Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers

Shantanu Godbole
Indian Institute of Technology - Bombay
Annual Progress Report

January 2002

Abstract

A common way to evaluate a multi-way classifier is a confusion matrix that plots, for each of the learned concepts, the true class of test instances against the predicted classes. Aggregate accuracy figures of the classifier are obtained by summing up the diagonal entries of the confusion matrix. However, invaluable information about the relationships amongst classes is often ignored. In this report we show various ways in which the notion of similarity amongst subsets of classes from the confusion matrix can be exploited.

First, we provide a mechanism of generating more meaningful intermediate levels of hierarchies in large flat sets of classes. This provides valuable navigational aid in browsing large text collections like Internet directories.

Second, we show how large multi-class classification tasks can be scaled up with the number of classes. This angle to text classification has been ignored so far in much existing work. New methods like Support Vector Machines have high accuracy but are expensive to run, do not scale to large number of classes, and are not inherently designed for multi-class tasks. We propose a two stage scheme where a confusion matrix from a fast, mediocre accuracy classifier like naive Bayes can be used to derive a graph, where classes are linked to each other based on their degree of confusion with each other. For each class we then identify a sub graph where classes confuse with it. We have now broken up the initial large multi-class problem into smaller sub tasks where, for each class only its relevant sub graph needs to be considered. We use high accuracy, expensive classifiers like SVMs for these sub tasks. The results are promising with significant performance gains of the graph-based method over multi-class SVM classifiers. The resulting accuracy is also significantly higher than the original naive Bayes classifier and is comparable to the best multi-class SVM classifier.

1 Introduction

A common technique of evaluating a multi-way classifier is to see the confusion matrix output as a result of the testing phase of the classifier. The confusion matrix plots, for each of the learned concepts, the true class of test instances against the predicted classes. Aggregate accuracy figures of the classifier are obtained by summing up the diagonal entries of the confusion matrix. However, invaluable information about the relationships amongst classes is often ignored. We explore various methods of exploiting the notion of similarity amongst subsets of classes using the confusion matrix.

One use of the confusion matrix is to determine pairwise similarity between all sets of classes using some similarity metrics. The resultant similarity values can be used to run standard hierarchical agglomerative clustering algorithms, to get a dendrogram of the original set of classes. This is a very interesting method to get hierarchies of classes where none exist initially. These intermediate levels of hierarchies prove to be valuable navigational aids when browsing large text collections like Internet directories. We can also identify subsets of classes which are similar to each other, and we propose a method of building multi-level classifiers. Having identified such similar sets of classes, we train a second level of classifiers for these similar sets. These classifiers can be expensive, high accuracy Support Vector Machines. Training times for SVMs scales up very well for these second level classifiers and they perform well in terms of accuracy.

One dimension to large scale classification which has not been explored in sufficient depth in the literature, is the scaling up of the classification systems with respect to a very large number of classes. One domain with potentially large number of classes and having very high dimensional data is text. In text classification, except Naive Bayesian systems, most of the other systems involve solving complex sub-problems often exponential in time complexity. Support vector machines [Joa98], which are known to give very good accuracy, are not geared toward solving multi-class classification problems. Training time for SVMs blows up with the number of dimensions, and text corpora are well known to have several thousands of useful features. It is well known that Naive-Bayesian classifiers based on a multinomial generative model of documents perform fairly well on standard test sets. We propose a two stage scheme where a confusion matrix from a fast, mediocre accuracy classifier like naive Bayes is used in the first stage to derive a graph. In this graph, classes are linked to each other based on their degree of confusion with each other as indicated in the confusion matrix. For each class we then identify a sub graph where the class has outgoing links to other classes with which it is most likely to have caused confusion. In the second stage, we take the predictions for each test instance from the first stage classifier, and identify the sub graph of classes with the predicted class. We have now broken up the initial large multi-class problem into smaller sub tasks where, for each class only its relevant sub graph needs to be considered. We use high accuracy, expensive classifiers like SVMs for these sub graphs where only binary classifiers are used to finally come up with a prediction for the test instance. The results are promising with significant performance gains of the graph-based method over multi-class SVM classifiers. The resulting accuracy is also significantly higher than the original naive Bayes classifier and is comparable to the best multi-class SVM classifier.

Section 2 begins with a detailed discussion of confusion matrices and gives details about the idea of building multi-level classifiers to increase classification accuracy. Section 3 outlines a general method for automatically generating a hierarchy of classes from a given flat set of classes. We introduce our technique for breaking up a multi-class classification problem into smaller binary classification sub-problems using a graph-based approach in section 4. Section 5 describes in detail our experiments with standard text datasets as well as with one very large crawl of a public Internet directory. Section 6 contains comments, conclusions, and future work.

2 Multi-level classifiers

The confusion matrix is a standard output representation of classification problems. It plots the true class of instances in a classification problem against the predicted class. It gives the predicted distribution of the test instances into each of the trained classes. For a n-way classification task, it is a nxn matrix, with entries in row i summing up to the number of test instances in class i. Ideally, with 100% accuracy, the nXn matrix only has diagonal entries corresponding to all test documents being correctly predicted to their true class. In reality, the confusion matrix is often smudged with small values being distributed all over. In our experience, the confusion matrix gives much more information about the nature of the classification problem involved.

Figure 1 shows a sample confusion matrix of a 20 class problem in text categorization. 20-newsgroups [20N] is a standard text categorization benchmark used in the literature. The dataset contains $1,000$ articles from 20 newsgroups about various topics. We perform a random $70\% - 30\%$ train-test split on the documents. In figure 1 the confusion over the 300 test documents for each of the classes is shown.

.	Classname	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	alt.atheism	251	6	1	3	32	1	1	2	1	2	0	0	0	0	0	0	0	0	0	0
2	soc.religion.christian	9	277	0	1	6	0	0	1	0	0	0	0	0	1	2	2	0	0	0	1
3	sci.space	3	1	273	1	0	1	2	0	1	1	9	0	0	1	2	3	0	0	1	1
4	talk.politics.misc	2	0	3	213	24	3	0	17	3	0	0	0	0	0	0	1	0	1	33	0
5	talk.religion.misc	88	36	2	23	132	0	1	0	0	0	0	0	0	0	0	2	0	1	15	0
6	rec.autos	0	0	0	3	1	272	0	0	0	7	1	2	1	6	4	1	0	0	2	0
7	comp.windows.x	1	1	2	1	0	1	246	0	2	2	30	5	3	1	1	2	1	1	0	0
8	talk.politics.mideast	0	3	1	18	0	0	0	275	0	1	0	0	0	0	0	0	0	1	1	0
9	sci.crypt	1	0	1	2	1	0	3	0	284	0	3	0	1	0	0	1	0	0	3	0
10	rec.motorcycles	0	0	0	1	0	4	1	0	0	286	1	2	0	1	2	1	0	0	1	0
11	comp.graphics	0	1	2	1	1	0	10	1	2	0	243	23	7	3	3	3	0	0	0	0
12	comp.sys.ibm.pc.hardware	0	0	0	0	0	2	7	0	1	0	5	243	23	12	3	1	3	0	0	0
13	comp.sys.mac.hardware	0	0	1	1	0	2	1	0	0	0	7	10	260	8	9	1	0	0	0	0
14	sci.electronics	1	0	1	0	1	5	2	0	2	0	7	13	13	245	6	3	0	1	0	0
15	misc.forsale	0	1	4	2	0	12	1	0	0	4	1	19	10	8	233	1	0	1	1	2
16	sci.med	0	1	5	0	1	1	0	0	0	1	2	0	2	7	2	275	0	1	1	1
17	comp.os.ms-windows.misc	1	0	2	0	1	1	58	1	3	0	38	71	17	3	6	0	97	1	0	0
18	rec.sport.baseball	2	1	1	0	0	0	0	0	0	0	4	0	0	0	1	1	0	282	1	7
19	talk.politics.guns	0	0	0	9	5	1	0	0	1	0	0	0	0	1	0	0	1	1	281	0
20	rec.sport.hockey	0	1	0	0	0	1	0	0	0	2	0	0	1	1	0	0	0	3	0	291

Figure 1: 20-newsgroups confusion matrix

Observations from a Confusion matrix: A large number of test instances of a particular class may be misclassified into another class or a set of classes. From domain knowledge we often know that some classes are similar to each other. In the setting of the classification problem it is not unreasonable to expect that these similar classes will 'confuse' amongst each other. That means that if A and B are two classes similar to each other, then test instances from A, when misclassified, get predicted quite often as B and vice-versa. A and B are thus said to be confusing classes.

For the 20-newsgroups data, from domain knowledge we know that the articles in the newsgroup *alt.atheism* are apt to be confused with articles in the class *talk.religion.misc* and *soc.religion.christian*. Similarly, articles in *rec.autos* are more likely to be similar to

articles in *rec.motorcycles* than any of the other classes. Following this we can form sets of classes, which purely by domain knowledge and intuition can be clubbed together manually on the basis of perceived similarity. We form the following 4 groups:

- **Politics/Religion:** *alt.atheism*, *talk.religion.misc*, *soc.religion.christian*, *talk.politics.mideast*, *talk.politics.guns*, *talk.politics.misc*
- **Computers:** *comp.graphics*, *comp.windows.x*, *comp.os.mswindows.misc*, *comp.sys.ibm.pc.hardware*, *comp.sys.mac.hardware*, *misc.forsale*
- **Recreation:** *rec.autos*, *rec.motorcycles*, *rec.sport.baseball*, *rec.sport.hockey*
- **Science:** *sci.crypt*, *sci.space*, *sci.med*, *sci.electronics*

We re-organize the confusion matrix by interchanging the rows and the corresponding columns of the classes. This maintains the diagonal property of the confusion matrix. This is shown in Figure 2.

.	Classname	0	4	1	7	18	3	10	6	16	11	12	14	5	9	17	19	8	2	15	13
0	alt.atheism	250	32	6	2	0	3	0	1	0	0	0	0	1	2	0	0	1	1	0	0
4	talk.religion.misc	88	132	36	0	15	23	0	1	0	0	0	0	0	0	1	0	0	2	2	0
1	soc.religion.christian	9	6	277	1	0	1	0	0	0	0	0	2	0	0	0	1	0	0	2	1
7	talk.politics.mideast	0	0	3	275	1	18	0	0	0	0	0	0	0	1	1	0	0	1	0	0
18	talk.politics.guns	0	5	0	0	281	9	0	0	1	0	0	0	1	0	1	0	1	0	0	1
3	talk.politics.misc	2	24	0	17	33	213	0	0	0	0	0	0	3	0	1	0	3	3	1	0
10	comp.graphics	0	1	1	1	0	1	243	10	0	23	7	3	0	0	0	0	2	2	3	3
6	comp.windows.x	1	0	1	0	0	1	30	246	1	5	3	1	1	2	1	0	2	2	2	1
16	comp.os.ms-windows.misc	1	1	0	1	0	0	38	58	97	71	17	6	1	0	1	0	3	2	0	3
11	comp.sys.ibm.pc.hardware	0	0	0	0	0	0	5	7	3	243	23	3	2	0	0	0	1	0	1	12
12	comp.sys.mac.hardware	0	0	0	0	0	1	7	1	0	10	260	9	2	0	0	0	0	1	1	8
14	misc.forsale	0	0	1	0	1	2	1	1	0	19	10	233	12	4	1	2	0	4	1	8
5	rec.autos	0	1	0	0	2	3	1	0	0	2	1	4	272	7	0	0	0	0	1	6
9	rec.motorcycles	0	0	0	0	1	1	1	1	0	2	0	2	4	286	0	0	0	0	1	1
17	rec.sport.baseball	2	0	1	0	1	0	4	0	0	0	0	1	0	0	282	7	0	1	1	0
19	rec.sport.hockey	0	0	1	0	0	0	0	0	0	0	1	0	1	2	3	291	0	0	0	1
8	sci.crypt	1	1	0	0	3	2	3	3	0	0	1	0	0	0	0	0	284	1	1	0
2	sci.space	3	0	1	0	1	1	9	2	0	0	0	2	1	1	0	1	1	273	3	1
15	sci.med	0	1	1	0	1	0	2	0	0	0	2	2	1	1	1	1	0	5	275	7
13	sci.electronics	1	1	0	0	0	0	7	2	0	13	13	6	5	0	1	0	2	1	3	245

Figure 2: 20-newsgroups re-organized confusion matrix

We can see natural block-diagonal clusters appearing for the *Politics/Religion*, *Computers*, and *Recreation* sets of classes. This ad-hoc and manual grouping of classes into sets gives us surprisingly clean block diagonals. Such manual grouping of classes is clearly not possible for larger datasets with larger number of classes. We clearly need a better way to identify sets of confusing classes. We outline our method for this in section 3. We can now train a classifier for each of the clusters visible in the reorganized confusion matrix. The justification for this is similar to the one presented in previous work in hierarchical classification [CDAR98]. Consider a tree-like topic taxonomy like the ODP. *Car* and *Auto* may be very good discriminative words at the top level of the taxonomy leading into *Buiness/* or *Recreation/*. However, they are useless from the point of discriminating classes at the level *Recreation/Autos*. Similarly *can* is very often used as a stopword, but may be an excellent discriminative feature at the *Science/Environment* level.

We train smaller classifiers on these identified sets of classes. We find that we get a significant boost in accuracy. One reason for this is the better discrimination between noise words and features as we move from a coarse-grained to a more fine-grained classification granularity. This is the same experience with hierarchical classification. Since we have a significantly smaller number of classes to deal with, we can now use more sophisticated classification techniques and do better feature selection. This grouping of classes enables us to use SVMs for higher accuracy at the level of smaller groups of classes(level 2 classifiers in the experimental section. Using Naive-Bayes classifiers itself gives us quite a boost in accuracy.

Consider the multinomial model for Naive-Bayesian text classification [Cha00].

$$Pr(d|c) = \binom{d_2}{\{n(d,t)\}} \prod_{t \in d} \theta_{c,t}^{n(d,t)}$$

Here $\theta_{c,t} = \frac{1+\sum_{d \in D_c} n(d,t)}{|T|+\sum_{\tau} \sum_{d \in D_c} n(d,\tau)}$

We note here that the parameter smoothing quantity $\theta_{c,t}$ is dependent on the number of word occurrences in documents of a class c and the total size of the vocabulary T. In breaking up the classification problem into smaller sub problems, we do not alter the documents contained within a class. That leaves us with the vocabulary size T. With a smaller text classification task, T is bound to be smaller, $\theta_{c,t}$ is bound to be larger, and hence discriminative words which were earlier swamped out by the sheer size of T and restrictions on feature set size now make significant contributions to $\theta_{c,t}$ and hence to $Pr(d|c)$.

3 Automatic generation of hierarchies

The confusion matrix gives us an insight into the similarity of classes in 'confusion space'. Two classes are similar if they confuse or mis classify test instances amongst a similar set of classes. This confusion set includes the two classes themselves, represented by the corresponding diagonal elements. We can treat each row of the confusion matrix to be a vector belonging to the corresponding class. The i^{th} coordinate of the vector is the degree of confusion of the class in question with the i^{th} class. We normalize all row vectors to one. Ideally in the case of 100% accuracy, the i^{th} coordinate of the i^{th} vector will be 1 and all other coordinates will be 0.

Consider the sample 4-class confusion matrix given in table 3. In confusion space representation, the vector $\vec{R} = \{4, 0, 6, 0\}$ represents the class R. We now have a choice of various similarity measures to find similarity of the class vectors in confusion space. We choose the simple $L1$ distance measure. The $L1$ distance measure sums up the absolute differences in coordinate values between the two vectors. We then take pair-wise similarity between all pairs of classes of the matrix and make an upper triangular similarity matrix. Other alternative distance measures are the L_2 and $KLdistance$ measures. $L2$ distance between the class vectors would give slightly extreme scores but would still preserve the similarity between classes. Dot product between the class vectors would indicate the inverse of similarity. $L1_{dist}(\vec{R}, \vec{G}) = 8$ and $L1_{dist}(\vec{B}, \vec{Y}) = 6$ The resultant similarity matrix is shown in Figure 4

This clearly shows us that the classes B and Y are very similar to each other. This

is what we also observe from the confusion matrix. The class pairs RG and RB are less similar to each other. Classes R and Y are quite dissimilar.

	R	G	B	Y
R	4	0	6	0
G	0	4	6	0
B	0	0	7	3
Y	0	0	4	6

Figure 3: Confusion matrix for 4 class problem

	R	G	B	Y
R	0	8	8	12
G	-	0	8	12
B	-	-	0	6
Y	-	-	-	0

Figure 4: Similarity matrix with L1 distance

	R	G	B	Y
R	0	32	26	88
G	-	0	26	88
B	-	-	0	18
Y	-	-	-	0

Figure 5: Similarity matrix with L2 distance

This similarity matrix is then given as an input to a Hierarchical Agglomerative Clustering (HAC) algorithm [JW98]. Various HAC algorithms exist which differ on their policy of combining clusters. We choose Ward's method as it is known to give the most stable clusters as investigated in previous work in hierarchical clustering of documents [EHW86]. The result of HAC is a dendrogram. In our case we get a dendrogram where similar classes are clustered together along a reference distance axis showing similarity of classes. Various HAC algorithms will give vastly different dendrogram structures [JW98]. Single-linkage clustering is known to be confused by nearby overlapping clusters and it's tendency to pick out long string like clusters is known as *Chaining*. Complete-linkage clustering on the other hand is known to give oblong spheroids as clusters. Other methods are known to be prone to inversion. Inversion occurs when a cluster joins an existing cluster at a smaller distance than that of a previous merge. Ward's method is a known stable algorithm and we found that it gives the best clustering for our particular problem. With Ward's method and L1 distance as the the inter-class similarity metric, the resulting dendrogram for the 20-newsgroups dataset based on the confusion matrix in figure 1, is shown in figure 6. In our experiments, we found similar quality of clusters with the L_1 and $KLdistance$ measures, though we report figures only with the L_1 measure.

We note here that *sci.electronics*, which we manually grouped in a science cluster in figure 2, actually falls in the computers cluster. This can also be seen in the last column of *sci.electronics* in figure 2. The other science topics are not related to electronics whereas the computer hardware classes, like *comp.sys.mac.hardware*, are bound to be about electronics related things.

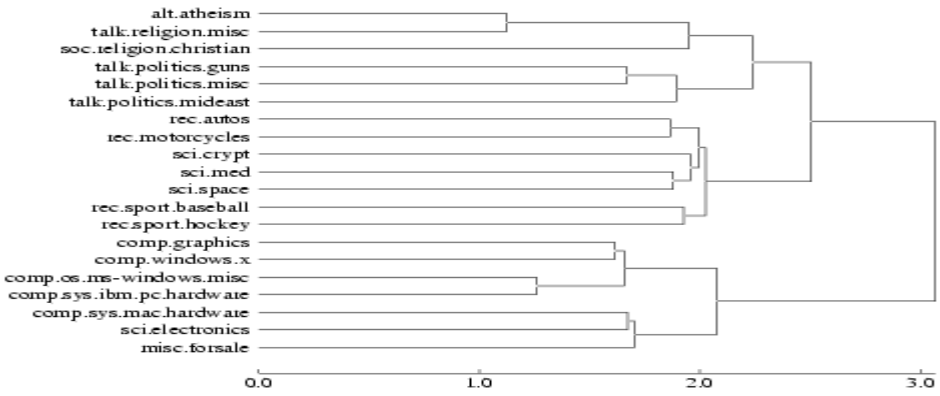


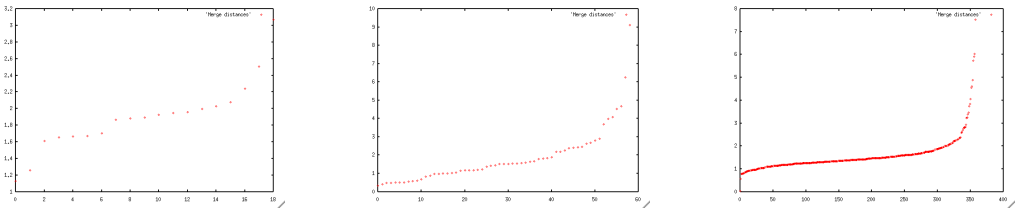
Figure 6: Dendrogram of 20-newsgroups

We can observe from the dendrogram the semantic similarity of classes in confusion space. We propose to use the dendrogram to find the best grouping of classes at a top level. Note that from the root of the dendrogram, as we proceed to the left of the figure beyond a base distance of 1.0, the number of clusters merging per unit distance change drastically. We can plot the distances at which clusters merge against the merge numbers for the dendrograms of each dataset. Cluster merge distances for the 20-newsgroups, Reuters, and ODP datasets are given in Figure 7.

We clearly note a point of inflection in the cluster merge distance plots. Beyond this point of inflection, there are a very small number of highly dissimilar clusters. This can be explained by the nature of Ward's algorithm for hierarchical agglomerative clustering. Consider some vector space with an underlying pattern of clusters. Very similar clusters are consolidated first as the algorithm proceeds. Toward the end of the HAC run, we will reach a stage when huge clusters are left to be merged but the distance between them is very large. This indicates that these clusters are not similar to each other. This analogy can be applied to our clustering of classes in confusion space. After the point of inflection, only highly dissimilar groups of classes are left to be merged. For the ODP dataset, we see this point to occur at merge number 335. Since there are 359 classes, we are left with 22 top level clusters of classes largely dissimilar to each other. Similarly, the 20-newsgroups dataset gives us 5 groups and the Reuters dataset gives us 8 groups. These are the groups we use for the multi-level classifiers we build as detailed in section 2.

4 Graph based method using binary classifiers

The breaking up of a multi-class classification task into a set of binary classification tasks is a well studied technique. This technique is employed in Support Vector Machine classifiers. In standard binary SVM classifiers, the aim is to find the maximum margin hyper-plane separating instances of two classes. Two techniques are used to solve multi-class problems using SVMs. The standard technique for N-class SVMs is to construct N one-vs-others binary SVMs, 1 for each class. The winning classifier is the one that gives the maximum positive distance to the test instance from the separating hyper-plane. In the other technique, all possible pairs of classes are used in making binary SVM classifiers. A majority voting scheme is followed, wherein each classifier votes for it's winning class, and the winner is the class with the largest number of accumulated votes or largest sum of margins as output by the classifiers. Recent work along these lines is the use of binary SVMs to construct



(a) 20-newsgroups with 20 classes

(b) Reuters with 60 classes

(c) ODP with 359 classes

Figure 7: Merge distances for the datasets plotted against cluster merge number

Decision DAGs [PCST00]. Experiments with DAGSVM showed it to be at least as accurate as the other SVM multi-class methods and sometimes better in terms of accuracy and also slightly better in terms of running time and kernel evaluations [PCST00].

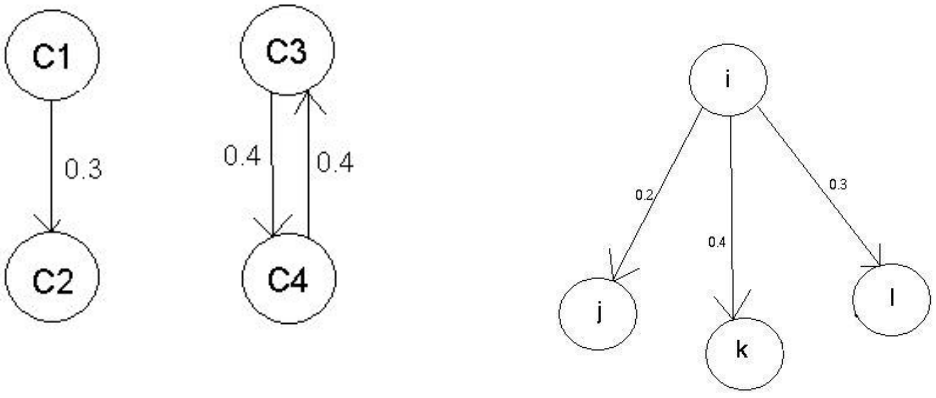
	C1	C2	C3	C4
C1	1.0	0	0	0
C2	0.3	0.4	0.2	0.1
C3	0	0.1	0.5	0.4
C4	0	0.1	0.4	0.5

Figure 8: Normalized 4x4 confusion matrix

	C1	C2	C3	C4
C1	0	0.3	0	0
C2	0	0	0	0
C3	0	0	0	0.4
C4	0	0	0.4	0

Figure 9: Equivalent Incidence matrix

We introduce an algorithm to efficiently solve multi-class classification problems. From the confusion matrix, for a given class, we can find which other classes its test instances get misclassified into. This can be very efficiently done with reasonable accuracy using a Naive Bayesian classifier. Let C_{ij} denote the entry in the i^{th} row and j^{th} column in the Confusion matrix. If C_{ij} is above a threshold t, say 5% of the sum of all entries in the i^{th} row, then we put the value of C_{ij} in the position I_{ji}, where I_{ji} denotes the entry in the j^{th} row and i^{th} column of the Incidence matrix, which is defined in the following. Thus we are collecting in the same j^{th} row in the incidence matrix, the classes which are likely to be confused with class j. The incidence matrix then directly leads us to a weighted directed graph, where a non-zero value in I_{ji} indicates an edge from node j to i. A sample normalized example confusion matrix is shown in table 8 and its corresponding incidence matrix is shown in table 9. The corresponding incidence graph is shown in figure 10(a).



(a) Sample from Confusion Matrix

(b) A more general case

Figure 10: Incidence Graphs

Approach 1: The graph in figure 10(a) can be interpreted as follows. If instances of class i are confused with a class j as per the threshold parameter, then the node j in the graph points to all such i-like classes where j causes confusion. During the testing phase of the classifier, we run binary classifiers between a node and each of the nodes pointed to by arcs

going out it. Thus, if an instance is predicted as belonging to class i, then we must check this against all ij combinations indicated in the I matrix.

Special cases: We will often get graphs which are more complicated in connecting similar confusing classes. A more realistic sub-graph is shown in figure 10(b). In this graph, i has 3 outgoing edges to nodes j, k, and l. This means that when a classifier predicts a test instance to belong to class i, we must further check this with ij, ik, and il binary classifiers.

1. If all binary classifiers label the test instance as i, then the test instance is labeled as i.

2. If exactly one classifier gives a contrary prediction (say k in figure 10(b)) but all other classifiers predict i, then we give the test instance the class label k. We justify this by saying that ik is a classifier with the ability to finely judge between instances of class i and k. The other classifiers choose i because they do not have the ability to make fine decisions on problem cases like the test instance. The ik classifier has more discriminatory power in deciding whether the instance belongs to class i or not.

3. If all but two classifiers (say k and l in figure 10(b)) predict the test instance to be i, then the final class of the test instance is the result of running a binary classifier between the two disagreeing classes i.e. the result of a kl binary classifier.

4. In case there is more disagreement between 3 or more binary classifiers, we will simply pick the prediction of the classifier with the highest weight edge from i. In the figure 10(b), if the ij classifier predicts j, the ik classifier predicts k, and the il classifier predicts l, then we will choose the prediction of the ik classifier because ik is the edge with the highest weight.

Maximum possible improvement: The maximum possible improvement by this method is bounded by the Confusion matrix. If the test instance is classified into a class node from where there is no outgoing edge to the true class, then that test instance cannot be salvaged by this method.

Approach 2: Some of the special cases in the above approach do not stand to closer inspection. Cases 3 above is a case in point. Consider a pool based tournament with an all-pairs contest scenario. If player A wins her matches against all opponents except player B, but player B has won some and lost some of her other matches, it is hard to determine who is a better player between A and B. Some contention resolving scheme like margin of victories or total points scored is then used to pick a winner. This can get cyclically complicated if B wins all her matches except one, say against some C.

In our second approach we identify the incidence graph in the same manner as in approach 1. Following this however, we run binary SVM classifiers amongst the nodes i and any other nodes it points to in one of the following configurations.

Max-Wins: All pairs of classes in the sub-graph originating from the predicted class i are considered and individual winning classes of the binary classifications are considered. For each class, the winning margin in each binary classification is considered. The winning class is the one with the highest sum of margins.

One-vs-Rest: In a sub-graph of k number of classes originating from the predicted class i, k one-vs-other classifiers are constructed. The result of each binary classification is a real number, with positive numbers indicating that the solitary class has won, and a negative number indicating that the group of 'other' classes has won. The winning class is the one with the largest real value of the margin. In case all k one-vs-rest classifiers return negative numbers, the above heuristic still holds. It means that the classifiers with

the largest negative margins were most confident about the test instance not being of the solitary class. This is the method we use in subsequent experiments.

5 Experiments

5.1 Datasets

20-newsgroups: The 20-newsgroups dataset [20N] is a collection of 20,000 news wire articles from 20 Usenet groups containing 1,000 articles each. This dataset is not pre processed into training and testing sets. We randomly chose 70% of the documents for training and the remaining 30% for testing. The corpus contained around 75,000 words. A number of features were selected by the mutual information metric. All words were stemmed using a Porter stemmer [Por80], all HTML tags were skipped, and all header fields except subject and organization of the posted article were ignored.

Reuters-21578: The Reuters-21578 Text Categorization Test collection [Reu] is a standard text categorization benchmark. It contains 135 classes. We choose only those 60 classes which have more than 10 training documents. This resulted in 8819 training documents and 1887 testing documents. The Reuters dataset comes pre-processed in that training and testing documents are separated according to some standard known split criteria. We used the Mod-Apte split but further ignored multi-class test instances. We ignored multi-class test instances because we wanted to see if confusion amongst classes can be resolved by using multi-level classifiers. For features, XML tags were ignored and the words were stemmed.

The ODP dataset: The Open Directory Project crawled dataset is a crawl of the ODP, a popular Internet directory [1]. The crawl consists of 177,600 documents from 359 classes. The crawl took categories from the ODP tree where approximately 1,000 documents were present. The classes included both internal nodes and leaf nodes of the taxonomy. In the automatic hierarchy generation experiments reported subsequently, all classes were considered participants in a flat classification scheme where parents classes from the directory directly competed with their children, if any. Again, for these HTML pages, HTML tags were skipped and the words were stemmed.

Experimental setup: The 20-newsgroups and Reuters datasets were used in experiments with multi-level classifiers. All datasets were used for automatic generation of class hierarchies. Only the 20-newsgroups dataset was used to evaluate the binary graph based method using binary SVM classifiers. All experiments were performed on a $P4$ $1.4GHz$ machine with 512 MB RAM running Linux. The software used for Naive Bayes and multi-class SVM experiments was *Rainbow* [McC96]. For binary SVM experiments SVM_Light [Joa99] was used.

5.2 Results

Multi-level classifiers The accuracies for the multi-level classification technique are shown in tables 11 and 12. *All* is a flat naive Bayes classifier from which the initial confusion matrix is obtained. $L1 - Root$ denotes the first level classifier which only classifies test

[1] http://dmoz.org/

Dataset	Group	Method	Accuracy
20ng	All	NB	81.69%
.	L1-Root	NB	88.19%
.	L2	NB	88.99%
.	L2	SVM	89.82%
Reuters	All	NB	69.23%
.	L1-Root	NB	85.07%
.	L2	NB	76.51%
.	L2	SVM	76.33%

Figure 11: Overall accuracy

Group	Class	Flat	L2-NB	L2-SVM
1	alt.atheism	83.66	84.32	72.92
.	soc.religion.christian	89.13	95.33	93.27
.	talk.religion.misc	42.40	66.67	71.93
2	comp.graphics	84.00	90.13	85.86
.	comp.os.ms-windows.misc	12.87	21.80	84.53
.	comp.sys.ibm.pc.hardware	79.13	95.60	85.07
.	comp.windows.x	84.00	88.07	86.13
3	comp.sys.mac.hardware	88.87	95.73	90.73
.	misc.forsale	78.93	91.33	89.93
.	sci.electronics	79.27	93.27	89.80
4	rec.autos	91.60	95.86	95.06
.	rec.motorcycles	95.00	97.00	94.80
.	rec.sport.baseball	95.53	97.60	97.73
.	rec.sport.hockey	97.67	98.00	98.47
.	sci.crypt	93.00	99.13	98.13
.	sci.med	91.93	96.80	97.07
.	sci.space	93.07	98.33	96.80
5	talk.politics.guns	89.93	96.40	90.93
.	talk.politics.mideast	91.40	95.07	92.47
.	talk.politics.misc	72.40	83.40	84.67

Figure 12: Details for 20-newsgroups

Figure 13: Experiments with multi-level classifiers

Method	Threshold t	Running time (mins.)	Acc%
MC NB	-	1.5	79.50
MC SVM	-	115	84.05
Graph SVM	0.03	50	83.33

Figure 14: Performance and Accuracy

No.	Class	MC NB	MC SVM	Graph SVM t=0.03
1	alt.atheism	81.67	71.91	74.58
2	comp.graphics	73.33	79.33	81.00
3	comp.os.ms-windows.misc	09.33	83.00	74.00
4	comp.sys.ibm.pc.hardware	82.33	75.33	80.00
5	comp.sys.mac.hardware	83.67	87.67	84.33
6	comp.windows.x	80.00	80.00	79.67
7	misc.forsale	71.33	77.67	78.67
8	rec.autos	90.00	89.33	91.33
9	rec.motorcycles	94.33	95.33	90.00
10	rec.sport.baseball	91.33	92.67	94.67
11	rec.sport.hockey	96.67	98.00	92.67
12	sci.crypt	92.00	96.33	92.00
13	sci.electronics	76.00	79.00	81.00
14	sci.med	86.33	94.33	86.33
15	sci.space	90.33	93.00	90.33
16	soc.religion.christian	89.33	92.00	89.67
17	talk.politics.guns	88.33	86.00	85.33
18	talk.politics.mideast	90.67	91.33	92.33
19	talk.politics.misc	71.00	69.00	68.33
20	talk.religion.misc	52.00	49.67	60.33

Figure 15: Details for 20-newsgroups

Figure 16: Experiments with graph-based method

articles into on of the sub-groups of classes. $L2$ are the second level classifiers trained on a particular sub-group. The sub-groups are obtained from the dendrograms as explained in section 3.

Automatic generation of hierarchies The dendrogram for the entire 20-newsgroups dataset is already shown in figure 6. A partial dendrogram for the Reuters dataset is shown in figure 17 in the appendix. Snippets from the dendrogram for the ODP dataset are shown in figures 18, 19, and 20 in the appendix.

Graph based method using binary classifiers The running time and accuracy figures for the graph-based method of binary classifiers are given in tables 14 and 15. $MC\,NB$ is a multi-class naive Bayes classifier. $MC\,SVM$ is a multi-class SVM classifier. The graph-based method of binary classifiers is implemented as the one-vs-rest method of approach 2 in section 4.

5.3 Observations

Automatic generation of hierarchies From the dendrograms of figures 6, 17, and 18 19 20, we see an excellent semantic grouping of classes. For the 20-newsgroups dataset, as expected we see that newsgroups pertaining to the religious topics are clustered together. Similarly newsgroups pertaining to computers, science, and recreation are also clustered together. This is in line with our expectations and very closely matches the manual reorganization we did in figure 2.

For the Reuters dataset, we see the clustering of the classes related to finance together. The classes dlr, yen, dmk, money-fx, interest are clustered together. The classes barley, corn, grain, wheat, cotton, rice, orange and sugar are clustered together, being classes related to foodstuff. Economic indicator classes like cpi, wpi, gnp, jobs, housing are also grouped together.

For the ODP dataset too we see similar interesting results. The interesting clusters for figure 18 are

- Recreation/Antiques, Recreation/Collecting, Shopping/Antiques and Collectibles
- Games/Puzzles, Shopping/Children, Shopping/Toys and Games, Home/Kids, Society/Holidays

In figure 19, the most striking result is the grouping together of all the Sports classes. In figure 20, the interesting clusters of some of the classes are

- Computers/Internet/WWW, Reference/Directories, Reference/Archives, Reference/Libraries, Reference/Geography
- Computers/Education, Computers/Internet
- Computers/Open Source, Computers/Software/OS/Linux
- News/Weather, Science/Earth Sciences, Reference/Maps, Regional/Polar Regions, Science/Methods and Techniques.

We observe that classes from completely separate parts of the original ODP tree come together based on their similarity. The best example of this is the News-Reference-Science-Regional cluster in figure 19. Documents about News/Weather, Science/Earth Sciences, Reference/Maps, and Regional/Polar Regions are bound to talk about similar things, contain the same words in comparable frequencies, and often be confused with each other in classification. This is exactly the motivation of 'soft-links' in Internet directories like the ODP. Since a particular top level theme like News/Weather prevents documents about Science, Reference, or Regional from being put together, soft-links are used which are characterized by the 'Also See' links we find in the directories. The ODP is a manually populated directory and our technique by identifying clusters as above can suggest the soft-links to be put to provide a better browsing experience to the user. This holds good for the Shopping and Regional sub-trees also which are known to be extremely miscellaneous in nature, and our dendrogram identifies similar classes based on co-confusion and not on their actual location within the ODP tree.

Graph based method using binary classifiers We can draw some interesting observations from tables 14 and 15. Multi-class Naive-Bayes is the fastest in terms of running time and has a mediocre accuracy of 79.50% for the 20-newsgroups dataset. Multi-class SVMs have a far better accuracy of 84.05%, but at the same time take a long time to train. The graph-based method employing binary SVM classifiers, is in the middle in terms of training time. The accuracy of this new method is very good at 83.33% which is comparable to multi-class SVMs. This slight accuracy disadvantage is offset by the gain in performance and training times that we observe. Especially on the issue of scalability, for extremely large datasets like crawls of public Internet directories, running multi-class SVMs will be impractical. Our method however requires only multiple binary classifiers to be built, which is fast. Moreover, constructing an Incidence matrix is a one time job, given the confusion matrix for the multi-class Naive-Bayes classifier. The accuracy improvement of our method over multi-class Naive-Bayes justifies building the multi-class Naive-Bayes classifier to get it's confusion matrix to give as input to the Incidence matrix construction step.

For any given classification task, we spend some time in the beginning exploring the optimal feature set size to use. We do the same in case of our two multi-class classifiers. For our graph-based method, we would still like to do optimal feature set selection to get the best possible advantage of the constituent binary classifiers. However, this is difficult to do for all the varying sizes and configurations of the binary classifiers. Still, choosing a reasonable number of features, 3000 in this case, we achieve significant accuracy and performance benefits.

6 Conclusion

We have seen some interesting uses of the confusion matrix in identifying similar sets of classes. The automatic generation of hierarchies as dendrograms gives very interesting results as we have observed. As a part of future work, this needs to be directly compared with word based hierarchical document clustering to see the quality of the resultant clusters. The notion of intermediate levels of similar classes can lead to interesting future work on approximately labelling test instances amidst lot of confusion. When the classifier cannot confidently label a test instance into one of the learned target concepts, we can predict the instance to belong to one of the intermediate group of similar classes.

The graph based method of breaking up a multi-class problem into sets of binary subproblems yields excellent performance gains and gives accuracy comparable to the highly accurate multi-class SVM techniques. Hence, we see that using the confusion matrix in these various ways gives us some useful techniques to scale up large scale multi-class classification systems.

References

[20N] The 20 newsgroups dataset at http://www.ai.mit.edu/~jrennie/20_newsgroups/.

[CDAR98] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal: Very Large Data Bases*, 7, 1998.

[Cha00] Soumen Chakrabarti. Data mining for hypertext: A tutorial survey. In *ACM SIGKDD Explorations, 1(2)*, 2000.

[EHW86] A. El-Hamdouchi and Peter Willett. Hierarchic document clustering using ward's method. In *Information Processing and Management*, 1986.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1998.

[Joa99] Thorsten Joachims. Svmlight - support vector machines. http://svmlight.joachims.org/, 1999.

[JW98] Johnson and Wichern. *Applied Multivariate Statistical Analysis*, chapter 12. Clustering, Distance Methods, and Ordination. Prentice Hall, India, 1998.

[McC96] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[PCST00] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification, 2000.

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program, 14(3):130–137*, 1980.

[Reu] The Reuters-21578 Text Categorization Test Collection at http://www.research.att.com/∼lewis/reuters21578.html.

7 Appendix

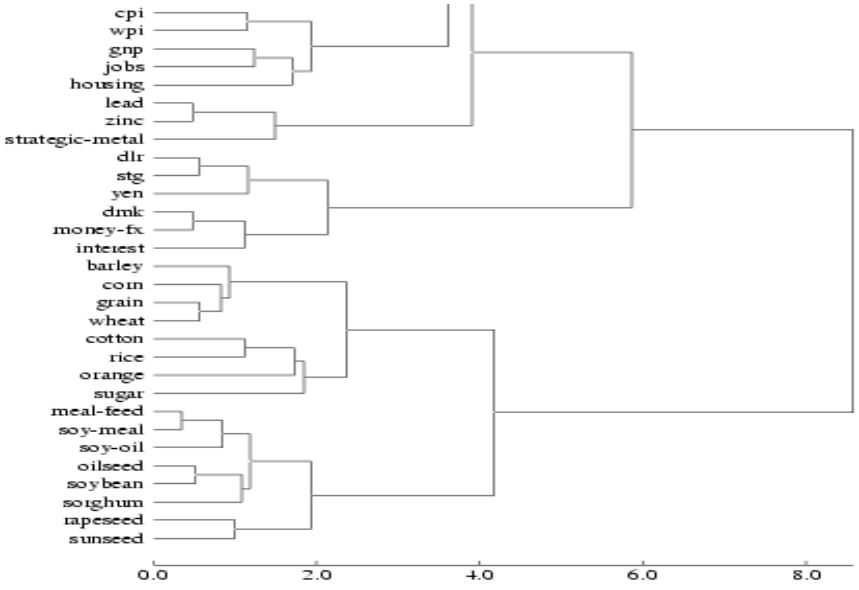


Figure 17: Partial dendrogram of Reuters

Figure 18: Partial dendrogram of ODP - 1

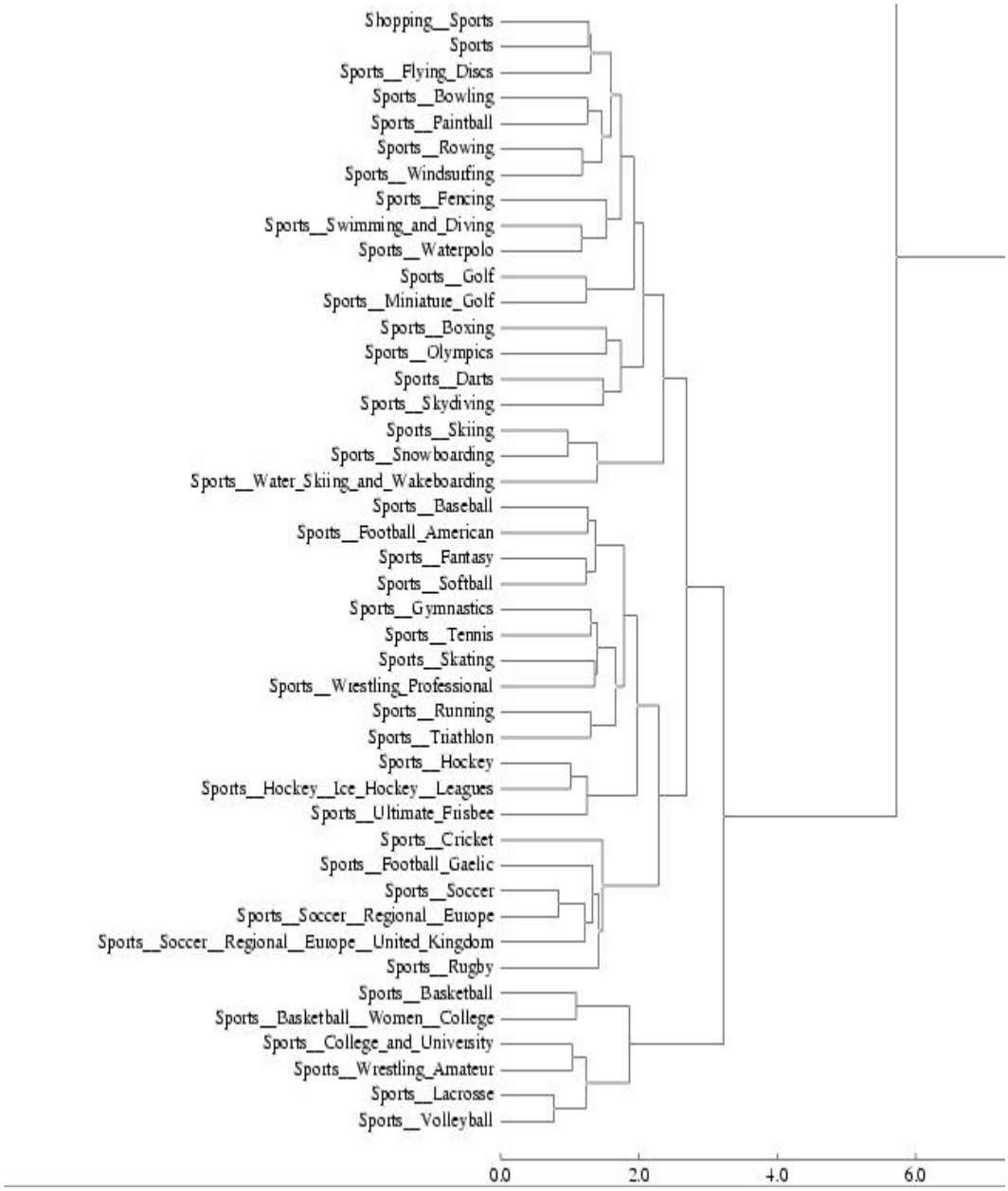


Figure 19: Partial dendrogram of ODP - 2

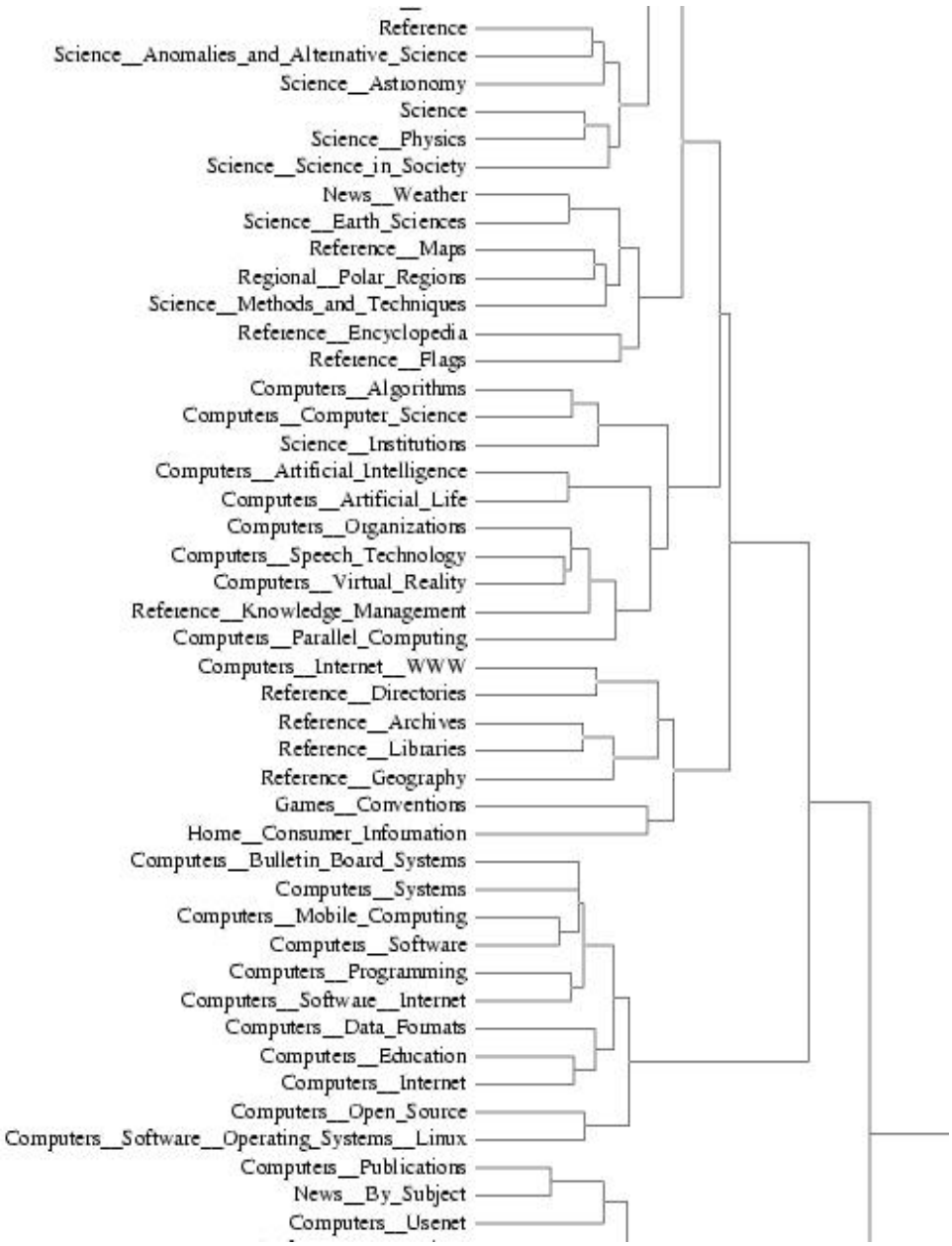


Figure 20: Partial dendrogram of ODP - 3