ANALYSIS OF STRUCTURE-BASED TECHNIQUES FOR PREPROCESSING THE INDUCED GRAPH OF MASSIVE DATASETS FOR TRANSDUCTIVE LEARNING

Aishwarya P (CS11B004), Dhivya E (CS11B012), S K Ramnandan (CS11B061)

***Abstract*—The proliferation of the Internet has resulted in a rapid increase in the production of published information. This has led to the emergence of a large class of machine learning problems where the challenge has shifted from scarcity of data to lack of labelled or reliably labelled data. Such tasks are not amenable to inductive learning techniques. In these and many other situations, transductive learning is good alternative. The aim of this project is to compare the propagation of labels in a true underlying graph structure and a hypergraph representation of the data, derived purely from non-graphical attributes. This project focusses on identifying points whose labels are particularly important for classification, a result which could be used in active learning.**

I. Introduction

TRANSDUCTIVE is a method of reasoning from specific training cases to specific test cases, in contrast to inductive learning where the aim is to obtain a generic functional mapping from an input space to an output space.

A simple situation in which transductive learning techniques would be preferred is a problem for which the training data and many test instances are available a-priori but the number of labelled training instances given is far less than the number of available unlabelled test instances. As most induction methods rely on the availability of an extensive, representative training set, an inductive learning function is unlikely to be able to capture all the facets of the input space. Further, in such a problem, as the test set is known beforehand, a transductive algorithm, which by its nature would be tailored to the specific instances available, would possibly construct a richer model than that which a generic (inductive) approach would create. Hence, the transductive algorithm could be expected to perform better.

Another situation in which transductive learning could be useful is a binary classification problem in which inputs naturally tend to cluster into two groups, based in their class labels as seen in figure 1.

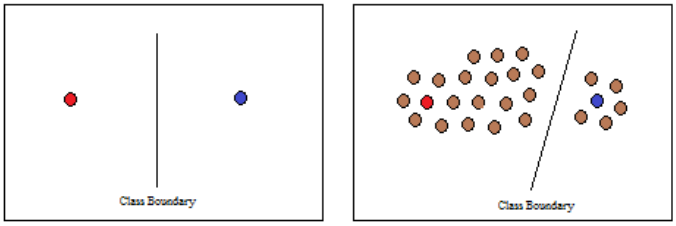

Fig. 1. Inputs cluster based on classes

Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query the user (or some other information source) to obtain the desired outputs at specific data points. This project is aimed at identifying important points, which can be used for such queries, when the data is represented using a graphical model. One of the main benefits of such a representation is that it aids visualization of the data. Further, many properties of graphs such as degree, connectedness or clusters provide rich information that would otherwise remain latent. It is intended to compare the performance of graph-based classification on a true underlying graphical structure and an induced graphical structure for the same dataset. This would enable creation of a general mechanism by which a graphical representation can be created for a dataset and suitable points can be identified from thsi as being important from the point of view of active learning.

II. Dataset and Data Model

The dataset used for this project was the Genes dataset obtained from http://pages.cs.wisc.edu/ dpage/kddcup2001/. This dataset is about genomes of many animals that have been completely sequenced. Interest within bioinformatics is shifting somewhat away from sequencing, to learning about the genes encoded in the sequence. Genes code for proteins, and these proteins tend to localize in various parts of cells and interact with one another, in order to perform crucial functions. The present data set consists of a variety of details about the various genes of one particular type of organism. The aim of this project if to predict the localization of the

gene. The other information provided includes the class of the gene/protein, whether it is essential, the phenotype (observable characteristics) of individuals with a mutation in the gene (and hence in the protein), the motif, the chromosome, and the other proteins with which each protein is known to interact.

The interactions result in a natural graphical representation of the genes on which diffusion techniques could be experimented. It is reasonable to assume that genes connected in this network are likely to have the same localization as genes are more likely to interact with genes in the same region of the cell, which is the localization being predicted.

All the relational attributes of the genes were discrete-valued. Thus, a natural graphical representation using these is a hypergraph. In mathematics, a hypergraph is a generalization of a graph in which an edge can connect any number of vertices. Formally, a hypergraph H is a pair H = (X,E) where X is a set of elements called nodes or vertices, and E is a set of non-empty subsets of X called hyperedges or edges. Here, genes having the same value for an attribute are made to lie on the same hyperedge of the hypergraph.

The relational attributes of a gene, in some cases take multiple values. Thus, the encoding used is to have a binary attribute corresponding to each value of each of the above attributes. Thus, for every such derived attribute, there would be 2 possible hyperedges - the Yes edge and the No edge. Only the Yes edges were included in the final model, because othwrwise, it appears to imply that two genes that do not have a particluar value for an attribute are as correlated as two genes that have the same value. This is not really true as some of the attribues can in fact take a up to a couple of hundred values (specifically the motif).

III. Centrality Measures

A. Degree Centrality

The degree of a node in a graph is one of the most intuitive properties used to determine its importance. While there are issues with using degree to detect influential points, it provides a simple first-cut solution.

The degree of a node in a graph is well-defined. For the attribute-hypergraph, a slightly modified degree calculation is used in this project. Let the weight of a hyperedge be the number of nodes participating in it. Then, the degree of a node is (sum of weights of hyperedges it participates in - number of hyperedges it participates in). The intuition behind this defintion is as follows. The degree is indictive of the number of nodes that can be reached in one step. A node can reach any node on any hyperedge it is a part of in one step. The weight of a hyperedge is thus indicative of teh numebr of nodes reachable using that hyperedge. The number of hyperedges the node participates in is subtracted because this node itself adds 1 to the weight of all the edges it participates in.

B. PageRank or Eigenvector Centrality

Eigenvector centrality is a measure of the influence of a node in a network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. Google's PageRank is a variant of the Eigenvector centrality measure.

The eigenvector centrality for node d is given by

$$x_d = \frac{1}{\lambda} \sum_{t \in N(d)} x_t = \frac{1}{\lambda} \sum_{t \in G} a_{d,t} x_t$$

where $a_{i,j}$ is the entry in the adjacency matrix corresponding to vertices i and j.

The defintion can be extended to hypergraphs as well. The relevant equations then are,

$$\begin{aligned} c_x &= Wy && (1)\\ c_y &= W^T x && (2)\\ \Rightarrow WW^T x &= \lambda x && (3)\\ W^T W y &= \lambda y && (4)\\ with \lambda &= c_1 c_2 && (5) \end{aligned}$$

where W is the matrix whose entries correspond to the weights of edges between two nodes and c_x and c_y are the centralities of the node x and the edge y respectively.

C. Relative Coverage

Footprint-based retrieval is a novel retrieval technique that is employed in case base systems today. The idea that is central to this method is a measure called *relative coverage*, which is used to obtain a subset of the case base (called *footprint set*) that provides the same coverage as the case-base as a whole. We first describe that concept of relative coverage in the context of case base retrieval systems and then present the adapted version that we have used in our implementation.

In the context of Case Base Systems

Consider a set of cases, C, and a space of target problems, T. A case, $c \in C$, can be used to solve a target, $t \in T$, if and only if two conditions hold. Two important competence properties are the coverage set and the reachability set which are defined as follows.

$$\text{Coverage Set}(c) = \{c' | c' \in C \text{ and } c \text{ solves } c'\}$$

$$\text{Reachability Set}(c) = \{c' | c' \in C \text{ and } c' \text{ solves } c\}$$

The size of the coverage set of a case is only a measure of its local competence. For instance, case coverage sets can overlap to limit the competence contributions of individual cases, or they may be isolated and exaggerate individual contributions. It is actually possible to have a case with a large coverage set that makes little or no contribution to global competence simply because its contribution is subsumed by

the local competences of other cases. At the other extreme, there may be cases with relatively small contributions to make, but these contributions may nonetheless be crucial if there are no competing cases.

$$\text{Relative Coverage}(c) = \sum_{c' \in \text{Coverage Set}(c)} \frac{1}{|\text{Reachability Set}(c')|}$$

For a true picture of competence, a measure of the coverage of a case relative to other nearby cases, is needed. This is where relative coverage comes in. It essentially weights the contribution of each covered case by the degree to which these cases are themselves covered. The importance of relative coverage is that it provides a mechanism for ordering cases according to their individual, global, competence contributions.

In the context of our dataset

In both our classifying techniques (as will be elaborated in a later section), we use the interaction data to recursively obtain the label of a node from the label of its neighbors. So, the preprocessing step can be viewed as the construction of the footprint set of nodes from an input training data. In the context of our interaction graph $G = (V, E)$, the above definitions can adapted as follows:

$$\text{Coverage Set}(v) = \text{Reachability Set}(v) = N(v)$$

$$\begin{aligned}\text{Relative Coverage}(v) &= \sum_{u \in \text{Coverage Set}(v)} \frac{1}{|\text{Reachability Set}(u)|} \\ &= \sum_{u \in N(v)} \frac{1}{\text{degree}(u)}\end{aligned}$$

Once this is calculated for every node, we just have to pick the k nodes with the highest relative coverage values as the most influential nodes in the graph and use them in our classification algorithms.

IV. Classification on a graph

As the data model used was graphical, suitable classification methods were needed that utilize structural properties of a graphical representation to classify points. In this regard, the following mechanisms were considered -

- Graph Diffusion
- Label Propagation

A. Graph Diffusion

This is an adaption of the method in [3]. the probabilistic formulation of the classification problem and uses a relaxation labeling technique. Given a graph G where each node d is an input point and the set of labels C, this thenique allows classification of nodes based on a prior probability $\pi(d, c)$, which is the prior probability of node d belonging to class c, using information from the neighbourhood $N(d)$ of node d.

Let $\phi(d, c)$ represent the probability of node d belonging to class c, given the graph G and the prior, $\pi(d, c)$. Assuming that the label of a node (as a random variable) is conditionally independent of the labels of other nodes in the graph given the labels of its immediate neighbors (MRF assumption), $\phi(d, c)$ only depends on the values of $\pi(d, c)$, $\phi(d', c)$ for all nodes $d' \in N(d)$ and the probability that a node d belongs to class c, given that it's neighbour belongs to class c'. This probability can be treated as independent of the specific node d and d' involved, in which case, it is sufficient to have the probability $\chi(c, c')$ which is the probability that a node has label c, given that it's neighbour has label c'.

For tractability, the additional independence assumption that there is no direct coupling between the prior probability of a node belonging to a particular class and the labels of its neighbors, the following central equation holds.

$$\phi(d, c) = \pi(d, c) * \prod_{d' \in N(d)} \left(\sum_{c' \in C} (\phi(d', c') * \chi(c, c')) \right)$$

This is because $\sum_{c' \in C} (\phi(d', c') * \chi(c, c'))$ is the probability of node d being in class c, considering all possible label assignments to its neighbour d'. Treating the labels of these nodes and the prior as independent contributors towards the probability of the label of the node of interest, the above equation is obtained.

Then, $\phi(d, c)$ can be computed in an iterative RL manner as follows. Let $\phi^{(r)}(d, c)$ represent the probability of node d belonging to class c at the end of round r. Then,

$$\phi^r(d, c) = \pi(d, c) * \prod_{d' \in N(d)} \left(\sum_{c' \in C} \left(\phi^{r-1}(d', c') * \chi^{r-1}(c, c')\right) \right)$$

In the above update rule, χ is also assumed to vary. This is because no a-priori information in available about the the probability of a node d belonging to class c, given that it's neighbour belongs to class c'. This information is also inferred from the graph by a smoothed estimator based on the frequencies of edges tentatively labeled with c, c' in the previous round (using Laplace smoothing).

B. Label Propagation

Label propagation is a popular semi-supervised classification method.The basic idea of label propagation is to use random walks in order to propagate labels through the graph. We assign a normalized vector F_i to every node i. The jth component of F_i represents the probability with which node i belongs to class j. Also, a weight matrix is used. Weight w_{ij} is the fraction of hyperedges containing both node i and node j. The label propagation algorithm then iteratively re-weighs the F-vectors for all nodes till convergence.

Algorithm IV.1: ITERATIVE LABEL PROPAGATION()

repeat
 for each $v \in \mathcal{V}$
  do
 $F_v = \sum_{(v,u)\in E} w_{uv} F_v$
 $Normalize(F_v)$
until $convergence$ **or** $maxIterations$

A more detailed treatment of label propagation on graphs may be found in [2].

When executing the algorithm in practice, we often introduce a stepping factor γ. Instead of adding the weighted sum of F vectors across neighbours with a weight of 1, we take a slightly shorter step which is a fraction γ of the earlier step. This helps prevent overfitting, since we do not take sudden steps towards a local optimum.

$$F_v = \gamma \sum_{(v,u)\in E} w_{uv} F_v \tag{6}$$

V. PREPROCESSING OF THE DATASET

The given dataset contained a combination of the relational attributes and the interaction data in the file *Full_file.data*. Thought the relational data was separately present in the file *Genes_relation.data*, the data format in this file was not convenient for programmatic usage. Despite the fact that some genes had more than one value for some nominal attributes, a binary representation was not used. Also, there was no clear demarcation between the values of one attribute and those of another. Thus, the required relational data was extracted from *Full_file.data* and stored in *UsefulRelationData.txt*. During this process, the attribute values were replaced by more concise forms if needed and converted to a standard form (for example, in some records attribute values were enclosed in quotes, whereas in other they were not) to enable simple parsing in the programs. The interaction data could be used as in from *Genes_interaction.data*.

The next stage involved creation of the hypergraph using the relational attribute values. For those nominal attributes for which each node had only one value, a hyperedge could be created for each value and a node included in it if it had that value for the attribute. For those nominal attributes that had to be encoded as binary attributes, one for each possible value, as a node could potentially take multiple values, an edge was created for each such value and all nodes that had a "Yes" corresponding to the binary attribute (which implied that one of the values for that nominal attribute for that node was the value corresponding to the hyperedge) were included in the hyperedge. Intially, for each such attribute, a hyperedge was created for the "No" instances as well but this was discarded because it resulted in every node becoming a neighbour of nearly every other node (where a neighbour of a node is any node that is present in at least one hyperedge this node is a part of). This can be explained by the fact that one of the nominal attributes, the motif, takes a couple of hundred possible values, but a single gene typically has only a few motifs, generally just one. Thus, using every such "No" edge, almost all genes would become neighbours, which does not indicate the true nature of the relation between genes.

A. Distribution of Node Properties in the Dataset

From the graphical representation, it is possible to observe trends of node properties such as degree and importance (as calculated by a form of PageRank).

The following is the distribution of various classes in the dataset -

Class	Count
Nucleus	366
Plasma membrane	43
Peroxisome	10
Endoplasmic Reticulum	43
Cytoplasm	192
Cytoskeleton	58
Integral membrane	3
Mitochondria	69
Vacuole	18
Golgi	35
Transport vesicle	17
Cell wall	1
Lipid particles	1
Extracellular	2
Endosome	4

Due to the heavy class skew and the large number of classes, multi-class classification was difficult on the dataset. The large fraction of points in the "nucleus" class prompted us to attempt one-vs-rest classification on this class as an alternative test-bed for the different experiments.

Some other properties of the data distribution observed are as follows. The distribution of the degree of the nodes of the graph (true underlying graph obtained from interaction data) is shown in figure 2.

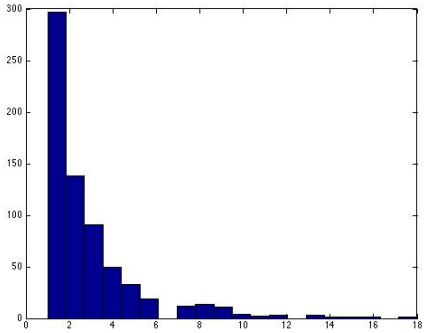

Fig. 2. Degree distribution of true graph structure

Since many points are of low degree and only a few points are of high degree, it appeared reasonable to treat high-degree

points directly as important points, without searching for the more exact influence maximization criterion of number of neoghbours not already covered by a selected node.

The distribution of degrees of the nodes of the hypergraph, calculated as decsribed earlier, are shown in figure 3.

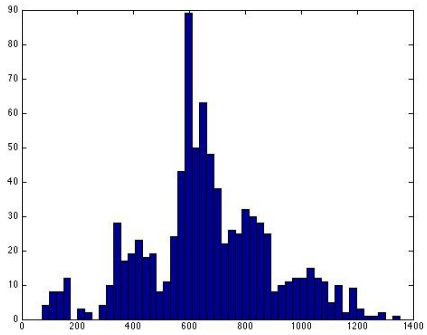

Fig. 3. Degree distribution of induced hypergraph

Again, there appear to be sufficiently few points of very high degree, thus enabling it to be directly used as a centrality measure.

The distribution of PageRank of the nodes of the hypergraph is shown in figure 4.

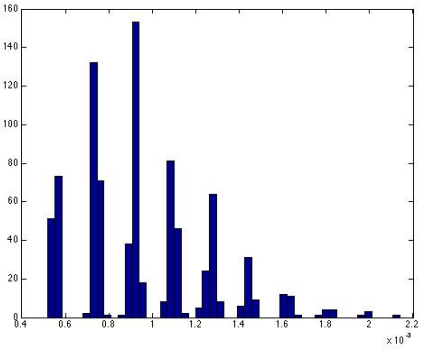

Fig. 4. PageRank distribution of induced hypergraph

The distribution of relative coverage of the nodes of hypergraph is shown in figure 5.

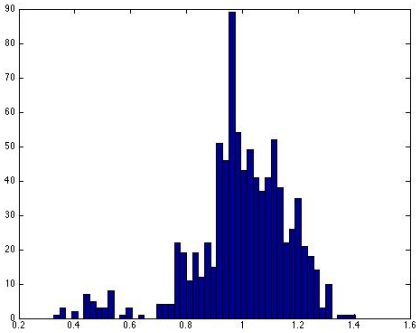

Fig. 5.

For a basic benchmark to compare the classification results, the Naive Bayes classifier from Weka was used. This obtains an accuracy of 61% on the train set. The Naive Bayes classifier was chosen for comparison because it was one of the base classifiers that was designed to work well with nominal attributes and missing values.

VI. Experiments

Firstly, the classification was done on the true graph, using Graph diffusion, by selecting the training set using degree as the centrality measure. To check the performance, this was compared with the choice of a random training set. Figure 6 shows the results for different train set sizes.

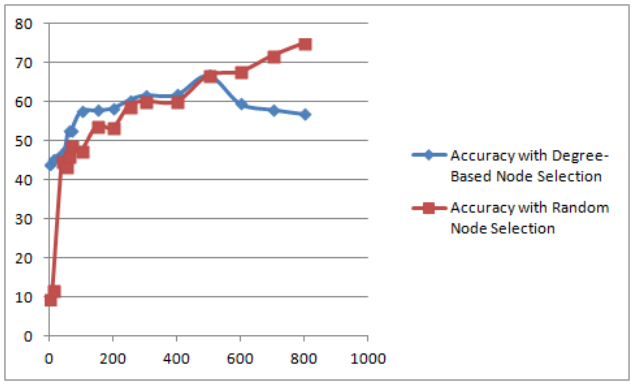

Fig. 6.

It was observed that for small training set sizes, choosing the training points using degree centrality provides a better result. This confirms the fact the fact that degree centrality can be used as a criterion to select points to query for labels for active learning, since the query set is expected to be small. For larger training set sizes, the degree criterion may not work well because it is probable that points of intermediate degree are neighbours of each other. Thus, the set of important points may not be well-spread out in the graph. However, this can be expected to happen with a random choice of train set, hence the performance of a random train set scales better. However, as the aim of this project is to identify training sets for active learning, which are preferably small, the train set

obtained using centrality is preferable.

Following this, a comparison was done between graph diffusion on the graph and label propagation on the hypergraph (the methods which worked better on the respective representations) for One-vs-rest classification for the nucleus class, using a trainging set based on degree centrality. The results are in figure 7.

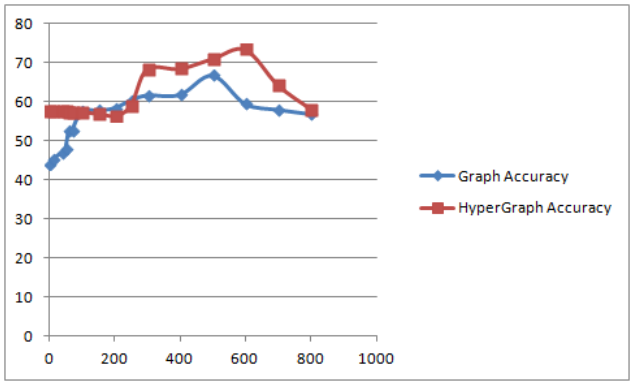

Fig. 7.

It was observed that for most training set sizes, the results on the hypergraph were much better. Thus, the hypergraph model appears to be a suitable graphical model for the dataset, at least in the context of active learning. Further experiments on different centrality measures were then restricted to this model.

The next set of experiments involved the comparison of the different centrality measures - degree (as defined above for a hypergraph), PageRank and relative coverage. The results are displayed in figure 8.

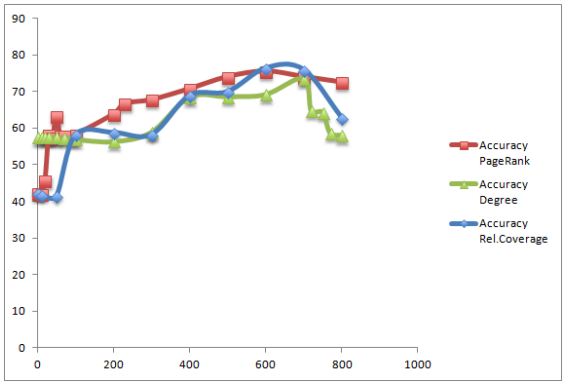

Fig. 8.

It was observed that the best results are obtained using PageRank centrality to obtain the train set. PageRank centrality was, in fact, expected to perform well because it is designed to measure the importance of a node based on its link structure. Also, relative coverage was expected to perform well because it attempts to decrease the overlap between the influence region of the different points in the train set. In this case it has not resulted in a significant improvement over degree centrality. This is probably because, as there are few nodes of high degree, their coverage areas are possibly sufficiently non-overlapping, without an explicit check for that property.

The same set of experiments were then repeated using the original class labels, that is a full 15-class classification was done. The results for this are shown in figure 9.

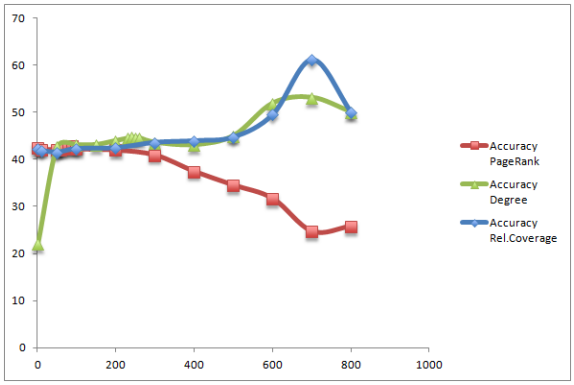

Fig. 9.

On this, it is observed that PageRank did not perform as well as expected. This is possibly because there are too few points with sufficient;ly high PageRank, hence when only these are chosen (very small training set), PageRank performs well but as the number of train points is increased, more non-authoritative points get selected and since label diffusion is being used, important points now have to contend with a large number of weak points to diffuse the correct label. On the other hand, relative coverage works very well, as expected. This is because, in multi-class classification, with such significant class skew, it is important that as many test points as possible are well-covered. This experiment demonstrates the importance of truly performing an influence maximization, as against extreme approxiamtions such as the degree.

VII. Centrality Incorporating Class Imbalance

The general centrality techniques considered so far did not take class information into account. Since there is considerable class imbalance, it is reasonable to assume that there is a need to incorporate class imbalance while picking important training points.
Two proposed mechanisms to handle this -

A. Incorporating class imbalance into the centrality computation

In this method, after the appropriate centrality measure is computed, the centrality value is boosted by the scarcity of finding a training point of that class. Consider any training point with centrality score s. If its class label is c, let the probability of a point from the dataset belonging to class c be $p(c)$. Then the modified centrality score becomes $\frac{s}{p(c)}$.
This method was attempted using relative coverage as a base centrality measure and the classifcation results are as follows

NumInTrain	Acc(%)
100	0.92
200	4.08
300	0.0
400	20.78
500	5.25
600	39.31
700	96.3
800	95.16

It can be seen that there was considerable variation of accuracy and the results were in general not appreciable. This is probably because there are a lot of classes with only 1 member. Thus when the score is divided by the probability of such a class, it shoots up and becomes the most important node. However, this point would not be helpful in classifying other points as there are no other points form its class. Thus, for a small train set size, not enough truly important points are captured, resulting in very poor performance. Whereas for a large train set, Since the scarce class points get forcibly included in the train set extremely high accuracies are obtained.

It can be observed that if the points are sorted according to this centrality measure, they are almost sorted in descending order of scarcity of their class, which is not desirable. Thus, another method was experimented.

B. Sample incorporating class scarcity

Since it was observed that directly incorporating the class probability into the centrality measure was not helpful, the original centrality scores are used. However, when choosing important points, while traversing the nodes in decreasing order of centrality, for any node d, with label c, d is included into the training set with probability $1 - p(c)$.

This method was experimented using relative coverage as the base centrality measure and the results were very impressive.

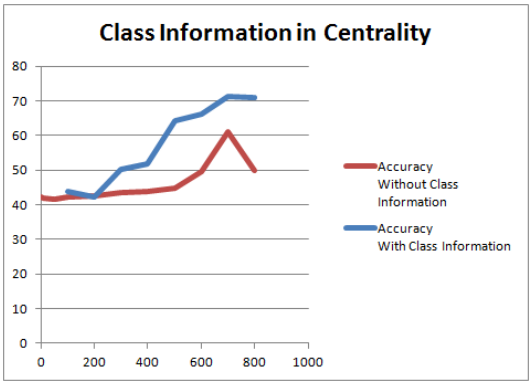

Fig. 10.

It can be seen that the performance is significantly better than the train-set accuracy of Weka's Naive Bayes classifier (known to be 61 %).

Thus it can be concluded that a combination of the information from the structural properties of the graphical model - the attribute hypergraph, captured using relative coverage centrality and the class distribution results in a reasonably strong classifier, that is designed for active learning.

VIII. Future Work

A possible extension of this project is to use the above information to detect noise in class labels. Some possible ideas for this are -

- In the graph diffusion classifer, $\chi(c, c')$ is the fraction of edges between points of classes c and c' respectively. Thus, intuitively, χ values can be treated as indicators of the probability of an interaction. If all interactions of a node are low-probability ones, it is quite possible that the node's label is incorrect. However, this assumption has a shortcoming. Any node interacting with a minor class node will always have a low χ value. So some normalization based on scarcity of points belonging to a particular class is necessary. One possibilty is to divide the χ value by the number of points in the class.

- In the Label Propagation classifier, each node has an F vector. Each component of the F vector denotes probability of belonging to the corresponding class. The F vector is modified through the iterations of the algorithm. Finally, the class with highest probability, as indicated by the F vector is chosen as the label of the point. Consider look at difference between highest and second highest values in F for a node. If that difference is less than some threshold, it is probably a noise point. This is because there are two equally contending classes for the label of that point.

Acknowledgment

The authors would like to thank Dr B Ravindran for their assistance in identifying the project area, topic and the invaluable advice during the course of the project.

References

[1] Barry Smyth, Elizabeth McKenna - *Footprint-Based Retrieval*
[2] Delip Rao, David Yarowsky - *Ranking and Semi-supervised Classication on Large Scale Graphs Using Map-Reduce*
[3] Ralitsa Angelova, Gerhard Weikum - *Graph-based Text Classification: Learn from Your Neighbors*