Information Theoretic Text Classification

Amir Aavani[1,2], Amin Farjudian[1], and Mehdi Salmani[2]

[1] Sharif University of Technology, Iran
[2] Sepanta Robotic Research Institute, Iran

Abstract. The assignment of natural language texts to one or more predefined categories based on their content is an important component in many information organization and management tasks. This paper presents an information theoretic approach for text classication problem which we call it ITTC. When classification task is performs over dynamic data or noisy data, ITTC performs better than Bayesian Classier. We have proven that ITTC is theoretically equivalent to Bayesian Classifier. We have used some newsgroups, to evaluate the capability of ITTC. The achieved results shows ITTC outperfomes over Bayesian Classifier.

1 Introduction

With the increase of electronic documents, such as web pages, emails, academic publications, etc., application of automatic text classication is growing significantly. Many websites offer a hierarchically-organized view of the web. E-mail clients offer a system of ltering e-mail. Academic communities often have a website that allow searching papers and show an organization of papers. The heart of these tasks is human, i.e, the classication is preformed by humans and depends on their backgrounds and knowledge [13].

There are some approaches for automatic text classication. One common technique is employing a rule-based similar to those used in expert systems. However, they generally require manual construction of the rules, making rigid binary decisions about category membership, and are typically difcult to modify [14]. Decision trees are also used for this purpose. Decision tree classiers which are used in this area, consist of a tree in which internal nodes are labeled with words, branches departing from them are labeled with tests on the weight that the term has in the representation of the test document, and leaf nodes are labeled with (not necessarily different) categories [10]. Other machine learning approaches such as neural networks can be found in [14].

Some of these approaches, e.g. Naive Bayes, consider that the text is generated by a stochastic process [7]. A source outputs a sequence of symbols, conceptually infinite. In the text classication, a symbol can be a character, a word or a sentence [13]. In this paper, symbols are assumed to be words . A document is a sequence taken from some sources. So, the statistical properties of documents are determined by the nature of the stochastic processes which generate them. Generally, a class of documents can be generated by a single source. A new document is supposed to belong to the class for which the probability of its

generation is the highest [5]. Bayesian classier is a powerful stochastic method which is used for text classication [13]. Naive Bayes has been successfully applied to document classication in many research efforts [8].

Markov Chain is a discrete stochastic process (as a random walk) in which the probabilities of occurrences of various fu-ture states (or symbols) depend only on the present state of the system or on the immediately preceding state and not on the path by which the present state has been achieved [3]. It is clear that no Markov chain is able to describe text documents completely. However, they are used as models (stochastic process) in this field [2].

It has been proved that Bayesian classiers have the lowest mistake ratio in classication problems [9].

However, when training data are noisy or the nature of data is dynamic e.g. news or the training data do not cover all of the probable cases Bayesian classifier does not performe well. To solve these two important issues, we propose a new approach base on information theory. The proposed method is proven to be equivalent with Bayesian classifier in theory. The experimental results show that it outperformes over Bayesian classifier in these three cases. 20 news groups are selected for evaluating the ITTC's capability in classification. News-groups [13] data is noisy, incomplete and dynamic.

This paper is organized as such: Section 2 describes the background and formulates the problem. The proposed method, ITTC, is explained in Section 3. The implementation and achieved experimental results are elucidated in Section 4. And Finally, Section 5 concludes with the finalizing remarks and future work.

2 Background and problem formulation

In this section the problem definition is presented. Due to ITTC being a stochastic approach, the text classification problem is reformulated to suit being solved by a stochastic process.

2.1 Problem definition

The problem can be formulated as follows: There are C sets[3] of documents, $S_1, S_2, \cdots, S_C$ as training sets. The set $S_i = \{D_{i,j}\}$ contains N_i vectors of words. All members of a set belong to the same class and all classes described by sets are distinct. After preparing the system, it will be employed for working in the real environment in which it receives text documents as vectors and returns a class. The input vectors are typically denoted by V_T.

2.2 Document Representation

There are two models for document representation in stochastic approaches. One model specifies that a document is represented by a vector of binary attributes

[3] In this paper, duplication is allowed in set

indicating which words occur and do not occur in the document. The number of times a word occurs in a document is not captured. The second model specifies that a document is represented by the set of word occurrences from the document. The order of the words is lost in the latter, however, the number of occurrences of each word in the document is captured [7].

In this paper, we use the second model. Assigning to each word an integer value, and using a vector whose i-th entry is the i-th word of the text is a common way to represent a document. We limited the number of possible words by restricting the words' lengths, i.e., each word whose length is bigger than a threshold is represented by 0. Let W be set of all possible words, f be a function that maps each word w to an integer value $f(w)$ and the maximum value that f takes over W be M, then, the number of all possible words in our representation will be $M+1$, which is a bounded integer instead of infinite possible words.

2.3 Stochastic Process Construction

A memory-less source S, emits a sequence of symbols (integer numbers less than or equal to M in this case) according to a probability distribution $\{P_i\}$ with the condition that $\Sigma P_i = 1$ [3]. The source S performs as a stochastic process producing the members of a specific class, T. There is no source which is able to describe the process of generation of documents, completely. This is because the natural languages do not have a grammar, neither context-free, nor context sensitive [6]. Therefore, all documents generated by S are not forced to be in T. On the other hand, the documents in T can be generated using other sources too [5]. The source S is not necessarily memory-less and it could be a complex process. However, the complexer the source, the more parameters to be configured. For example, when a memory-less source is used, we have to find $M+1$ probabilities, while if one wants to use a source which is able to memorize the last symbol, he/she must find $(M+1)^2$, Figure 1. Therefore, adding the ability to remember the last symbol(word), does not necessarily, improve the accuracy of source, because there may be some parameters whose values are very uncertain.

Usually S is inaccessible, and all available knowledge about S is that members of S_t —which are some of documents — belong to T. There is a maximum likelihood memory-less source S which generates S_t. This source maximizes $P(\{D_{t,j}\}|S)$, the probability that the given document will be generated by the source. The distribution $\{P_i\}$ for this source is obtained by dividing the number of times each word occurs by the number of total words in the set [1]. This approach is called *Maximum Likelihood* or ML. If the training set is not general enough, the resulting model will be too far from the original source [16]. Note that, naive Bayes classifiers are the best general classifiers, i.e., their overall errors on problems, both real-world problems and non existing problems, are minimal. There may be some real-world problems on which naive Bayes performs badly and on the other hand, there are some unreal problems on which naive Bayes performs very well.

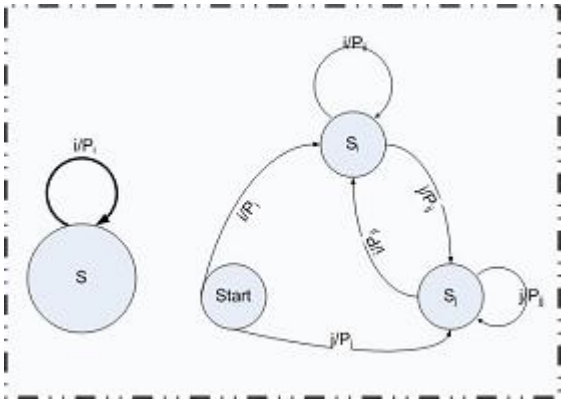

Fig. 1. Sample of a memory-less source and a source with ability of saving last symbol.

2.4 Selecting a class for a new document

Bayesian **D**ecision **T**heory (BDT) deals with the problem of making optimal decisions or taking optimal actions which minimizes expected value of loss. It is proven that the optimal classification is the one which is expected to maximize the probability of membership [11]. As belows:

$$OptimalClass = argmax_C P(V_T|S_C) \tag{1}$$

3 Proposed Algorithm

3.1 Introduction

As we said in the last paragraph, the expected value of miss classification when using BDT is minimal. However, there are some states, on which Bayesian Decision Theory does not perform well:

1. When the training set does not have enough coverage [16].
2. When the class properties are changing during the time. In this case, the probabilities are changing, so the main probabilities of the Markov chains become invalid. One way to solve this problem is to update the probabilities during time.
3. When the feature space of the problem is too big. The big feature set,i.e., dictionary, may cause some features not appear in training set.

The algorithm presented in the next section is going to deal with third type problem. These kinds of problems are common in text classification. Because the domain of the words are usually too big. Also, due to the occurrences of misspelling using BDT, $P(D|S_c)$ will be zero. Therefore, we reject the class c to be the class of D. At first glance, it may look correct but thinking deeper, one could

certainly find some examples in which setting $P(D|S_c)$ to zero is not rational. For example, if there is a mis-spelling in a document then that document could not be in any of classes.

When a word does not occur in the training set at all, there can be two situations. The first one is to consider that the word can not be generated in the main (inaccessible) stochastic process. BDT acts like this. The second situation is that the probability of generating that word by the main process is too small, but unequal to zero. The word does not occur in the training set because of its low rate of occurrence.

3.2 Some backgrounds

BDT uses the probability concept to find the winner class. Let us use another definition for this purpose. Let S be a set of generated vectors from a memory-less Markov chain M, and M' be a memory-less Markov chain constructed from S. If S has some features, M and M' could be very similar. A vector V belongs to the same class as that of members of S, if, with a high probability, it is generated by M. Let V be a long enough string (vector) generated by M, and P_i^V, P_i^M be the probability of the i-th word occurring in vector V and the probability of generating the i-th word in machine M, respectively. It is very likely to have, for all i, $P_i^V = P_i^M$. The idea is, instead of calculating the probability of V being generated by M' (an estimation of M), add V to S and generate another memory-less Markov chain, M'' from this new set. If V is really belonged to the same class as members of S, as discussed above, the probability distribution of M'' must be to a great exact close to that of M'. In fact, we want to find out, adding V to M', how much its output will be changed. That is, how much uncertainty will be added/removed from M'.

Entropy is defined as a measure of the average uncertainty in the random variable or process [15]. Therefore, it can be a good candidate to be used in the proposed algorithm. Entropy for a memory-less Markov chain M is defines as follows:

$$E_M = -\Sigma P_i^M \log P_i^M \tag{2}$$

Entropy is non-negative. The more the value of E_M, the more the uncertainty.

3.3 Algorithm Description

ITTC has two separate phases: training and test.

Training For each S_t, a Markov chain M_t' is generated using ML algorithm. The training is fast and can be completed in O(Size(S_i)) by the following algorithm:

1. Use an array $Count$ with size of $M + 1$, initially set to zero. Also set $TotalCount$ to zero.
2. for each vector V in S_t:
3. for each word w in vector V:
4. Inc ($Count[w]$) and Inc($TotalCount$)
5. For all $0 \leq i \leq M$, set $P_i = Count[i]/TotalCount$.

Test For each vector V waiting to be assigned a class, do the following steps:

1. For each class C,
   (a) set $S'_c = \{V\} \cup S_c$ and build M''_c from S'_c using ML.
   (b) Calculate $\Delta_{E_c} = E_{M''_c} - E_{M'_c}$
2. return $Argmin_c(\Delta_{E_c})$.

The computational complexity of brute force implementation of this algorithm is $O(C * Size(V) + \Sigma_c Size(S_c))$— constructing a new Markov chain for each class, without using existing information — which is not a good performance, because for each test, it should process each member of the training set, to construct the corresponding M'' and calculate $E_{M''}$. There is a better implementation for this algorithm which is in $O(C * Size(V))$. We will present that implementation later on this section.

3.4 Proof

In this subsection, it is proved that under some specific conditions the ITTC performs as BDT. Let define some notations:

- S_M : The set used to construct the Markov chain M.
- n_i^V : The number of occurrences of i-th word in vector V.
- n_i^M : The number of occurrences of i-th word in S_M.
- n_V : The size of the vector V, which is obtained from the number of words in V, i.e., $\Sigma_i n_i^V$.
- n_M : The size of S_M, which is obtained from number of words in the S_M, i.e., $\Sigma_i n_i^M$.
- p_i^M : The probability of M generating i-th word. If M is constructed by ML, then $p_i^M = n_i^M / n_M$.
- p_V^M : The probability of M generating vector V. $p_V^M = \Pi_i (p_i^M)^{n_i^V}$.
- $Logp_V^M = \log p_V^M = \Sigma_i \; n_i^V \log p_i^M = \Sigma_i \; n_i^V \log n_i^M - n_V \log n_M$

We want to relate Δ_{E_c} to p_V^M using a decreasing function F, i.e., $\Delta_{E_c} = F(p_V^M)$. Once F is found, it automatically follows that the higher the p_V^M, the less the Δ_{E_c}. Therefore, **the class with minimum delta entropy, would have the maximum probability for generating** V.

Entropy can be calculated as belows:

$$E_M = -\Sigma_i P_i \log P_i = -\Sigma_i P_i \log \frac{n_i^M}{n_M} = \Sigma_i P_i \log n_M - \Sigma_i P_i \log n_i^M$$

$$= \log n_M \Sigma_i P_i - \Sigma_i P_i \log n_i^M = \log n_M - \Sigma_i \frac{n_i^M}{n_M} \log n_i^M$$

$$\Rightarrow E_M = \log n_M - \frac{1}{n_M} \Sigma_i n_i^M \log n_i^M \tag{3}$$

It can be checked that $0 \leq E_M \leq \log|W|$, in which W is the set of all possible words, as described later, this set is finite. The following formulas calculate $n_i^{M''}$ and $n_{M''}$:

$$n_i^{M''} = n_i^M + n_i^V \tag{4}$$

$$n_{M''} = n_M + n_V \tag{5}$$

Therefore,

$$(3),(4),(5) \Rightarrow E_{M''} = \log(n_M + n_V) - \frac{1}{n_M + n_V} \Sigma_i (n_i^M + n_i^V) \log(n_i^M + n_i^V) \tag{6}$$

, and

$$(3),(6) \Rightarrow \Delta_{E_c} = \log(n_M + n_V) - \log n_M - \frac{1}{n_M + n_V} \Sigma_i (n_i^M + n_i^V) \log(n_i^M + n_i^V) + \frac{1}{n_M} \Sigma_i n_i^M \log n_i^M \tag{7}$$

We can assume that $\log(n_i^M + n_i^V) \simeq \log(n_i^M)$, which is legitimate since under the normal conditions the size of the training set is much larger than that of a single test vector($n_V \ll n_M$). Thus:

$$\Delta_{E_c} \simeq \log(n_M + n_V) - \log n_M + \Sigma_i (\frac{n_i^M}{n_M} - \frac{n_i^M + n_i^V}{n_M + n_V}) \log n_i^M$$

$$= \log(n_M + n_V) - \log n_M + \Sigma_i \frac{n_i^M n_V - n_i^V n_M}{n_M (n_M + n_V)} \log n_i^M$$

$$= \log(n_M + n_V) - \log n_M + \frac{n_V}{n_M (n_M + n_V)} \Sigma_i n_i^M \log n_i^M - \Sigma_i \frac{n_i^V}{(n_M + n_V)} \log n_i^M$$

By replacing $\Sigma_i n_i^M \log n_i^M$, with $n_M (\log n_M - E_M)$ and simplifying we have:

$$\Delta_{E_c} \simeq \log(n_M + n_V) - \log n_M + \frac{n_V (\log n_M - E_M)}{n_M + n_V} - \Sigma_i \frac{n_i^V \log n_i^M}{(n_M + n_V)} \tag{8}$$

We assumed that $|W| \ll n_M$, so $\log|W| \ll \log n_M \Rightarrow 0 \leq E_M \ll \log n_M$. Therefore, $n_V (\log n_M - E_M) \leq n_V \log n_M$. Due to n_M is very larger than both n_V and $\log n_M$, we can consider that it is also larger than $n_V \log n_M$, then, we can ignore the third term. This assumption is the most restrictive assumption in this proof. However it is often legitimate. For example, consider a problem with the following specifications:

There are 20 classes, $C = 20$. Each S_i contains 550 documents (vectors) on the average. Under the condition that the size of all documents are approximately equal, Then, $\frac{n_V \log(n_M - E_M)}{n_M + n_V}$ will be in range $[0 - 0.01]$.
We rewrite Δ_E by removing the third term:

$$\Delta_{E_c} \simeq \log(n_M + n_V) - \log n_M - \Sigma_i \frac{n_i^V \log n_i^M}{(n_M + n_V)} \tag{9}$$

By replacing $\Sigma_i n_i^V \log n_i^M$ by $Logp_V^M + n_V \log n_M$, Δ_E will be:

$$\Delta_{E_c} \simeq \log(n_M + n_V) - \log n_M - \frac{n_V \log n_M - Logp_V^M}{n_M + n_V} \tag{10}$$

If for all i, the i-th training set S_i has approximately the same number of words, then for each M generated during training phase, n_M is approximately the same. For a queried vector V_T, the three first terms of Δ_E are same over all classes (all Markov chains), and only the fourth term is changing. Therefore, Δ_E is in reverse relationship with $Logp_V^M$ and p_V^M .

3.5 Proposed Implementation

The notation used here are the same as those used in the last subsection. To calculate the exact value of Δ_E for the class c, the following algorithm/implementation performs better than the previous one:

Let M be the Markov chain constructed in the training phase. Based on the entropy formula, instead of saving p_i, we save all n_i^M's in a list. The exact value of Δ_{E_c} is:

$\Delta_{E_c} = \log(n_M + n_V) - \log n_M - \frac{\Sigma_i (n_i^M + n_i^V) \log(n_i^M + n_i^V)}{n_M + n_V} + \frac{\Sigma_i n_i^M \log n_i^M}{n_M}$

Let $Cte = \log(n_M + n_V) - \log n_M$ then

$$\Delta_{E_c} = Cte + \frac{\Sigma_i n_i^M \log n_i^M}{n_M} - \frac{\Sigma_{n_i^V \neq 0} (n_i^M + n_i^V) \log(n_i^M + n_i^V) + \Sigma_{n_i^V = 0} n_i^M \log n_i^M}{n_M + n_V}$$

$$= Cte - \frac{\Sigma_i n_i^M \log n_i^M}{n_M + n_V} + \frac{\Sigma_i n_i^M \log n_i^M}{n_M} + \frac{\Sigma_{n_i^V \neq 0} (n_i^M \log n_i^M - (n_i^M + n_i^V) \log(n_i^M + n_i^V))}{n_M + n_V}$$

$$= Cte + \frac{n_V \Sigma_i (n_i^M \log n_i^M)}{n_M (n_M + n_V)} + \frac{\Sigma_{n_i^V \neq 0} (n_i^M \log n_i^M - (n_i^M + n_i^V) \log(n_i^M + n_i^V))}{n_M + n_V}$$

By replacing $\Sigma_i n_i^M \log n_i^M$ with $n_M (\log n_M - E_M)$ and simplifying we have:

$$\Delta_E = Cte + \frac{n_V (\log n_M - E_M)}{n_M + n_V} + \frac{\Sigma_{n_i^V \neq 0} (n_i^M \log n_i^M - (n_i^M + n_i^V) \log(n_i^M + n_i^V))}{n_M + n_V}$$

The complexity of calculating Δ_E from this formula, if E_M is saved once, in worst case is $O(n_V)$ which shows that run time is linearly growing by the size of the queried vector V_T.

4 Experimental Results

To evaluate the proposed method, we have used a test set from the literature [4]. This test set is collected by Ken Lang, contains about 20,000 articles evenly divided among 20 UseNet discussion groups. Many of the categories fall into confusable clusters; for example, five of them are comp.* discussion groups, and three of them discuss religion. These data are publicly available on "http://people.csail.mit.edu/jrennie/20Newsgroups/".

We have used a hierarchical model which is proposed in [17], using ITTC to decide which class is the winner in each level. The correct classification rate, using hierarchical ITTC, was 90.3%. In Table-1, we have compared our method results with four other proposed techniques in the literature. The first method is a Hierarchical SVM-based (H-SVM)text classifier reported in [17]. The idea in H-SVM is to use many SVM classifiers, one in each step, and using a hierarichal approach to reach to a decision. The other three methods are Multinomial Naive Bayes (MNB), Transformed Weight-normalized Complement Naive Bayes (TWCNB) and SVM which are reported in [12]. Also, Table-2 shows the details for each class.
As it is shown in Table 1, the achieved results are better than other four approaches. But, the acheived results from some news-groups are not very well. This is due to two problems: shortage of training data and harsh similarity. The false results which were reported for these groups were trapped in other similar groups, therefore, the results were not very strange.

Method	Accuracy
ITTC	90.3%
H-SVM	89.1%
MNB	84.8%
TWCNB	86.1%
SVM	86.2%

Table-1- Overall results

Group Name	Accuracy
alt.theism	84%
comp.graphics	79%
comp.os.ms-windows.misc	70%
comp.sys.ibm.pc.hardware	67%
comp.sys.mac.hardware	80%
comp.windows.x	80%
misc.forsale	85%
rec.autos	83%
rec.motorcycles	89%
rec.sport.baseball	92%
rec.sport.hockey	95%
talk.politics.guns	93%
talk.politics.mideast	90%
talk.politics.misc	94%
talk.religion.misc	97%

Table-2- Details of parts of results for ITTC.

5 Conclusions and Future Work

In this paper, we have proposed a new Information Theoretic Text Classifier (ITTC). ITTC uses Δ_E as its optimum class criteria. The proposed criterium in this paper was a novel which has outperformed over other proposed approach. It is proven that it performs theoretically like Bayesian classifier. We also represented an efficient implementation for test phase. The experimental results have shown ITTC has better performance than some other previous methods.

In this paper, we used a predefined hierarchy. It may be the case that, this hierarchy is not the suitable one for ITTC. One can use the entropy for constructing the hierarchy. Our future plan is applying the ITTC for other classification problems and evaluating its performance.

References

1. Aldrich, J., and R.A. Fisher and the making of maximum likelihood. *Statistical Science* (1997).
2. Burger, A. Statistical machine learning for information retrieval. *PhD thesis, School of Computer Science, Carnegie Mellon University* (2001).
3. Cover, T., and Thomas, J. *Elements of Information Theory.* John Wiley, 1991.
4. Joachims, T. A probabilistic analysis of the rocchio algorithm with tfidf for text categorizationm. *ICML-97* (1997).
5. Kalt, T. A new probabilistic model of text classification and retrieval. *Technichal Report IR-78* (1996).
6. Manning, C., and Schutze, H. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambride, MA, 1999.
7. McCallum, A. K., and K.Nigam. A comparison of event models for naive bayes text classification. *Working Notes of the ICML/AAAI Workshop on Learning for Text Categorization* (1998).
8. Mehran Sahami, Susan Dumais, D. H., and Horvitz, E. A bayesian approach to filtering junk e-mail. *AAAI-98* (1998).
9. Mitchell, T. *Elements of Information Theory.* New York: McGrow-Hill, 1997.
10. Murthy, S. K. Automatic construction of decision trees from data: A multi-descipline survey, data mining and knowledge discovery.
11. Neumann, J. V., and Morgenstern, O. Theory of games and economic behavior. *Princeton University Press, NJ* (1953).
12. Rennie, J., Shih, L., Teevan, J., and Karger, D. ackling the poor assumptions of naive bayes text classifiers. *ICLM-2003* (2003).
13. Rennie, J. D. M. Improving multi-class text classification with naive bayes. *PHD thesis* (2001).
14. Sebastiani, F. Machine learning in automated text classification. *ACM Computing Surveys (CSUR)* (2002).
15. Shannon, C. E. Prediction and entropy of printed english. *Bell Systems Technichal Journal* (1951).
16. Vapnik, V. N. An overview of statistical learning theory. *IEEE Transation, Neural Networks* (1999).
17. Yongwook Yoon, C. L., and Lee, G. G. Systematic construction of hierarchical classifier in svm-based text categorization. *IJCNLP 2004* (2004).