt (0.994), george w bush (0.988), bill clinton (0.976), george w. bush (0.974), winston churchill (0.946), woodrow wilson (0.898), richard nixon (0.808), yearbook (0.502), martin luther king (0.465), saipan (0.364), christopher columbus (0.329)
allergies (1.000), thrush (1.000), poison ivy (1.000), alcohol (1.000), acne (1.000), asthma (1.000), cellulite (1.000), vertigo (1.000), arthritis (1.000), hair loss (1.000), allergy (1.000), constipation (1.000), menopause (1.000), hives (1.000), herpes (1.000), insomnia (1.000), adhd (1.000), diarrhea (1.000), ear infection (1.000), sunburn (1.000), fatigue (1.000), leukemia (1.000), heartworm (0.999), yeast infection (0.997), infertility (0.997), candida (0.995), sore throat (0.995), yeast infections (0.993), bronchitis (0.983), fibromyalgia (0.979), lupus (0.976), high blood pressure (0.966), gout (0.965), heartburn (0.955), epilepsy (0.914), eczema (0.905), multiple sclerosis (0.871), psoriasis (0.640), lyme disease (0.632)
filipino (1.000), japanese (1.000), korean (1.000), swedish (1.000), sweedish (1.000), chinese (1.000), indian (1.000), thai (1.000), kerala (1.000), latin (1.000), pakistani (1.000), malay (1.000), farsi (1.000), indonesian (1.000), iranian (1.000), malaysian (1.000), tamil (1.000), lebanese (1.000), arabic (1.000), russian (1.000), arab (1.000), malayalam (0.999), jap (0.999), bangladeshi (0.996), hmong (0.990), bengali (0.984), taiwanese (0.982), sri lankan (0.981), somali (0.972), telugu (0.969), vietnamese (0.966), gujarati (0.922), irani (0.874), pilipino (0.840), paki (0.812), srilankan (0.772), nepali (0.734), japanes (0.552), cantonese (0.539), hindi (0.535), tantric (0.344)

nect between the similarities within the class and across the document corpus, resulting in more similar document being placed in different classes by manual categorizers. One example when stressing the similarity can hurt could be documents with extremely short description of what the item is and extraneous elaborations on the shipping policy. However, other transforms, that are better tailored to boosting discriminative power, such as TF, IDF and weight normalization heuristic, improved the classifier accuracy by over 2.2%. Other manipulations, including complement class fitting failed to improve performance over the regular Naive Bayes model.

We conclude that the conclusions of the “no free lunch” theorem still hold valid, every specific case and data set call for a separate evaluation and there is no universally best model, even for a given type of data. A diligent researcher can just populate his toolbox with different methods that are reportedly known to work, and use his own judgement as to what approach would be the best in any given case.

6. ACKNOWLEDGEMENTS

We would like to thank Kamal Ali, Pavel Berkhin, Cliff Brunk, Igor Cadez, Darya Chudova and Padhraic Smyth for many productive discussions.

7. REFERENCES

- [1] A. Berger. Error-correcting output coding for text classification. In *IJCAI Workshop on machine learning for information filtering*, 1999.
- [2] Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103—130, 1997.
- [3] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
- [4] I.Csiszar. A geometric interpretation of darroch and ratcliff’s generalized iterative scaling. *The Annals of Mathematical Statistics*, 17(3):1409–1413, 1989.
- [5] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
- [6] G. McLachlan and K. Basford. *Mixture Models*. Marcel Dekker, New York, 1988.
- [7] D. Pavlov, J. Mao, and B. Dom. Scaling-up support vector machines using boosting algorithm. In *International Conference on Pattern Recognition (ICPR)*, 2000.
- [8] Dmitry Pavlov, Darya Chudova, and Padhraic Smyth. Towards scalable support vector machines using squashing. In *Knowledge Discovery and Data Mining*, pages 295–299, 2000.
- [9] J. Platt. Using sparseness and analytic QP to speed training of support vector machines. In *Advances in Neural Information Processing Systems*, 1999.
- [10] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, 2003.
- [11] R. Vilalta and I. Rish. A decomposition of classes via clustering to explain and improve naive bayes. In *Proceedings of European Conference on Machine Learning*, pages 444—455, 2003.