Document Classification as an Internet service: Choosing the best classifier

Shantanu Godbole
K. R. School Of Information Technology
Indian Institute of Technology - Bombay
Powai, Mumbai - 400076, India
shantanu@it.iitb.ernet.in

February 1, 2001

Abstract

This project investigates some of the issues involved in a new proposal for expanding the scope of the field of Data Mining by providing mining models as services on the Internet. This idea can widely increase the reach and accessibility of Data Mining to common people because one of the primary stumbling blocks in the adoption of mining is the extremely high level of expertise and data resources needed in building a robust mining model. We feel this task should be left to the specialists with access to data and resources, who can provide their most up to date model as a service on the Internet for public use.

In this report we investigate the problem of choosing the right prediction when document classification is provided as an Internet service. When several autonomously learnt classifiers predict different classes for the same document, the problem of choosing the right prediction becomes challenging because we cannot assume a single synchronous step for learning the strengths and weakness of the different classifiers using a training dataset. We present a new algorithm that uses a novel dynamic, instance-based approach to model selection. We evaluate the algorithm on two real-life document classification datasets. Our algorithm performs better than a plurality voting scheme which is the only known method that can operate in such a dynamic setting.

Contents

Chapter 1

The Learning Web

1.1 The Internet as a collaboration resource

The World Wide Web connects millions of machines backed by intelligent human beings. This has heralded an era of sharing. Vast treasure houses of authoritative information that were previously confined to books with limited accessibility are now freely available on the Internet. The Internet has surpassed all expectations of information sharing. The mesh of loose, heterogeneous intelligence that is the Web today is capable of fostering even higher levels of sharing. Complex computational problems are being solved today using the Internet as a very large scale distributed system. Many collaborative efforts aim to build better systems using the Internet by fostering human knowledge and experience sharing.

Sharing of computing power: A large scale distributed computation project called 'The Search for Extra-terrestrial Intelligence at Home[1] (SETI@Home)', has been running for a few years now. Radio signals from the world's massive radio telescopes generate massive amounts of data daily. These signals need to be analyzed for interesting patterns. This analysis is a very computationally intensive task. In this project, a central server coordinates various volunteers who are willing to donate their unused computer time. The controlling server breaks up its large data sets into smaller parts. These smaller data sets are distributed to the volunteers. Whenever the volunteers have idle CPU time to spare, a custom program uses the available resources for its computations. Processed data sets are submitted to the central server which can then put the results together and consolidate the results. All this exchange of data is facilitated by the Internet.

Distributed.net[2] is another collaborative computing effort. This project handles very large scale, mathematically intensive problems. Optimal Golumb Rulers, Mersenne primes, and factoring cryptographic keys are some of the problems being worked upon. Some of these problems are attempted by brute force, requiring large computational power. Volunteers can download client software which communicates

[1] http://setiathome.berkeley.edu/

[2] `http://www.distributed.net/`

with a central server. The client downloads small parts of the problem task and uses spare CPU resources for its computation. Results are submitted to the central server via the Internet where the results are consolidated.

Sharing human knowledge and expertise: The Open Directory Project[3] (ODP - also known as Directory Mozilla or DMOZ), is a human edited directory of websites. It was started with the aim of being the largest, most comprehensive human-reviewed directory on the Web. Over 33,000 volunteer editors have contributed to the directory over time. The directory currently lists over 2.2 million sites in more than 330,000 categories and is growing everyday. All site classification and listing issues are resolved internally via human discussions. The voluntary editors generally manage small categories in their own areas of interest and specialization. hence, the domain knowledge of so many individuals is put to good use. This taxonomy along with the listings and editorial site descriptions is made available freely on the Internet. Many web directories and search engines like Google[4], Hotbot[5], and AOLSearch[6] directly use the ODP data. Other directories try to follow a similar model of human edited directories and develop their own taxonomy. However, the standardization level and scale of success of the ODP is unmatched.

Many search engine companies are capable of crawling a significant part of the Internet. The ODP taxonomy offers a level of standardization for classification. These companies could build automatic document classifiers for the Web based on this standard taxonomy. In such a case, it would be a very useful idea to offer document classification as a service on the Internet.

1.2 Data Mining services

The main goal of this project is to explore some of the issues involved in the proposal of extending the Internet's power of information sharing to knowledge sharing[10]. Anyone with huge amount of data and expertise can host 'knowledge servers' by building models from their accumulated data on any aspect of data mining. A potential user can consult these servers and choose from the opinion of these various sites to make her final decision. These mining servers are used in a totally ad hoc, per-user and per-instance basis much like the way documents are accessed on the web. Some ideas for providing mining models like Risk Prediction, and Collaborative Filtering as services are discussed in detail in [10]. While rich document sources have long since found their way to the Internet, rich sources of data are still within the confines of disconnected large databases. The few instances of knowledge sharing practiced today are all based on model buy-outs. We would like to have better ways to exchange models between two entities. One-time use of someone else's model without explicit

[3] `http://dmoz.org/`
[4] `http://directory.google.com/`
[5] `http://dir.hotbot.lycos.com/`
[6] `http://www.aolsearch.com/`

model exchange would be facilitated if mining models are offered in a service oriented setting.

Document classification services: An imminently useful mining service is a document classification service that accepts documents and predicts its class from a predefined category tree. The Internet has several large categorized document sources. Some examples are

- **PubMed**[7]. This is a search service for medical documents, articles, and papers. It provides access to over 11 million citations in medical databases with links to participating online journals.
- **CoRR**[8], **CiteSeer**[9] and **NCSTRL**[10]: These are online services for computer science papers and articles. They provide a classified collection of papers with searchable citations.
- **The Open Directory Project**[11], and **Yahoo!**[12]. These are popular directories for general web documents.

Many directories like Google, Hotbot, AOLSearch, etc. base their directories on the ODP directory structure. Other directories like Yahoo!, Altavista, and LookSmart have their own structure, but are not as comprehensive and up to date as the ODP. The taxonomies of these directories do not match. For example, the ODP has 'Music' listed under the Arts category, whereas Altavista lists Music as a subcategory under Entertainment. While none of them may be wrong, such discrepancies hinder any effort at standardizing the classification structure.

Search engines crawl the Web without any taxonomy in mind. An exciting extension would be for search engines to classify all pages they crawl according to a directory hierarchy. It is possible for these companies to use a standard taxonomy like the ODP and build large automatic document classifiers. Google already takes its version of the ODP tree into account while returning search results. If a page listed in it's directory matches the search criteria, Google returns that category as a related category along with its top ten results according to the PageRank[1] metric. Pages listed in the ODP have high visibility on the Internet due to replication of it's data. Directory listed pages figure very high in the results. Companies of this scale and size are excellent candidates for participating in a document classification service scenario. They can build large scale automatic document classifiers. The idea of automatic document classifiers as a service will be effective when implementations of this kind materialize.

[7] `http://www.ncbi.nlm.nih.gov/PubMed/`
[8] `http://xxx.lanl.gov/new/cs.html`
[9] `http://citeseer.nj.nec.com/cs/`
[10] `http://www.ncstrl.org/`
[11] `http://dmoz.org/`
[12] `http://dir.yahoo.com/`

Consider a new digital library or newspaper agency that wants to automatically categorize its submissions over a standard taxonomy. Instead of painfully downloading the huge amounts of data on its site and spending money and effort in building a good automatic classifier, it might be more willing to use the categorization service, even if it involves a nominal fee. Portals for selling knowledge is not a new concept in the Internet as is evident in the recent proliferation of "Ask an expert" sites[13]. These are however backed by human experts who answer the questions posed by users either free or at a nominal charge.

FindSame[14] is a free service which already exists on the Internet. It finds documents similar to the one submitted to the site. Similar documents are defined here as documents that contain phrases, sentences, and paragraphs of the submitted document. It is hence different from keyword based search engines which have to deal with a subsequent problem of relevance ranking. This service takes as input an entire document and returns a list of documents that contain any fragment of that document longer than about one line of text. There is a speciality use of this kind of matching in detecting illegal use of speeches, quotes, and press releases.

Our goal is to automate these document classification services in a distributed ad hoc setting, at the same time being instance-based to be practicable in an Internet-like scenario. Several interesting research issues however crop up before such a concept can be deployed in practice. We look at some of these issues in detail in Section 1.2.2.

1.2.1 Advantages of the mining service model

The service model of data mining is more advantageous than the existing model buy-out practice.

- The service model encourages more sharing and provides greater accessibility to end users.
- The user gets access to the most up to date model at all times.
- As the mining server gains new data, its model can be refined and the latest results be made available right away.
- There is no software installation cost and no cost for occasional or one time use.
- The user can be mobile needing only Internet access.

1.2.2 Challenges and research issues

Standardizing vocabulary: Standardization in the access of data and models is an oft-repeated problem. The reason for the widespread sharing of documents is that they are by and large self-describing. In contrast, data even with the schema

[13] `http://dmoz.org/Reference/Ask_An_Expert/`
[14] `http://www.findsame.com/`

definition in hand is too hard to make sense of outside the established community of DBAs and regular users. This is slowly but gradually changing thanks to the push given by the B2B E-Commerce industry. There are a growing number of consortia on standardizing various aspects of day-to-day business information like ebXML[15]. Even in the mining area there is an increasing push for standardization of processes, exchange of predictive models, and APIs.

- CRISP-DM[16] is a working group developing the "CRoss-Industry Standard Process for Data Mining". This group aims at standardizing the way practitioners do data mining. It puts a process in place for data mining practitioners to follow. The current CRISP-DM draft consists of a Process and Users Guide which contains the phases in a typical data mining project. The tasks involved and the relationships between them are described in detail. Other relationships between goals of the project, background of the user, and more importantly the data are also taken into account. The process identifies the main stages of a project as *Data Understanding, Data Preparation, Modeling, Evaluation, Business Understanding, and Deployment.*

- The Data Mining Group[17] is developing the Predictive Model Markup Language (PMML). This is a XML-based language for defining predictive models like decision trees, and facilitates sharing of similar models. Being a vendor independent method, there are no incompatibility barriers to applications. However this does not address the model combination problem.

- OLE-DB for Data Mining. There is an API standardization effort at Microsoft, to incorporate data mining operators in the OLE-DB application programming framework.

For our document classification scenario mentioned in Section [1.2], one option for a classification standardization is that all directories and search engines employ the classification structure of a well-accepted directory like the ODP. This is easier said than done, because the amount of rework required to re-classify existing listings is huge.

Confidentiality of data and model: This is perhaps another major reason for limited sharing of some kinds of data or models on the Internet. Two phenomenon are promising to remove this limitation. First, there are newer data sources of popular appeal that are available freely on the web and are equally or more important to mine. This is wholly true for the document classification example and partially true for the collaborative filtering scenario. Second, better B2B E-Commerce and security infrastructure enables companies to share confidential models on the Internet. Also, companies are much more ready to share summarized conclusions drawn from mining models rather than raw data. For instance, a credit company would perhaps not be

[15] `http://www.ebxml.com/`
[16] `http://www.ncr.dk/CRISP/`
[17] `http://www.dmg.org/`

willing to share its raw data with another credit or insurance company, whereas it may be willing to share its risk prediction model.

Ad hoc model selection: When the multiple sites offer prediction services on the same domain, there is another issue which arises. In the document classification scenario, the user has multiple candidate sites to choose from. How can a user choose between the predictions of these sites? This can be interpreted as a model integration problem for which well known solutions exist but the setting here is very different. First, the participating sites are autonomous which means the input data, method, and time of model construction proceeds autonomously without any co-ordination with other sites. Second, the sites might be constantly evolving and changing their model at will. Thus, in most cases, a knowledge server may not even know about the existence of some other knowledge server. The user therefore, cannot use a single off-line training phase for chalking out a criteria for combining the different models. Finally, even the type of people using the model is drastically different. Unlike the trained data mining specialists, we will now have more one-time occasional users who most likely will not have historical data to benchmark the models and choose between different models. Therefore, existing methods of meta learning that require a separate validation set are ruled out. *"Which prediction to trust?"* becomes a central question which needs to be addressed.

In addition to the above there are several other interesting issues that will arise once mining services become commonly used.

- How does one discover relevant servers for the task at hand?
- How can servers express their scope and capability?
- How does a server's credibility gets established? In the document world, people indirectly establish a site's credibility by linking to that site. What are similar credibility measures for other mining models?

Chapter 2

Evidence-Boosting

2.1 The algorithm

We deal with the specific problem domain of providing mining services in the document classification domain. We set a few criteria upfront that our algorithm should satisfy.

- Site autonomy must be maintained. We can make no assumptions on the classification method each site uses. The proposed method should work on any classification method.
- Do not require a distinct global synchronization step where sites exchange data. In fact, a site should not even have to be aware of the existence of some other peer site. The sites should be totally independent.
- Do not require sites to hold the entire training data.
- Do not assume access to a validation dataset for learning over models.

There exists a simple method that can operate under the constraints specified. This is a Plurality voting or majority voting scheme. No separate training phase as in meta-learning is required. This method can work in an instance-based setting. The predictions of each site's classifier on a particular document are collected at a coordinating site, and the class predicted by a majority of the classifiers wins. This method gives equal weight to all sites and cannot handle speciality classifiers. If it is known that a particular site is an extremely well trained classifier for one class, then this fact and the strength of its predictions do not get special treatment. We would like to favour any speciality classifiers even if they may be the only ones predicting a particular class. This is not possible in the plurality voting method.

Evidence-Boosting We now present our algorithm called the Evidence-Boosting algorithm. First, each site classifies the test instance in a particular class. Following this, a co-ordinator challenges each of the participating sites to prove their prediction. Each site then provides some evidence documents which it deems similar to the test

document. This is motivated by the real-life observation of experts backing their actions by citing similar precedents. Evidence selection is not an easy problem to solve. A site can employ its own metrics to determine documents similar to the query document. Some alternatives are discussed in section [2.2]. At the time of training each classifier there is hence the additional requirement of storing some extra information. This information should be sufficient to determine similarity of training documents and the query instance.

By producing training instances similar to the query instance, a site lays its claim to more trust. After each site passes on its evidences to the co-ordinating site, these evidences are passed on for cross-validation to the other sites. All other sites then predict the class of each of these evidences according to their own classification model. Some voting mechanism then assigns weights to the evidences of each site. This weight measure depends on how sites vote upon the evidences of other sites. A higher weight establishes higher credibility of a particular site. This weighting scheme and various options are discussed in section [2.3]. Finally the site having the highest weight is chosen, and its predicted class is chosen as the winning class.

The Evidence-Boosting algorithm:

1. Send the test instance t to each of N sites (N = number of sites)
2. Collect from each site, the prediction on t,
   and l evidences that support this prediction.
3. For each of the $N * l$ evidences,
   get prediction of the remaining $(N - 1)$ sites on it
4. For each evidence e,
   define a proximity measure P_e from the evidence e to t
5. For each site S_i define weight W_i as
   the sum of the proximity of all its evidences ($\Sigma_{e=0}^{e=l} P_e$)
6. Return the class of the site with the largest weight W_i.

Various proximity measures are discussed in [2.3]. Various methods of evidence selection follow in the next section.

2.2 Evidence Selection

The essence of the Evidence-Boosting algorithm is the central idea that a site should be trusted more because it can produce evidences which are similar to the test document. The site is then sure to have been trained on some documents similar to the test instance. The difficulty is to establish the relevance or closeness of training documents to the test instance and select them as evidences. We discuss below, the various methods we have tried and implemented below. Experimental results are given in section [3.3].

In our particular setting of document classification across multiple sites, we make no assumptions about the kinds of individual classifiers used. For evaluation purposes

however, we use a Naive-bayes text classifier using the multinomial model [2] at each of the sites. We impose a small requirement upon each of the classifiers. Along with the predictions on the test instance and all the evidences, the classifier should provides us with a class probability vector. (In a practical setting, evidence selection is left to the site itself and it can use any method to find similar documents.) The class probability vector of each document should contain the probabilities of the document belonging to various classes.

$$V_e = \{P_c \mid \forall c \in C\}$$

where, each P_c is the probability of the instance/evidence e belonging to one of C classes. By this requirement, we are trying to come up with a generalized method to define similarity of documents given a class probability vector. We then employ some distance measures to rank candidate evidences. According to those distance measures, the l documents closest to the test instance t are then chosen as evidences. The distance measures we explored are discussed below.

L_n distances: Considering the large number of classes we have to potentially deal with, we consider each document to lie in a high dimensional classification space. Each dimension is a class to which the document belongs with some probability. The class probability vector then represents a point in this classification space. Intuitively, we can take the Euclidean ($L2$) distance between any two such points, the test and candidate evidence in this case, and expect that this will be a fair measure of the closeness of documents in the classification space. Against our expectations this method did not perform very well. Upon looking at the results we inferred that one possible cause of this distance measure not returning good evidences was the problem of outliers. The probability values are log probability values as returned by the classifiers. Very low probabilities thus translate to very large negative numbers. We were led to think that taking a Euclidean distance, and so squaring the differences was the cause of the problem. To handle this, we then employed the Manhattan ($L1$) distance measure which we thought would perform well.

Even with the L1 distance measure we did not come up with satisfactory results. Upon further inspection we found some interesting reasons for the above behaviour. The cause of this behaviour was the multinomial model of text classification which the Naive-bayes classifier was using. While a multinomial model is very good to give very accurate predictions about the winning class of a test document, it does not very well reflect closeness of the other loosing classes in its probability scores. We expect the sum of the probabilities to be 1. This particular classification model gives a very high probability (almost always 1) to the winning class, whereas all the loosing classes get very low probability scores. The log of these values thus generates one number very close to zero, and all other values as large negative numbers. We assumed that a pretty uniform distribution of class probabilities exist, which was not the case and hence both the $L1$ and $L2$ distance measures for evidence selection failed.

Further, the multinomial model prefers long documents. Results of evidence selection were good as long as the candidate evidences were long documents. However, these distance measures often came up with low scores for non-winning classes in case of short documents. This led us to observe that in the case of short documents, they were often bad instances from the Web. These short documents often turned out to be either URI redirects or javascript and messages about browser incompatibility. This led us to an important data cleaning decision of removing all very short documents from the dataset. Details about these and other experimental related implementations are discussed in section [3].

These reasons motivated a search for a better method to define similarity of documents in the classification space. We had to somehow neglect the actual probability values, and at the same time not ignore the relative ranking of the classes.

Ranked correlation: Ranked correlation neglects the probability scores and takes relative ranking into consideration. Apart from the same top class, relative ranking of the top few classes is a good indicator of similarity between documents. While choosing evidences from a set of candidates, we are more concerned about a class slipping from the 2nd to the 5th position in the sorted order of classes, than a class slipping from 19th to 20th position.

In this method, we transform the c-dimensional class probability vector to a c-dimensional discrete valued vector by putting in the i^{th} co-ordinate, the position of the i^{th} class in the sorted order of class probabilities. So, for an instance with 6 classes, the class probability vector is transformed as

$$(-101,\ 0,\ -36,\ -5,\ -254,\ -67)\ \Rightarrow\ (4,\ 0,\ 2,\ 1,\ 5,\ 3)$$

Following this we take the L1 distance between these transformed set of points corresponding to t and e.

Ranked correlation across top k classes: This particular transformation described above can be done on the entire vector or only upon the top k classes. Considering only the top k classes is more interesting because we are more interested in the way those classes are confused which are close together on their probability scores and are equally likely to have lost out from being the winning class. Thus it is important to capture the fact that the class which got predicted as the second most likely class for the test instance, is the fifth most likely class for the evidence instance. Capturing a similar phenomenon for the least likely class between the test and evidence instances is not equally important.

Clustering of training instances: Consider a large site which has many thousands of training documents. On the Web scale this is very common. Applying any online evidence selection method across all training documents at query time, will be very time consuming. We note that clustering[5] of documents based on class probability vectors is an optimization that can be well utilized. Such a clustering

can be followed by storing only a fixed set of representative evidences from each cluster. When a new test instance arrives, we determine the cluster of that instance and choose from the stored representatives those l instances that are closest to the test instance in the classification space having the same class prediction. The basic requirement here is to return the instances that are close to the test instance in the decision space not necessarily in the data space.

2.3 Choosing the site with the best evidences

Once each site draws up a set of evidences, we face the problem of selecting the site with the best evidences. This problem is very different from the evidence selection problem. In the prior case we had internal information of the classifier in the form of the class probability vector. Practically, evidence selection is left to the classifier. We now face the more difficult job of selecting the site with the best set of evidence without any information other than the evidences themselves.

For this we define a proximity measure on a per site per evidence basis. The querying site by itself cannot use a distance measure to establish proximity because that is part of the learning problem for which it is asking external help. This proximity is a measure based on the votes other sites give to each evidence. The proximity measure is addressed using the prediction expertise of the other sites to *challenge* the evidence of its peer. The querying site coordinates this challenge process and therefore the participating sites cannot even distinguish between a normal test case and an evidence case. Thus, the sites do not have to know of each other's existence. An evidence is good if all sites give it the same prediction as they gave the test instance. This intuitively holds true irrespective of whether that prediction is correct or not because that document is thus very close to the test instance.

Proximity or distance between instances is not a very easily answerable problem in this setting. In a supervised learning setting, the only distance that makes sense is belongingness to the same class - otherwise distance and classification are the same problem. It would be meaningless to define distance independent of classification. Thus each evidence gets attached with a proximity value to the test instance, calculated through democratic means. For each evidence the true class label is also known. The site that originated that evidence will know its true class label as it originates from its training dataset. This class label is used with the votes of the other sites to define proximity.

Proximity is measured as follows. Every time a site gives the same prediction to both the test and an evidence instance, it is implicitly voting for the relevance of this evidence. We define at each site, the proximity of an evidence e as the fraction of sites which gave e the same class label as their own prediction for t. We now look at some methods of combining the votes of others sites on a sites' evidences.

Site-based method: We define a constant proximity increment at each site. This is $1/(\#sites - 1)$. For each evidence of each site, when any other site gives this evidence

e and the test instance t the same class label, then the proximity of that evidence is incremented by the proximity increment. Thus the proximity of each evidence to the test instance t ranges from 0 to 1. When proximity is 0, this is interpreted as the evidence being very bad because it is not at all like the test instance. On the other hand if the proximity is 1, then all sites label it the same as t and hence the evidence is deemed to be very good.

The constant increment method does not perform well in certain situations. Consider 3 sites voting for the test instance T and two evidences $E1$, and $E2$ of some other site. For the test instance T, the site-based prediction vector is $\vec{T} = \{c_1, c_2, c_3\}$. This means that the first site votes for class 1, and the next two sites vote for class 2. Similarly, $\vec{E_1} = \{c_2, c_1, c_2\}$ *and* $\vec{E_2} = \{c_3, c_1, c_2\}$. The constant increment method gives the same proximity score of 0 to E_1 and E_2 whereas intuitively E_1 is a better evidence. E_2 contains a prediction for c_3 which is a totally different class. This leads us to another method for proximity described below.

Class-based method: We initialize a new vector to zero values, whose size is equal to the number of classes. There is one such vector for the test instance t and each evidence e, which are called T, and E respectively. The vector for T is initialized to $\vec{T} = \{0, 0, \ldots, 0\}$. As each site votes on t or e to belong to class c, the c^{th} value in the vector is incremented by 1. After all sites have voted upon each e, the absolute vector difference between T and E is taken. This vector difference gives a distance measure between the documents. This distance is divided by $2 * \#sites$ to normalize between 0 and 1. Finally the proximity is calculated by subtracting the distance from 1.

$$\text{diff} = \Sigma_{i=0}^{i=C} |\vec{T_i} - \vec{E_i}| \; ; \; proximity = 1 - \{\text{diff}/(2 * \#sites)\}$$

Again this proximity is normalized from 0 to 1 and is interpreted the same way as in the previous method.

The vector difference method also falls short in some situations. If sites agree on the top predictions of evidences, then choosing evidences of a particular site is very hard. If a particular classifier is strongly biased towards a particular class, it's evidences on that class will be agreed upon by other sites. However, it is likely to misclassifiy the evidences of other sites whose true class is an unfavourable class according to it's bias. Moreover, in this method, sites can invert predictions of successive evidences and the prediction vector will remain the same. Also, information that a site very often confuses between two given classes is very important to factor into the algorithm. A confusion matrix is the best structure to utilize this information about confusing classes.

Confusion Matrix method: Rather than considering only the top prediction of each site for the evidence, we can also take into account the site-wise confusion matrix to assign real valued proximity increments to each evidence. One restriction that we must bear in mind is that we can not take or assume any training summary data from any of the participating sites. For example, we can not take the training time confusion matrix from each site as it is validated over all its training documents.

This will violate our requirement of site autonomy. Thus the only confusion matrix we can build is that which we can gather at the time the evidence exchange phase goes on. As each site outputs its predictions on the evidences, the querying site can construct a site-wise confusion matrix. A confusion matrix is prepared for each site and records the predictions other sites on its evidences. The class label of the evidences is known, because each site pulls out evidences from its training data. Now, for each evidence, the proximity increment is a real-valued fraction of original proximity increment of $1/(\#sites - 1)$. This is weighted by the confusion matrix entry which gives the fraction of times that sites agree on predictions of that site. Again this is a normalized proximity value and interpreted same as in the previous cases.

2.4 Advantages and disadvantages of the Evidence-Boosting method

Advantages:

- We don't make any assumptions about the local classification method.
- It is not required that the site maintains the entire training data during model deployment and can even be unaware of the existence of other sites.
- Bad classifier sites are automatically given low weight on their predictions by the very nature of the algorithm, because of the quality of evidences they produce in the cross-validation phase.
- Speciality classifiers automatically get a high weight on grounds of better evidences and are thus favoured.

Disadvantages:

- The most important drawback is the large amount of document transfer required. For classifying one document, we are possibly transporting and classifying an order of magnitude more number of documents across various distributed sites.
- A site can play mischief by always returning a constant class label and the same set of documents as evidences, irrespective of the test instance. An overhead will be incurred to keep this in check.
- This algorithm works only for inductive learning where a site cannot predict a class unless it has seen at least a few examples from that class.

Chapter 3

Experiments

3.1 The data sets

For evaluating the Evidence-Boosting algorithm we performed experiments using some real life datasets. Specifically two datasets were chosen with very different characteristics. Both the datasets required data cleaning operations, which are subsequently described. We chose the 20 newsgroups dataset available from UCI KDD repository, and part of the Recreation directory subtree from the ODP.

3.1.1 The 20 newsgroups datasets

One interesting and standard dataset often used for text mining is the 20 newsgroups[7] dataset. It is a collection of $20,000$ newsfeed articles which are $1,000$ articles each taken from 20 usenet newsgroups. These 20 newsgroups form a diverse set of topics to make text classification interesting. Some of the included newsgroups are highly correlated like *alt.atheism*, and *talk.religion.misc*. Some of the newsgroups are very different with seemingly no relation amongst them like *rec.sport.hockey*, and *talk.politics.guns*. The strong correlation between some groups is apparent from the confusion matrix of the training articles on any classifier. Evidence selection is thus a challenging phase. We take the ranked correlation evidence selection approach discussed in section [2.2].

Cleaning the data: The home site for the 20 newsgroups dataset explicitly warns about the requirement of data cleaning. The articles contain news headers with many fields not being of any relevance to classification. Examples of these fields are: *Path, Lines, Message ID, NNTP-Posting-Host, References* etc. On the other hand there is a field called Newsgroups, which identifies the class of the article. We thus remove all these fields from the headers of all articles. We however retain some fields like *From*, and *Organization* as these fields can be useful for classification. The names of these fields are however stripped and only their content is retained.

Another cleaning operation we do is removal of all words lesser than two characters as these are often occurring words which do not help in classification like *a, an, or*. We also remove *stopwords* like *and, but, with*.

3.1.2 The ODP's Recreation subtree

The second dataset we use is a set of webpages collected from the pages listed in the Recreation subtree of the ODP directory. The Recreation subtree contains 39 categories like Air Hockey, Antiques, Kites, Guns, Motorcycles, and Collecting, which have documents varying from 12 to sometimes over $6,000$ in number. Due to the large variation in number of documents, we select an average size sample of 9 classes with 500 to 2000 documents. The classes we consider are *Amateur Radio, Birdwatching, Boating, Guns, Kites, Living History, Models, Roads and Highways, Scouting.* For a given class we consider all documents rooted at that node, including documents in any subcategory that it might have. A similar observation about correlated classes as in the previous section can be made here for classes like *Scouting, Guns*, and *Amateur Radio.*

Cleaning the data: The cleaning phase for this dataset was more exhaustive. We cleaned the documents of all HTML tags and words up to 2 characters in length. A further cleaning step was required in this dataset. Since we crawled the ODP directory tree at a particular time, often the documents fetched were either URI re-directors, page not found messages, browser incompatibility messages, or frame and javascript code. All these documents were noticed to be very short and hence we discarded all documents below 300 bytes in length including whitespaces. This was a tradeoff as it is likely that we may have overlooked a small number of documents which were very relevant, crisp and to the point about a particular subject. However, we justify our actions by claiming that it is very unlikely that such crisp but very short documents will get listed in the ODP after its strict policy of human-review and including only good-quality, good-content documents. A further step was the removal of common *stopwords*, browser related words and HTML phrases like *and, but, with, Microsoft, Internet, Explorer, Netscape, Navigator, nbsp, quot.*

3.2 Experimental setting

The experiments were conducted in a simulated distributed site setting. Different classifiers representing different sites were run on the same machine. The classifiers used the Naive-bayes method of text classification using the multinomial model[2]. For all classification purposes, the training and test data were obtained after a standard 70-30 split. The 70% of the data was then split across all the participating sites. All experimental test measurements as reported in section[3.3] were those which were obtained on the remaining 30% of the split data. The evidence selection method used in the experiments was the *ranked correlation* method discussed in section[2.2]. After selection of evidences, the site with the best evidences was chosen by using the class-based method discussed in section[2.3].

Test Parameters: The tests were performed on various combinations of the parameters listed below and experiments on these settings were conducted on both our

Method	1	2	3	4	5	6	7	8	9	Average
MAJ	90.1	62.4	87.4	88.7	78.0	47.5	30.7	88.5	90.5	73.8
EB	89.3	63.3	87.8	88.5	79.7	59.6	52.6	89.2	91.4	77.9
ALO	94.2	79.6	97.6	95.7	88.5	75.6	69.0	92.4	95.5	87.6

Table 3.1: Web, s=5, e=5

datasets.

- **Number of Sites:** Various settings for this parameters included values like $2, 3, 5, 7, 10, 12, 15$. The default setting was 5 sites.
- **Number of Evidence:** Various settings for this parameters included values like $1, 2, 5, 7, 10$. The default setting was 5 evidences.

3.2.1 Benchmarks

All results of the above configuration of the Evidence-Boosting algorithm are compared with the following two benchmarks.

- **MV:** A Majority or Plurality voting scheme is implemented as an alternative scheme.
- **ALO:** Additionally we compare our results with an At Least One scheme where, a hypothetical meta-learner will automatically select that site which outputs the true class label of the test instance as it's top prediction. This scheme is an upper bound on performance since if no site correctly predicts the true class label of the test instance, then no combination of the distributed models will yield the right answer.

3.3 Results

EB denotes the Evidence-Boosting algorithm. *MV* denotes the Majority or Plurality voting scheme. *ALO* denotes the At Least One method. All the results are reported as tables for both the datasets. The 20 newsgroups dataset has 20 classes and figures are the average figures across all classes. The Web dataset has 9 classes with figures reported for the average over all classes. Variations of the various parameters as in section[3.2] is given along with each table. $s = number\,of\,sites$, $e = number\,of\,evidences$, $News$ denotes the 20 newsgroups dataset, and Web denotes the Web dataset. Table 3.1 gives the details of the Web dataset in a setting of 5 sites, and 5 evidences.

Number of evidences	1	2	5	7	10
MAJ	84.1	84.1	84.1	84.1	84.1
EB	85.1	85.3	85.6	85.5	85.5
ALO	92.5	92.5	92.5	92.5	92.5

Table 3.2: Web, s=5

Number of evidences	1	2	5	7	10
MAJ	75.6	75.6	75.6	75.6	75.6
EB	75.3	75.9	76.8	76.3	76.8
ALO	87.7	87.7	87.7	87.7	87.7

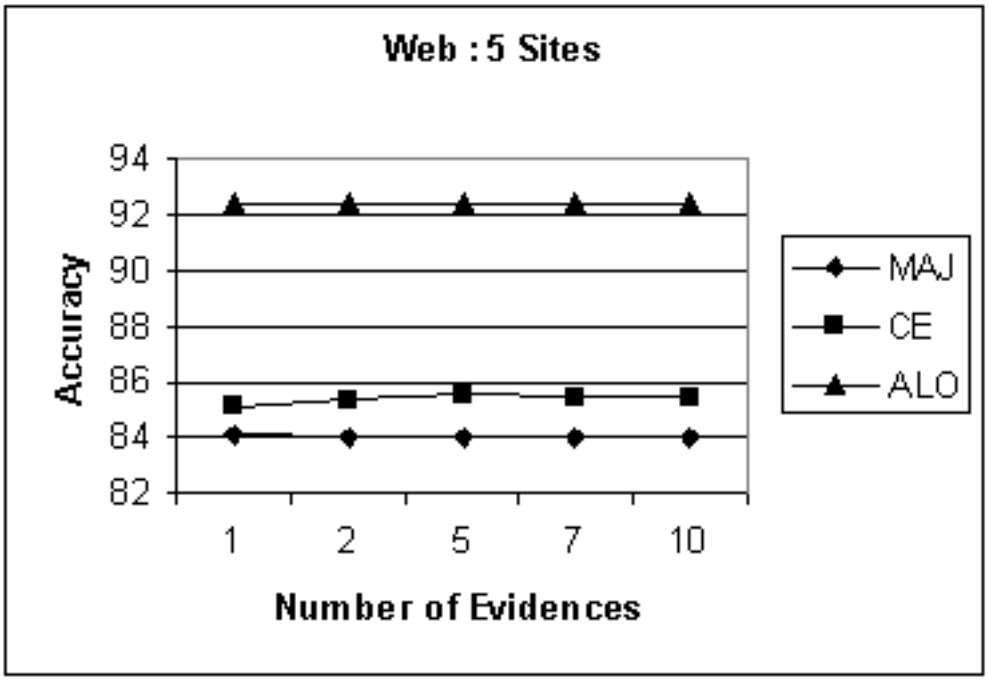
Table 3.3: News, s=3

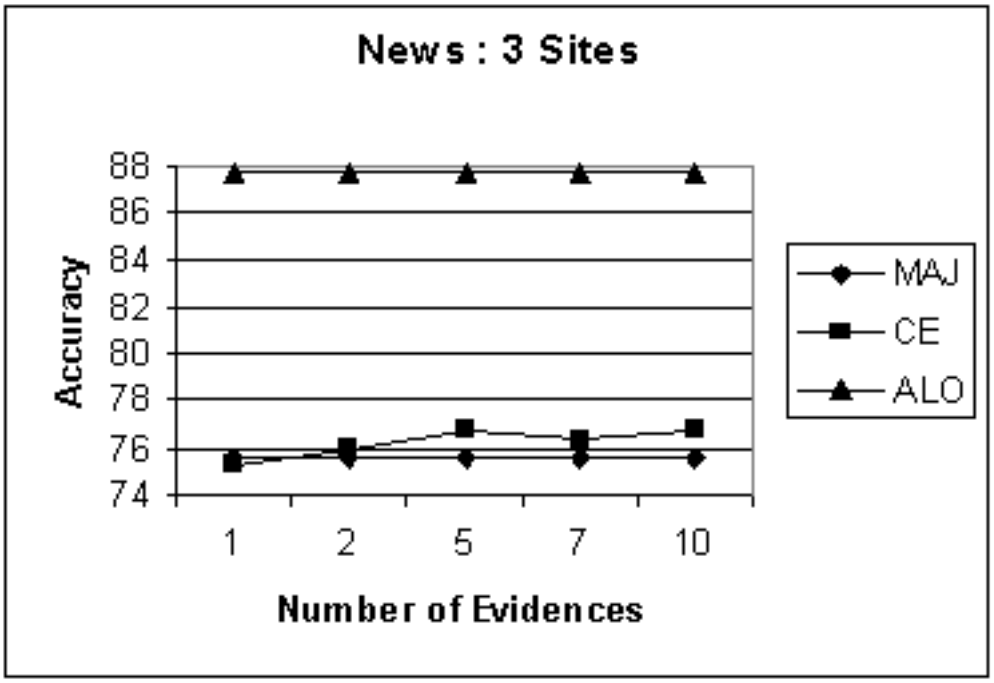
n	2	3	5	7	10	12	15
MAJ	76.4	78.3	79.1	79.7	80.5	81.9	80.6
EB	78.7	80.2	82.1	82.6	83.7	84.5	84.4
ALO	84.2	86.8	90.7	92.8	94.5	95.0	94.6

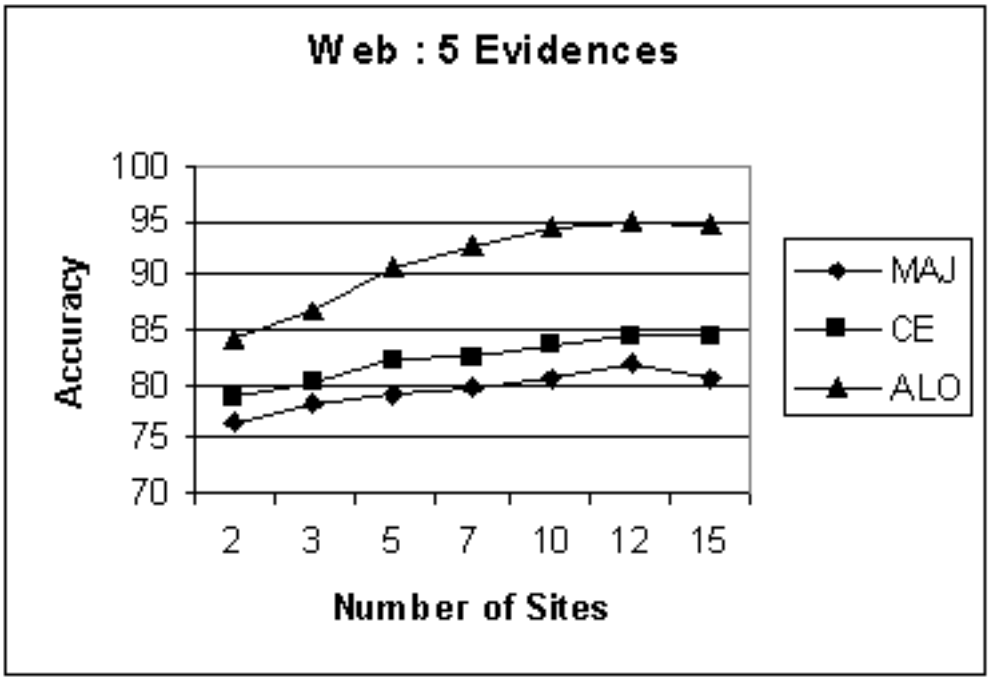
Table 3.4: Web, e=5

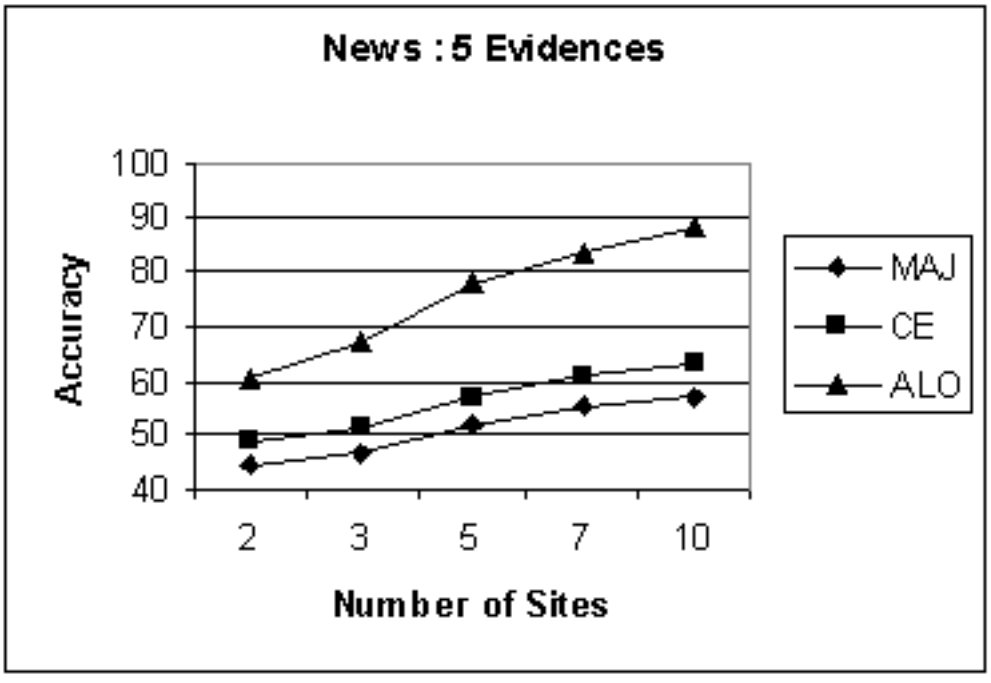
n	2	3	5	7	10
MAJ	44.3	46.8	52.1	55.5	57.2
EB	49.1	51.5	56.7	61	63.4
ALO	60.2	67.1	77.9	83.4	87.9

Table 3.5: News, e=5


Web : 5 Sites
Accuracy
94
92
90
88
86
84
82
1
2
5
7
10
Number of Evidences
MAJ
CE
ALO


News : 3 Sites
Accuracy
88
86
84
82
80
78
76
74
1
2
5
7
10
Number of Evidences
MAJ
CE
ALO





3.4 Observations

A few observations can be made about the results of the experiments. We list the main observations below.

- If we keep the number of sites constant and vary the number of evidences, the accuracy of our Evidence-Boosting approach increases. The accuracy for

the MV and ALO methods remains constant. This is observed for both the datasets. This can be explained by the fact that the evidences are not taken into account by either the MV or the ALO methods. For Evidence-Boosting however, varying the number of evidences changes the weight of each site. As the number of evidences increase, accuracy improves as the site with more and more good evidences gets more positive votes from the other sites in the cross-voting phase.
We keep the number of sites as 5 for the Web dataset and 3 for the News dataset. For the Web dataset, when evidences are increased from 1 to 10, accuracy increases by 0.5% and then steadies. For the News dataset, when evidences are increased from 1 to 10, accuracy increases by 1.5%.

- In the other setting, we distribute all available training data across the maximum number of sites and train the classifiers. Testing is then done over a random subset of sites. For the Web dataset we use 15 sites, and for the News dataset we use 10 sites.
  For the Web dataset accuracy of Evidence-Boosting gets boosted from 78.7% to 84.4%, while accuracy of MV gets boosted from 76.4% to 80.6%, and accuracy of ALO gets boosted from 84.2% to 94.6%. The gap between EB and MV increases from 2% to 4%.
  For the News dataset accuracy of Evidence-Boosting gets boosted from 49.1% to 63.4%, while accuracy of MV gets boosted from 44.3% to 57.2%, and accuracy of ALO gets boosted from 60.2% to 87.9%. The gap between EB and MV increases by 1.4%.
  In both cases EB performs better than MV with its advantage increasing with number of sites. ALO also gets boosted in both cases as is to be expected.

Chapter 4

Related Work

All prior work in combining classifiers in a distributed mining setting can be grouped along two orthogonal dimensions. One of these dimensions is the requirement of a separate meta-learning phase. This is a training phase over the results of the learned models. A learning run over a separate validation dataset is required. On the other hand there are instance-based methods which choose from amongst many models after seeing the test instance. This selection of models is either static or dynamic.

We first discuss the main methods and at the end we classify these methods on the basis of these two dimensions.

Combiner: This method explicitly trains a new meta classifier using the predictions of the component classifiers using a validation data set[3]. The meta learner can be of various types depending on the set of attributes used for meta learning. On the one extreme are meta-learners that use only the class predictions of the component models for training and on the other extreme are those that use both the class predictions and all the original input attributes. The latter are called Augmenters.

Plurality voting: In situations when one needs to select from multiple disconnected models in the absence of any synchronization or validation set, a majority or plurality voting scheme is quite often a good scheme to follow. A simple majority from among the output predictions is taken and selected as the answer. In case of a tie in the number of predictions of various classes, a random set of agreeing predictions is chosen to be the right answer. In chapter [3], we use the Plurality voting scheme as a benchmark to test our Evidence-Boosting algorithm against. We take it to be a lower bound on performance accuracy and expect our algorithm to be at least as good as, and better than this scheme.

Confidence of classification: Our notion of having a classifier output additional information based on which we could establish the certainty of this classifier's prediction is related to some prior work on outputting confidence with classifiers. One related approach[8] discusses the possibility of having classifiers output a prediction only when they are certain and this requires each classifier to evaluate the amount

of confidence or certainty it has in its classification. Confidence is a function of the accuracy of the training data surrounding the test instance. We extended this notion with another crucial factor of including the proximity of the training data to the test instance. The proximity factor becomes particularly important for combining several distributed classifiers. None of the previous work studies the ad hoc distributed setting that we are concentrating on. Our distributed setting presents a new angle not previously addressed, and that is making the confidence a function of the coverage of the data around the test instance also.

Coverage-based approaches: The motivation behind these approaches is to capture the difference in the data space covered by each classifier. For each test instance t it outputs a real-value that measures the coverage in terms of the concentration of local training data around t. There are two approaches in this category.

- **Independent-coverage approach:** Coverage is measured using a two classifier system. The first classifier arises from the original classification problem and the second is a new classifier that estimates the fraction of local training data around the test instance. In an ideal setting training data for the second classifier will consist of the union of training data (without the true class labels) from all sites. Each site trains its local classifier by assigning class label "1" to local training instances and '0' to all other instances from other sites. When a new test instance arrives, each site returns along with its class prediction, the coverage fraction from the second classifier. This fraction directly serves as a weight of that site's prediction. If the fraction is high, the site contains a large number of training data around it, if the fraction is zero the instance lies in a space where the local classifier has no training data, and therefore we should ignore the prediction of that site. Such exchange of training data or coverage information between sites and synchronization steps clearly violates the autonomy requirement.

- **The Relevant-coverage approach:** We address the disadvantage of measuring irrelevant proximity by integrating the second classifier with the first one. Thus coverage of different regions by different classifiers is measured in the classification space rather than the original data space.

The main problem of the coverage approaches is data exchange between the sites. One solution to this problem is to exchange compact summaries of data across sites instead of the raw tuples. However, since sites are allowed to change their models with the addition or deletion of data, there is also the issue of how frequently should these data summaries be exchanged. Due to this data exchange, coverage based techniques do not meet our goal that sites need not be aware of even the existence of other sites.

SCANN: This stands for *S*tacking, *C*orrespondence *A*nalysis, and *N*earest *N*eighbour search. The basic premise of $SCANN$ is to find out highly correlated dimensions in a classification space and reduce the high-dimensional output sample space to a low dimensional equivalent space of the highly correlated dimensions. SCANN relies on

correspondence analysis to map the estimates of the learned models into a new representation space upon which a higher level model is computed.

Stacking is used to generate extra data based on the predictions of the various learned models. This is followed by a correspondence analysis step to find out the highly correlated dimensions. In correspondence analysis, a scaled space is found from auxiliary structures defined on the stacking data. Lastly, a nearest neighbour method is used in this reduced dimensional space to predict the class label of the test instance. Details of correspondence analysis and the $SCANN$ algorithm can be found in [4] where experiments on standard datasets are also discussed in detail. Additionally, [9] discusses some typical statistical properties of standard datasets to keep in mind before generalizing from the results of small experiments.

Nearest neighbor approach: We could have one site where a classification region has 100% accuracy and another site where the accuracy is lower for the *same* region. Yet if we have a test instance that is closer to the training instances of the second site we would choose its prediction because we believe that the reason the first site has higher predicted accuracy is not because it is a better classifier but because it has not seen enough examples from the "confusion" region that the second site has seen.

The simplest approach is for each site to implement a nearest neighbor classifier and along with the standard prediction, output for the given test instance the average distance of the k nearest neighbors to it[6]. We then choose the prediction of the site with the smallest average distance. Alternately, each site can return average distances to each class within its k neighbors. The central site combines the distances for each class and picks the highest weight class. The problem here is that a nearest neighbour classifier is imposed on the system. Also, for meaningfully comparing distances of different sites, we require different sites to use the same features and distance metrics. Thus the goal of site autonomy is violated. Moreover presence of members of confusing classes in a neighbourhood is not taken into account.

Classification of these methods: Consider the requirement of a separate training phase over a validation dataset. Combiner, SCANN and the Coverage-based methods are examples of approaches that require a separate training phase over a validation dataset. Plurality voting and the nearest neighbour approaches do not require any such phase.

We can also classify these methods on the basis of their being instance-based where model selection occurs after the test instance is seen. SCANN, Nearest-neighbour and the coverage based approaches are instance-based methods selecting the winning model dynamically. On the other hand Plurality voting statically selects the winning model.

We see the nearest-neighbour approach is the only method requiring no separate training phase on a validation dataset. It also selects models dynamically after seeing the test instance. However, this method has the drawback that all participating classifiers are forced to be of the same kind.

Chapter 5

Conclusions

Implementing mining models as services on the Internet is a very useful idea. We have seen many possible settings where document classification services could be useful on the Internet. We have also taken some approaches in solving some of the issues involved. More work needs to be done before this idea can be deployed in practice.

We have discussed the Evidence-Boosting approach by giving its details in chapter[2], and have seen experimental evaluations. The Evidence-Boosting algorithm satisfies all conditions of site autonomy, model independence and requires no training phase on a validation dataset. It performs better than a Plurality voting scheme which is the only known contender which can work in a setting where models are independent of each other and selection is done after looking at the test instance.

5.1 Future work

This topic is flush with ideas for future work. The current approach is limited to a specific case of single-level classification. We would like the Evidence-Boosting algorithm extended to work for an entire hierarchical classification setting. Better evidence selection methods would increase the performance of our algorithm. We would also like to consider confusing classes and secondary predictions of the classifiers involved.

Bibliography

[1] Sergey Brin and Larry Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World Wide Web Conference*, Brisbane, Australia, 1998.

[2] Soumen Chakrabarti. Data mining for hypertext - a tutorial survey. In *SIGKDD Explorations - Volume 1, Issue 2*, pages 1 – 11, Jan 2000.

[3] Andreas Prodromidis Philip Chan and Salvatore Stolfo. Meta-learning in distributed data mining systems: Issues and approaches.

[4] Christopher Merz. Using correspondence analysis to combine classifiers. In *Machine Learning*. 1998.

[5] Tom Mitchell. *Machine Learning*.

[6] Sree Hari Nagaralu. Dynamic instance-based model selection. M.tech. thesis, Department of Computer Science and Engineering, IIT-Bombay, January 2000.

[7] The 20 newsgroups dataset. http://www.ai.mit.edu/∼jrennie/20_newsgroups/.

[8] F. Provost and D. Jensen. Evaluating knownledge discovery and data mining. In *Tutorial notes from the fourth KDD conference*, 1998.

[9] Steven Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. In *Data Mining and Knowledge Discovery*, pages 1:317–328, 1997.

[10] Sunita Sarawagi and Sree Hari Nagaralu. Data mining models as services on the internet. In *SIGKDD Explorations*, pages 24 – 28. July 2000.