ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

IMPROVING COLLECTIVE CLASSIFICATION BY INCORPORATING DIRECTED LINKS, FEATURE ENRICHMENT AND CLASSIFIER COMBINATION

Ph.D. THESIS

Abdullah SÖNMEZ

Department of Computer Engineering

Computer Engineering Programme

FEBRUARY 2014

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

IMPROVING COLLECTIVE CLASSIFICATION BY INCORPORATING DIRECTED LINKS, FEATURE ENRICHMENT AND CLASSIFIER COMBINATION

Ph.D. THESIS

Abdullah SÖNMEZ
(504052501)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Doç. Dr. Zehra ÇATALTEPE

FEBRUARY 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YÖNLÜ BAĞLANTILAR, ÖZNİTELİK ZENGİNLEŞTİRME VE SINIFLANDIRICI BİRLEŞTİRME İLE KOLEKTİF SINIFLANDIRMA BAŞARIMININ İYİLEŞTİRİLMESİ

DOKTORA TEZİ

Abdullah SÖNMEZ
(504052501)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Zehra ÇATALTEPE

ŞUBAT 2014

Abdullah SÖNMEZ, a Ph.D. student of ITU Graduate School of Science Engineering and Technology 504052501 successfully defended the thesis entitled **"IMPROVING COLLECTIVE CLASSIFICATION BY INCORPORATING DIRECTED LINKS, FEATURE ENRICHMENT AND CLASSIFIER COMBINATION"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Doç. Dr. Zehra ÇATALTEPE** Istanbul Technical University	.....
Jury Members :	**Doç. Dr. Zehra ÇATALTEPE** Istanbul Technical University	.....
	Prof. Dr. Ayşe ERZAN SİLİER Istanbul Technical University	.....
	Doç. Dr. Şule GÜNDÜZ ÖĞÜDÜCÜ Istanbul Technical University	.....
	Yrd. Doç. Dr. Mustafa BİLGİÇ Illinois Institute of Technology	.....
	Yrd. Doç. Dr. Fatih AMASYALI Yıldız Technical University	.....

Date of Submission : **31 December 2013**
Date of Defense : **13 February 2014**

To my spouse and children,

FOREWORD

First and foremost, I want to thank my advisor Zehra Çataltepe. Not only she helped me understand scientific concepts in detail during my research, but also she listened to me and suggested perfect solutions to my problems as well. Worked as hard as me, sometimes may be harder than me, for our research. I am sure it would be very difficult for me to complete the research without her great help and support.

Second, I want to thank especially to my spouse Emine and children Zehra, İbrahim and Sare. They were always patient and tolerant during my hard studies. My dear mother and father, my biggest supporters during all my life, thank you for everything. You are all the biggest share holders of this research. Thank you!

Third, I want to thank Prof. Dr. Ayşe Erzan and Doç. Dr. Şule Gündüz Öğüdücü for sparing their most valuable ideas and time for my research. The directions they showed, opened new horizons for my research.

I would also like to thank the members of our Machine Learning group at the ITU, especially to Yusuf Yaslan, İsmail Güneş, Eser Aygün, Mahiye Uluyağmur, Kadriye Bağlıoğlu and Barış Şenliol, whose collaboration made research easier and more fun.

February 2014

Abdullah SÖNMEZ
(M.Sc.)

TABLE OF CONTENTS

Page

ABBREVIATIONS

BN	**:** Bayes Net
CC	**:** Collective Classification
CO	**:** Content Only
HET	**:** Heterogeneous
ICA	**:** Iterative Classification Algorithm
kNN	**:** k-Nearest Neighbor
LO	**:** Link Only
LR	**:** Logistic Regression
NB	**:** Naive Bayes
OAA	**:** One Against All
OAO	**:** One Against One
PCA	**:** Principal Component Analysis
RBF	**:** Radial Basis Functions
RF	**:** Random Forest
RS	**:** Random Sampling
SS	**:** Snowball Sampling
SVM	**:** Support Vector Machine

LIST OF TABLES

Page

LIST OF FIGURES

Page

IMPROVING COLLECTIVE CLASSIFICATION BY INCORPORATING DIRECTED LINKS, FEATURE ENRICHMENT AND CLASSIFIER COMBINATION

SUMMARY

Most pattern recognition methods assume that inputs are independently and identically distributed, and they do not handle the dependency between the instances. Classification problems are solved using instances' features (content) and labels. Connections/dependencies/relations between instances are not taken into consideration. On the other hand, thanks in part to the social networks, networked data, where not only nodes' (instances') content and label information, but also links between them are known, have become abundant. The connected instances may have dependencies between their features or labels and they can no longer be assumed independent. Link-based classification (Lu and Getoor, 2003) and collective classification (Macskassy and Provost, 2007, Sen et.al., 2008) methods have been devised for classification of networked data.

Link-based classification takes into consideration the links between the objects in order to improve the classification performance. Attributes of objects and links together can be considered as node features. However, when two linked samples are not yet classified, they require each other's labels to decide on their own label. Collective classification methods have been devised to classify test instances in a network, simultaneously, based on each other as well as the training data.

The aim of collective classification (CC) algorithms is to classify networked data when the test nodes and their links to other test nodes and training nodes are known. In CC, first a base classifier is trained using both content and link information in training data. Then, using a collective inference method, test nodes are iteratively labeled, based on their content and neighbor information. Especially when there is class autocorrelation among the neighboring nodes in the network, test nodes are able to take advantage of their neighbors' class information and collective classification improves the test classification accuracy. Iterative Classification Algorithm (ICA), Gibbs Sampling and Relaxation Labeling are common methods of collective inference.

In this thesis, with the purpose of improving the test classification accuracy, we investigate a number of different directions for collective classification (CC):

- **Investigation of graph properties and their correlations to determine and improve collective classification accuracy:** Graph properties, such as homophily and degree distribution help us understand the networked data at hand. Graph properties can also be used as guidelines in finding out if a sampling method used for evaluation of algorithms is a good one or not, or which type of aggregation of neighbor labels should be used for classification. In this section, we not only define some of the important graph properties existing in the literature, but also introduce

some new graph properties. Among those properties are, the local alpha which is the average accuracy of a node's neighbors in training set, and the local beta which is the average homophily of the node's neighbors in training set.

Some graph properties such as degree distribution, clustering coefficient, betweenness centrality can be calculated using only the link information. These properties can be grouped under the unlabeled graph properties. Homophily, entropy and local beta are graph related labeled graph properties, while accuracy and local alpha are classifier related labeled graph properties.

We compute and compare the graph properties for the Citeseer, Cora, WebKb, HepTh and Synthetic datasets which are used in the thesis. We also present visualization of these properties with respect to each other and their correlations.

- **Training separate classifiers for content and link views and combining these heterogeneous classifiers to improve collective classification accuracy:** Networked data contains both nodes' features and links, therefore classifiers can be trained based on the content-based features, link-based features or both. Based on the characteristics of inputs and graphs on networked data, different classifiers may perform better than the others. Using the same classifier for both node content-based features and link-based features, could lead to loss of accuracy.

  We aim benefit from the diversity of the content and link views and classifiers trained on them. We propose seven different classifier combination methods to combine different classifiers trained on different views. In order to select the most diverse and useful set of classifiers to combine, we use a genetic algorithm based selection method. We set aside a portion of training data as validation set and use the validation sets to determine which classifiers should be combined to achieve better performance for the given train/test partition of the dataset. After we determine the classifier set to be combined, we use the same classifier set for the labeling test set. Our experiments on four different datasets, Citeseer, Cora, WebKb and HepTh, show that our genetic algorithm based classifier combination method outperforms the best base classifiers on the datasets we used. Our method can also be extended to collective classification scenarios with multiple types of content and link.

- **Using link direction information in collective classification:** Most classification methods for networked data assume undirected links. However, when link direction information is available, using this information could improve classifier accuracy. In the thesis, we investigate the effect of using directed link information on classifier performance in collective classification.

  Performance of collective classification algorithms using different local classifiers are evaluated on Citeseer, Cora, WebKb and HepTh datasets with random and snowball sampling, using directed and undirected links for different train/test ratios. It has been shown that by using directed graphs, significant performance increase is obtained when link only classifier is used. The link direction information also improved the collective classification (ICA) results. Since directed links are also used for snowball sampling and since the growing snowball has less directions to grow compared to undirected one, the content only classifier performance slightly decreases. Since random sampling does not depend on links, content only classifier for random sampling is not affected by link direction.

- **Investigation of different feature enrichment and selection methods for collective classification:** While it is usually very difficult to obtain labels on the whole dataset, node content features and links are usually easier to obtain. Semi-supervised learning methods, such as transductive classification (Vapnik, 1998), aim to make use of the labeled and unlabeled data for better classification accuracy. Since collective classification produces the instance labels for a specific set of test instances, collective classification is a transductive classification method. However, since the test instances' labels are not available, collective classification can use the test inputs not for classifier training, but only during inference based on trained classifier outputs. In the thesis, we introduce a new method of transductive network classification which can use the test node features when training classifiers. We train our classifier using enriched node features. The enriched node features include, in addition to the nodes' own features, the aggregated neighbors' features and aggregation of node and neighbor features passed through simple logical operators such as OR and AND. Enriched features may contain irrelevant or redundant features, which could decrease classifier performance. Therefore, we employ feature selection to determine whether a feature among the set of enriched features should be used for classifier training or not. The feature selection method used, FCBF# (Cataltepe, Sonmez and Senliol, 2013, Senliol et.al., 2009), is a mutual information based, filter type, fast, feature selection method.

  The methods introduced in the context of this section of the thesis, is an extension of Baris Senliol's previous work on feature enrichment (Senliol, 2010). In the previous work, enriched features are constructed as combinations of plain (node's own features), neighbors features and ORed features. In addition to those, we also introduce ANDed features and take into account all major combinations of those enrichment methods as separate cases. In (Senliol, 2010) features are enriched and then feature selection is applied (we call this method the EnrSel method). In this thesis, we introduce the SelEnr method, in which, feature selection among the node features is done and then those features are enriched. In addition to the Logistic Regression, we also experiment with the Bayes Net as the base classifier.

  Experimental results on three different network datasets show that classification accuracies obtained using network enriched and selected features are comparable or better than content only or collective classification.

- **Utilization of one against all collective classification for multi-class datasets with heterogeneous class homophilies:** In previous studies, it has been observed that in order to have an accuracy increase when neighbor information is used, homophily, which measures the label-label correlation between neighboring nodes, is required. On the other hand, for multi-class datasets, homophily for each class could be different. In this section, for classification of multi-class networked data, instead of using a single classifier to learn all classes, we use one-against-all scheme and learn a separate classifier for each class. We extend this one-against-all setting to collective classification also. Although one-against-all classification increases the training and testing time due to the increase in the number of classifiers, experimental results show that OAA content only and collective classification is better than single classifier content only and collective classification. The results of the OAA scheme is affected by the base classifier used. The benefit of OAA learning

becomes more emphasized with increase in homophily or decrease in available training data size.

YÖNLÜ BAĞLANTILAR, ÖZNİTELİK ZENGİNLEŞTİRME VE SINIFLANDIRICI BİRLEŞTİRME İLE KOLEKTİF SINIFLANDIRMA BAŞARIMININ İYİLEŞTİRİLMESİ

ÖZET

Örüntü tanıma uygulamalarında, genel olarak, örnekler birbirlerinden bağımsız kabul edilir, örnekler arasındaki bağlantılar/bağımlılıklar/ilişkiler dikkate alınmaz, yalnız örneklerin öznitelikleri ve sınıf bilgileri kullanılarak sınıflandırma yapılır. Ağ bilgisi olan veri kümelerinde sınıflandırma yapıldığında ise, örnekler arasındaki bağlantıları yok saymak, onları birbirlerinden bağımsız varsaymak mümkün değildir. İlişkisel sınıflandırma ve kolektif (beraber) sınıflandırma algoritmaları bu durumun üstesinden gelmek için oluşturulmuş algoritmalardır.

İlişkisel sınıflandırma, örnekler arasındaki bağlantıları kullanır. Sadece komşuların toplanmış sınıf etiketleri kullanılarak ilişkisel sınıflandırıcı eğitim kümesi üzerinde eğitilebilir. Bir test düğümünün sadece eğitim kümesinden değil, test kümesinden de komşuları varsa, bu düğümün sınıfının tahmin edilebilmesi için iteratif yöntemlerle, şu andaki etiket tahminleri kullanılarak bir sonraki tahminlerin yapılması ve tahminler değişmeyene kadar bu sürecin devam ettirilmesi gerekir. Kolektif sınıflandırma algoritmaları, bu şekilde test düğümlerinin birbirlerinin etiket tahminlerine dayanarak ve iteratif şekilde tahmin edildiği sınıflandırma algoritmalarıdır ve Iterative Classification Algorithm (ICA) bu algoritmalardan birisidir.

Bütün düğümlerin öznitelikleri ve bağlantıları bilindiğinde, hem düğüm özniteliklerini hem de komşuların toplanmış sınıf etiketlerini kullanan sınıflandırıcılar ile kolektif sınıflandırma yapılabilir. Kolektif sınıflandırmada öznitelik ve bağlantı bilgisi kullanılarak bir baz sınıflandırıcı eğitilir. Daha sonra iteratif olarak etiketleme işlemine geçilir. Ağ bilgisi içeren veri kümelerinde ağın ortalama derecesi ve homofilisi kolektif sınıflandırma için önem taşıyabilmektedir. Ortalama derece her düğümün komşu sayılarının toplamının (2*düğüm sayısı) ile bölünmesi ile hesaplanmaktadır. Bir düğümün homofilisi kendisi ile aynı sınıfta olan komşu düğüm sayısının düğümün komşu sayısına bölünmesi ile ve ağın ortalama homofilisi ise bütün düğümlerin homofililerinin ortalaması olarak hesaplanmaktadır. Kolektif Sınıflandırma Algoritması (ICA), Gibbs Örnekleme Algoritması (Gibbs Sampling) ve Gevşek Etiketleme Algoritması (Relaxation Labeling) en sık kullanılan kolektif çıkarım algoritmalarıdır.

Tez kapsamında, kolektif sınıflandırmada test sınıflandırma başarımını arttırabilmek amacı ile, aşağıda belirtilmiş olan konular üzerinde bir dizi araştırma gerçekleştirilmiştir.

- **Çizge özelliklerinin ve çizge özellikleri arasındaki ilişkilerin kolektif sınıflandırma başarımında etkili olan faktörleri belirleyebilmek ve sınıflandırma başarımını arttırmak amacı ile irdelenmesi:** Homofili, derece dağılımı gibi çizge özellikleri veriyi anlamaya yardımcı olmaktadır. Çizge özelliklerine bakılarak bir veri kümesi üzerinde hangi örnekleme metodunun kullanılması gerektiği, hangi komşu toplama (aggregation) yöntemlerinin kullanılmasının daha iyi olacağı hakkında fikir sahibi olunabilir. Tezin bu bölümünde, sadece literatürde mevcut çizge özellikleri ile sınırlı kalınmamış olup, yerel alfa (local alpha) ve yerel beta (local beta) gibi yeni bir takım çizge özellikleri oluşturulmuştur. Yerel alfa, bir sınıflandırıcının bir düğümün eğitim kümesindeki komşuları üzerinde ortalama başarımı, yerel beta ise yine düğümün eğitim kümesindeki komşularının ortalama homofilisi olarak tanımlanmıştır.

  Derece dağılımı (degree distribution), kümeleme katsayısı (clustering coefficient), ara merkezlilik (betweenness centrality) gibi bazı çizge özellikleri veri kümesinde bağlantılar mevcut ise hesaplanabilir. Bu sebeple bu tip çizge özelliklerini sınıf etiketsiz çizge özellikleri grubu altında toplamak mümkündür. Homofili, entropi ve yerel beta ise ancak sınıf etiketi bilindiğinde hesaplanabilir ve sınıf etiketli çizge özellikleri grubu altında toplanabilir. Başarım ve yerel alfa ise sınıflandırıcı bağımlıdır ve sınıflandırıcılarla ilgili çizge özellikleri grubu altında toplanabilir.

  CiteSeer, Cora, WebKb, HepTh ve Sentetik veri kümelerinin bir takım çizge özellikleri hesaplanarak, veri kümeleri arasında karşılaştırma yapılmıştır. Aynı zamanda bu çizge özelliklerinin birbirleri ile olan ilişkileri görselleştirilmiş ve aralarındaki korelasyonlar hesaplanmıştır.

- **Sınıflandırma başarımını arttırmak amacı ile içerik ve bağlantılar için ayrı sınıflandırıcıların eğitilmesi ve bu heterojen sınıflandırıcıların birleştirilmesi:** Ağ bilgisi olan veri kümelerinde içerik ve bağlantıların karakteristiklerine bağlı olarak farklı sınıflandırıcılar, farklı şekilde davranış gösterebilirler ve birinin iyi başarım gösteremediği şartlar altında diğer biri iyi başarım gösterebilir. Bu düşünceden hareketle, mümkün olduğunca birbirinden farklı ve doğru sonuç veren sınıflandırıcıların birleştirilmesi amaçlanmıştır. Birleştirmek için yedi farklı birleştirme yönteminin kullanılması önerilmiştir. Burada önemli olan hangi sınıflandırıcılar hangi şartlar altında birleştirildiğinde en yüksek başarım elde edilebilir sorusuna cevap verebilmektir. Bunun için geçerleme kümesi üzerinde hangi sınıflandırıcılar birleştirildiğinde en iyi başarım elde edildiğine bakılıp, test kümesinde de bu sınıflandırıcılar kullanılarak en iyi başarım elde edilebileceği varsayılmıştır. Ancak bu durumda en uygun çözümü bulmak için seçilebilecek sınıflandırıcıların yer aldığı kümenin bütün alt kümelerinin test edilmesi gerekmektedir. Alt küme sayısının küme elemanlarının sayısı ile birlikte üstel olarak değişmesi dolayısı ile özellikle sınıflandırıcı sayısı fazla olduğunda mevcut hesaplama kaynakları ile bu hesaplamanın yapılması mümkün gözükmemektedir. Bu sebeple genetik algoritma kullanan bir sınıflandırıcı seçme algoritması oluşturulmuştur. Genetik algoritmaya uygunluk fonksiyonu olarak en büyütülmek istenilen sınıflandırıcı birleştirme metodu verilmiştir. Seçme işleminin tamamlanmasının ardından test kümesi üzerinde aynı sınıflandırıcı kümesi ile başarım ölçümü gerçekleştirilmektedir. Citeseer, Cora, WebKb ve HepTh veri kümeleri üzerinde yapılan deneyler, önerdiğimiz yeni yöntemin en

iyi baz sınıflandırıcıdan çok daha iyi sonuçlar verdiğini göstermiştir. Yeni önerilen yöntemin çoklu içerik ve/veya çoklu bağlantı içeren veri kümelerine de uygulanabilmesi mümkündür.

- **Kolektif sınıflandırmada yön bilgisinin kullanılması:** Kolektif sınıflandırmada genellikle kolaylık olması açısından çizgeler yönsüz olarak ele alınmaktadır. Bununla birlikte bazı veri kümelerinin ham veri dosyalarında yön bilgisi mevcuttur. Mevcut olduğunda, bu bilgiyi kullanmamak, sınıflandırma için faydalı olabilecek bir bilgiyi kaybetmek anlamına gelebilir. Bu sebeple, çizgelerdeki yön bilgisinin kullanılmasının kolektif sınıflandırmada sınıflandırıcı başarımı üzerine etkisi araştırılmıştır.

  Farklı baz sınıflandırıcılar kullanan kolektif sınıflandırma algoritmalarının başarımı Citeseer, Cora, WebKb ve HepTh veri kümeleri üzerinde gerek yön bilgisi dikkate alınarak ve gerekse alınmadan, farklı eğitim/test kümeleri için hem rastgele örnekleme hem de kartopu örnekleme ile ölçülmüştür. Çizgelerde yön bilgisi dikkate alındığında, özellikle ilişkisel sınıflandırıcının başarımı ciddi oranda artmıştır. Bu yararlı bilginin kullanılması, kolektif sınıflandırmaya da katkıda bulunmaktadır ve ilişkisel sınıflandırıcıda olduğu kadar olmasa da bir miktar başarım artışı gerçekleşmektedir. Çizgedeki yön bilgisi, kartopu örnekleme ile kullanıldığında, içerik sınıflandırıcının başarımında bir miktar azalma olduğu gözlemlenmiştir. Bunun temel sebebi ise şudur: Kartopu örnekleme bilindiği gibi bağlantılara bağlıdır ve yön bilgisi dikkate alındığında adım adım büyüyen kartopu yön bilgisinin dikkate alınmadığı duruma göre daha az yöne doğru genişleme olanağına sahiptir. Bununla birlikte, içerik sınıflandırıcı rastgele örnekleme ile birlikte kullanıldığında rastgele örneklemenin bağlantılardan bağımsız olması dolayısıyla, içerik sınıflandırıcının başarımında herhangi bir değişiklik olmamaktadır.

- **Kolektif sınıflandırma için farklı nitelik zenginleştirme ve seçme yöntemlerinin araştırılması:** Genellikle bir veri kümesindeki bütün örnekleri sınıf etiketli olarak bulmak mümkün değildir. Buna karşılık örneklerin öznitelikleri ve örnekler arasındaki bağlantılar genellikle mevcuttur. Transdüktif (trunsductive) sınıflandırma (Vapnik, 1998) gibi yarı-gözetimli öğrenme yöntemleri sınıflandırma başarımını arttırmak için hem sınıf etiketli hem de sınıf etiketsiz örnekleri kullanmayı amaçlar. Kolektif sınıflandırma, test örneklerinin belirli bir alt kümesi için sınıf etiketlemesini gerçekleştirmesi dolayısıyla transdüktif bir sınıflandırma yöntemidir. Bununla birlikte, test kümesindeki örneklerin sınıf etiketlerinin bilinmemesi dolayısıyla, kolektif sınıflandırma sadece eğitilmiş sınıflandırıcı çıktılarına dayanan çıkarım esnasında test örneklerini kullanır, sınıflandırıcı eğitilmesi sırasında test örneklerini kullanmaz. Bu tez kapsamında, sınıflandırıcıların eğitilmesi sırasında test düğümlerinin özniteliklerini kullanabilen yeni bir transdüktif ağ verisi sınıflandırma yöntemi oluşturulmuştur. Sınıflandırıcı eğitimi sırasında düğümlerin zenginleştirilmiş nitelikleri kullanılmıştır. Zenginleştirilmiş nitelikler denildiğinde kastedilen, düğümün özniteliklerine ek olarak komşularının toplanmış nitelikleri ve düğümün kendi nitelikleri ile komşularının niteliklerinin VE ya da VEYA'larının birleştirilmiş halidir. Ancak bu şekilde oluşturulmuş yeni nitelikler arasında alakasız veya tekrarlı niteliklerin oluşma ihtimali yüksektir ki bu durum sınıflandırıcının başarımının düşmesine sebebiyet verebilir. Bu

durumun üstesinden gelebilmek için bir nitelik seçme algoritması kullanılarak hangi niteliklerin sınıflandırma için daha yararlı olacağı belirlenmektedir. Bu çalışmada karşılıklı bilgi tabanlı, hızlı bir öznitelik seçme algoritması olan FCBF# (Cataltepe, Sonmez and Senliol, 2013, Senliol et.al., 2009) kullanılmıştır.

Tezin bu bölümünde tanıtılmış yöntemler, daha önce Barış Şenliol tarafından nitelik zenginleştirme konusunda yapılmış bir çalışmanın geliştirilmiş ve genişletilmiş halidir (Senliol, 2010). Önceki çalışmada, zenginleştirilmiş nitelikler düğümün öznitelikleri, komşularının nitelikleri ve VEYA'lanmış nitelikler kullanılarak oluşturulmaktadır. Bu çalışmada ise bunlara ek olarak VE'lenmiş nitelikler de eklenmiştir. Ayrıca bu yöntemlerin farklı şekildeki kombinasyonlarla birleştirilmesi ile yeni zenginleştirme yöntemleri oluşturulmuştur. Önceki çalışmada (Senliol, 2010), sadece EnrSel yöntemi kullanılmıştır ki bu yöntemde öncelikle zenginleştirme yapılıp, sonrasında zenginleştirilerek elde edilmiş öznitelikler üzerinde öznitelik seçimi gerçekleştirilmektedir. Bununla birlikte, bu çalışmada SelEnr yöntemi geliştirilmiş ve mevcut EnrSel yöntemi ile karşılaştırması da yapılmıştır. Bu yöntemde, önce öznitelik seçimi gerçekleştirilmekte, ardından seçilmiş öznitelikler zenginleştirilmektedir. SelEnr yönteminin EnrSel yönteminden çok daha iyi sonuç verdiği görülmüştür. Ayrıca önceki çalışmada baz sınıflandırıcı olarak Lojistik Regresyon'a ilaveten, bu çalışmada Bayes Ağları kullanılmıştır.

Üç farklı ağ veri kümesi, Citeseer, Cora ve WebKb, üzerinde gerçekleştirilen deney sonuçları, zenginleştirilmiş niteliklerle gerçekleştirilen sınıflandırmada elde edilen başarımın, orijinal özniteliklerle gerçekleştirilen içerik sınıflandırma ve kolektif sınıflandırma ile elde edilen başarımdan çok daha yüksek olduğunu göstermektedir.

- **Sınıf homofilileri heterojen, çok sınıflı veri kümeleri üzerinde bire-karşı-hepsi kolektif sınıflandırma kullanılması:** Komşuluk bilgisinin kullanıldığı sınıflandırmalarda, düğüm etiketi ile komşu düğümlerin etiketleri arasındaki korelasyonu gösteren homofilinin yüksek olmasının, başarımı arttırmak için gerekli olduğu daha önceki çalışmalarda ortaya konmuştur. Bununla birlikte çok sınıflı veri kümelerinde her bir sınıf için homofili değeri farklı olabilir.

  Tezin bu bölümünde, çok sınıflı, ağ bilgisi bulunduran veri kümelerinde sınıflandırma için bütün sınıfları öğrenen tek bir sınıflandırıcı yerine, her bir sınıfı ayrı ayrı öğrenen sınıflandırıcılar eğitilmiştir. Ayrıca bu yapı kolektif sınıflandırmaya da uyarlanmıştır. Her ne kadar bire-karşı-hepsi sınıflandırmada, her bir sınıf için ayrı sınıflandırıcı eğitilmesi dolayısıyla eğitim ve test için harcanan süre artıyor olsa da, deneyler, bire-karşı-hepsi sınıflandırma ile elde edilen sonuçların hem içerik sınıflandırıcısında hem de kolektif sınıflandırmada elde edilen sonuçlardan daha iyi olabildiğini göstermiştir. Ayrıca bire-karşı-hepsi sınıflandırmanın sağladığı faydanın kullanılan baz sınıflandırıcı ile oldukça ilişkili olduğu tespit edilmiştir. Bire-karşı-hepsi sınıflandırmanın yararları özellikle homofili değerlerinin artışı veya eğitim kümesi boyutunun azalması ile daha fazla ortaya çıkmaktadır.

1. INTRODUCTION

Sharing is the magic word, especially for the last few years. Most of us are trying to introduce more about ourselves by sharing our special times, photos, meetings, greetings etc., in short, our special life frames via social networks with our friends. We also have linked work profiles as well. We are connected to each other that we know, working together in the same or similar fields. We share the work done together, our experiences or ideas with each other. So, we become members of these networks like any paper published is a part of literature network or any published web page is a part of Internet network. These connections between the members, cause huge, interconnected networks to occur. Examples include social (Xiang, Neville, & Rogati 2010), semantic (Tresp, Bundschus, Rettinger, & Huang 2008), financial (A. Bernstein, Clearwater, Hill, Perlich, & Provost 2002), communication (Dasgupta et al. 2008) and gene regulatory (Awan et al. 2007) networks.

Most pattern recognition applications assume that instances are independent and identically distributed and do not handle the dependency between the instances. Classification problems are solved using instances' features (content) and labels. Connections/dependencies/relations between instances are not taken into consideration. On the other hand, when instances are networked, they are now connected and can not be assumed independent of each other. Link-based classification and collective classification methods are devised to solve classification problems for networked data.

Link-based classification (Sen & Getoor 2007) takes into consideration the links between the objects in order to improve the classification performance. Attributes of objects and links together can be considered as node features. However, when two linked samples are not yet classified, they require each others' labels to decide on their own label. Collective classification methods have been devised to classify test instances in a network simultaneously, based on each other as well as training data.

The aim of collective classification (Chakrabarti, Dom, & Indyk 1998, Macskassy & Provost 2007, Sen et al. 2008) algorithms is to classify networked data when the test nodes and their links to other test nodes and training nodes are known. In collective classification, first a base classifier is trained using both content and link information in training data. Then, using a collective inference method, test nodes are iteratively labeled, based on their content and neighbor information. Especially when there is class autocorrelation, i.e. linked nodes are likely to have similar labels, among the neighboring nodes in the network, test nodes are able take advantage of their neighbors' class information and collective classification improves classification accuracy (Jensen, Neville, & Gallagher 2004). Iterative Classification Algorithm (ICA), Gibbs Sampling and Relaxation Labeling (Macskassy & Provost 2007, Sen et al. 2008) are common methods of collective inference. Collective inference methods have been studied in detail in the works of (Macskassy & Provost 2007) and (Sen et al. 2008).

Different choices of base classifiers that are able to use content and neighbors' link information, such as naive Bayes, logistic regression, decision trees, k-nearest neighbors, have been used in the literature (Jensen et al. 2004, McDowell, Gupta, & Aha 2009; 2007, Neville & Jensen 2000, Sen et al. 2008). The base classifier takes as input, usually, the content features of the node being classified and relational features, which are usually an aggregation of the class labels of the other linked instances (McDowell et al. 2009; 2007, Sen et al. 2008). ICA is generally known to get better performance than content only classification. However, the improvement depends on the structure and homophily of the network.

In this thesis, with the purpose of improving test classification accuracy, we investigate a number of different directions for collective classification (CC). We look closer to the datasets and propose changes to the basic collective classification setting. The contributions of the thesis are:

- Investigation of **graph properties** and their correlations to determine and improve CC accuracy

- Training separate classifiers for content and link views and combining these **heterogeneous classifiers** to improve CC accuracy
- Using **link direction** information in collective classification
- Investigation of different **feature enrichment and selection methods** for collective classification
- Utilization of **one against all collective classification** for multi-class datasets with heterogeneous class homophilies

The thesis is organized as follows: In the remaining part of this section, we provide a literature review related to the contributions of the thesis, introduce the notation, the datasets used and also the Iterative Classification Algorithm (ICA), which is used throughout the thesis as a method of collective classification. In Section 2, we compare certain graph properties of Citeseer, Cora, WebKb, HepTh and Synthetic datasets. Visualization of these properties with respect to each other and correlations between graph properties are presented. In Section 3, we introduce a new genetic algorithm based heterogeneous classifier combination method for networked data. In Section 4, we explore the effect of using direction information on test accuracy performance in collective classification. In Section 5, we introduce a new method of transductive network classification which can use the test node features when training the classifier (Cataltepe, Sonmez, & Senliol 2014). In Section 6, for classification of multi-class networked data, instead of using a single classifier to learn all classes, we use a one-against-all scheme and learn a separate classifier for each class. We extend this one-against-all setting to collective classification also. In Section 7, we present conclusions based on the results of the thesis.

1.1 Literature Review

In this section, we review the literature related to the contributions of the thesis.

In the thesis, we use content graphs that we generated based on the cosine similarities between content features of the nodes. There exist different approaches for generating content graphs. These approaches are reviewed in Section 1.1.1. After we present

graph properties for both content and link graphs, we introduce a new genetic algorithm based heterogeneous classifier combination method for networked data. Related methods for classifier combination on collective classification are reviewed in Section 1.1.2 and the methods using genetic algorithms for classifier combination are reviewed in Section 1.1.3.

As another contribution of the thesis, we introduce a new method of transductive network classification which can use the test node features when training the classifier. Other methods of using neighbors' features, feature selection and construction are reviewed in Section 1.1.5. Since transductive learning is a semi-supervised learning algorithm, we also review related semi-supervised approaches in Section 1.1.6.

In the thesis, we introduce a one-against-all scheme for collective classification of multi-class networked data. So, we review the work related to multi-class classification in Section 1.1.7.

1.1.1 Graph generation from content features

Similar to our content feature based graphs G_{CO}, graph construction methods have been used to improve classification in a number of studies.

Linear Neighborhood Propagation (LNP), a semi-supervised algorithm which constructs a graph from content features was introduced in (F. Wang & Zhang 2008). The algorithm first calculates the similarity between content features of the nodes. Then each node finds the K-nearest neighbors according to the similarity scores. The weights of the edges between these nodes are set according to similarity scores. The weight matrix of the whole graph is constructed by aggregating the edge weights for each partition.

The effect of using combination of links on test accuracy performance was explored in (Macskassy 2007) on six benchmark datasets. A graph from content features was created by calculating the cosine similarity between TF-IDF vectors of the nodes. If the similarity was above a threshold, the nodes were connected. The assortativity score, which is the correlation between classes, linked in the corresponding graph was calculated. After normalization of the edge weights, these weights were rescaled by

multiplying with the corresponding assortativity scores. wvRN (Macskassy & Provost 2003) was used for relational classification, while Naive Bayes was used for text classification. By combining links, better test accuracy performance was obtained than just using content or links individually.

Another graph construction method, which assumes that features of the node can be constructed from its k nearest neighbors was introduced in (F. Wang & Zhang 2008). Each neighbor had a contribution weight for the construction. The weights were directly correlated with the similarity between the nodes. After the weights were determined by solving an optimization problem with the constraint that the sum of the weights should be 1, all the nodes were labeled with a propagation through the network until convergence. Test accuracy performance of the proposed method was compared with two datasets on digit recognition and text classification. The method was able to reach stability more faster than the other methods and generally performed better or as good as the other methods.

A weighted graph from the content features of the nodes was also constructed in (Zhu, Ghahramani, Lafferty, et al. 2003). The weights were determined based on the similarity between nodes. To classify the unlabeled nodes, Gaussian fields with harmonic energy minimization was applied to the graph. The resulting algorithm is like a form of nearest neighbor algorithm where nearest nodes are determined by random walks in the graph. The method was evaluated with digit and text classification on real datasets and a synthetic dataset. For text classification, two nodes were connected if the node was in neighborhood of the other node's 10 nearest neighbors and the other node was in 10 neighborhood of the node. The neighborhood was measured with cosine similarity. The results were compared with two baselines, namely 1-NN and RBF. Better performance was obtained especially when the percentage of labeled data points is small.

1.1.2 Classifier combination for collective classification

There have been studies, mostly using relational classifiers, that combine a number of classifiers for collective classification.

Local attributes with relations were combined using ensemble classification in (Preisach & Schmidt-Thieme 2008). The aim was to produce a generic relational ensemble model that could incorporate both relational and local attributes for learning. The issues related to heterogeneity, sparsity and multiple relations were also addressed. A new method, PRNMultiHop, was introduced to handle the sparsity problem. In this method, two nodes are considered connected if they can reach one another in at most a certain threshold number of hops. Ensemble classification was performed as follows: First, a base classifier using the local features and a relational classifier for each type of relational feature (link type) were trained, then the results of these classifiers were combined using stacking or voting. Stacking gave better results than voting, however stacking suffered from high memory requirements. The performance of the method was compared with the performance of RBC, RPT and RDN on Cora and CompuScience datasets. The method PRN2MultiHop, outperformed those three methods.

In stacked graphical models (Fast & Jensen 2008, Kou & Cohen 2007), a relational model based on stacking was constructed. Due to the use of predicted labels instead of actual labels during training, a smaller variance or bias was obtained. The proposed stacked model was shown to perform as good as collective classification.

Thirty classifiers based on neural networks and fuzzy interface systems were evaluated on UCI and Real datasets in (Bulacio, Guillaume, Tapia, & Magdalena 2010). The classifiers were combined by applying heuristic search that takes into account the fuzzy integral behavior. It was possible to reach the best classifier's accuracy after combination.

A bagging procedure for networked data was introduced in (Murrugarra-Llerena & de Andrade Lopes 2011). The introduced method was compared with the bagging versions of well known relational classifiers such as wvRN, PRN, CDRN on sixteen data sets. The ensemble model generally performed better than the originating individual classifiers. The impact of diversity of the individual classifiers on the ensemble model accuracy was also evaluated. Measure of agreement was used as the diversity measure. The diversity was found to be directly correlated with the accuracy obtained using the ensemble.

Separate structured logistic regression models were trained for content and link in (Lu & Getoor 2003), instead of using both in a flat logistic regression model. Using separate models on Citeseer, Cora and WebKb datasets, better results were obtained. The effect of ordering for ICA was also evaluated. It was found that ordering does not affect the performance of ICA significantly. In addition, the effect of using different link-based models such as binary, mode and count methods on the performance of ICA was investigated. Count method performed statistically significant to the others. Like in this study, we use count aggregation method to construct link features and also train separate models for content and link. However, we train thirty different classifiers, ten for content only, ten for link only and ten for ICA, which is in contrast to the model that only two logistic regression classifiers were trained. The previously proposed model uses just Max Classifier Combination method (See Section 3.2.1), where as we use seven different classifier combination methods. Since that model consists of just two logistic regression models, there is no need for model selection. However in our case, we select the classifier set which maximizes the test accuracy on the validation set. The previous study also achieved to get better performance using separate models, however in our case the increase in performance is more significant.

In a recent study, a method to produce an ensemble of classifiers and combining them during collective inference through the Collective Ensemble Classification (CEC) was introduced (Eldardiry & Neville 2012). It was shown that reduction of variance was obtained when the classifier ensemble was produced using the Relational Subgraph Resampling (RSR) method. The RSR method aims to reduce the variance of the relational classifiers trained on the ensemble datasets. It produces a snowball sample with a number of initial starting points instead of just one.

There have also been studies that compared and combined classifiers trained using different types of (multi) link information. A relational classifier (Relational Neighbor (RN)) classifier was used as a baseline classifier in (Macskassy & Provost 2003). Common author and citation graphs were used to construct a new graph on Cora dataset. The sum of the authors within two papers in common and the number of citations between those two papers were used as the link weights of the new graph. The test accuracy performance of the method was compared with PRMs (Getoor,

Friedman, Koller, & Taskar 2003) and RNs for different ratio of known labels. On IMDB dataset, a small subset of movies released in United States between 1996 and 2001 was used with the aim of classifying if a movie would be blockbuster or not. Four different relations between movies, namely actor, director, producer and production company, were used. First, each link type was evaluated separately with RN. After determining the link type that RN performed best, incrementally other link types were added and the test accuracy performance of RN was assessed. The best test accuracy was obtained with RN using three link types together, namely director, producer and production company. On WebKb dataset a co-citation graph was produced and RN achieved close to the best accuracies using just 5% of the whole data. Similarly, (Popescul & Ungar 2003) showed that accuracies obtained using multiple types of link information, namely citation, authorship and publish date on Citeseer dataset, together with Structural Logistic Regression are better than using only citation links.

1.1.3 Genetic algorithms for classifier combination

Genetic algorithms (GA) were first used for classifier combination in (Kuncheva & Jain 2000). Two different schemes were used. In the first one, GA selected different non-overlapping subsets of features. Then, each individual classifier was trained with the selected subset of features. The training accuracy of the combination was used as the fitness function. For combination, majority voting method was used. In the second scheme, in addition to feature subsets, the individual classifiers to be combined were also selected. The feature subsets to be used for different classifiers were allowed to overlap. Extra bits corresponding to each individual classifier were added to the chromosome used in the first schema. Training accuracy performance of the classifier combination method was again used as the fitness function. The methods were evaluated on four real datasets, namely SatImage, Letters, Hear and Forensic glasses, and three classifiers, namely linear, quadratic discriminant and logistic classifiers. The results were compared when all features were used, a backward feature selection algorithm was used or GA was used just for feature selection with the best performing individual classifier. It was found out that the second schema performed better than the first one. On two of the datasets, namely SatImage and Forensic glasses, the second

schema had lower test error rates than individual classifiers. For the other two datasets, the error rates were higher, due to overfitting of the GA. The reason why the first schema did not perform well was that the individual classifiers had to use too few features because overlaps were not allowed and the best accuracy was obtained when all the features were used.

As in the work of (Kuncheva & Jain 2000), we also encode classifiers to be combined in chromosomes. Unlike that study, we do not use feature subsets but all the features when training individual classifiers. The fitness function we use is not training accuracy but validation accuracy. This allows us to overcome the overfitting problem of the GA. We use our GA based classifier combination method on networked data, while in (Kuncheva & Jain 2000) classifiers were non-relational (content only). While the number of individual classifiers to be combined in (Kuncheva & Jain 2000) was three, it is thirty in our case. The test accuracies obtained in this previous study were not always better for the datasets used. However, in our case, the test accuracy increase in performance is significant and consistent for all different train/test ratios for all the datasets.

1.1.4 Local classifier evaluation

Homophily (McPherson, Smith-Lovin, & Cook 2001), i.e. that linked nodes are more likely to have the same label, has been one of the important requirements for link information to be useful for classification. Algorithms have been devised to take into consideration the neighbors' labels. Weighted-vote Relational Classifier (wvRN) (Macskassy & Provost 2003), is a relational learning algorithm that aggregates the neighbors' labels and uses them as inputs to a classifier. Aggregation methods (Lu & Getoor 2003, Perlich & Provost 2006, Sen & Getoor 2007) which summarize the label information of the neighbors in a constant dimensional vector through taking the sum, average, max or existence of neighbor labels, have been used. By means of training classifiers with node content features, appended with aggregated neighbor labels, (Macskassy & Provost 2007, Sen et al. 2008) have been able to use both content and link information to train classifiers.

In the thesis, we use neighbor homophily (β) and accuracy (α) as a means to evaluate classifier performance on a certain node. Local methods of classifier evaluation have been used to improve collective classification in a number of previous studies.

(Angin & Neville 2008) stated that assuming label autocorrelation to be stationary for the whole graph may not be correct, different regions of the graph may show different characteristics. For computation of the global and node neighborhood autocorrelation, Pearson's corrected contingency coefficient was used. The class membership probabilities were also computed for a node given the neighborhood of a node globally on the whole graph and locally around the specific node. A linear combination of these two probabilities was used as the label probabilities. The number of labeled neighbors of the node were used as the weights of the local model.

McDowell and colleagues work is also related to our work. (ICA_{MC})(McDowell, Gupta, & Aha 2010) selects the set of predicted labels which are more reliable. The reliable node labels are selected by a meta classifier constructed with a meta training set from the original training set. They also use feature selection on meta-features (McDowell et al. 2010) that are related to a node and labels and predictions on the nodes' neighbors. They show that ICA_{MC} achieves better accuracies than ICA.

1.1.5 Feature construction and feature selection for collective classification

There have been previous methods of feature construction which aim to take advantage of network information to train better classifiers. The simplest method of feature construction was performed by weighted-vote relational classifier (wvRN) (Macskassy & Provost 2003). As mentioned before, wvRN determines the class of a node based on a weighted average of its neighbors' class probabilities.

Statistical models for classification of networked data were proposed in (Chakrabarti et al. 1998). The content features of the neighboring nodes were used by assuming the words in the neighboring documents as if they are local features. Naive Bayes classifier and the relaxation labeling methods were used. For the datasets used, while using the labels of the neighbors in addition to nodes' contents improved performance, using the neighbors' contents did not. The reason why using nodes' contents did not improve performance, may be related to heuristic feature selection algorithm used and

the cross-linkage in the datasets used. Also, the term distribution of the dataset was not found to be highly correlated with the labels. Even after feature selection, average degree of the nodes were much higher than the average degree of the nodes in the datasets we used. Tagging also increased the number of features but decreased the number of samples and also the correlation of these features with the labels. In our study, we do not tag the features from neighbors, instead we use them as if they are local features. We use simple logical operators such as AND, OR to reveal features that are common between neighbors or that may exist if exists in the neighbors. Thus, the schema we used was able to find more correlation with the enriched features and labels.

A web page categorization algorithm was proposed in (Oh, Myaeng, & Lee 2000). The algorithm tries to discriminate the unrelated nodes by calculating similarity between the content of the neighboring nodes and assign a trust level accordingly. The algorithm also takes care of the content of the neighboring documents to calculate the term weights of the terms of the document. When the content of the the neighboring documents were used, the performance decreased compared to using the original content features.

A hypertext classifier was constructed in (Slattery & Craven 1998), which used statistical text learning methods. In the method, the content features of the neighboring nodes were used in such a way that, predicates were produced and one of the relations used was to check if a word occurred at least five times in the neighboring nodes. The experiments preformed on WebKb dataset, outperformed classification with Naive Bayes using bag of words representation.

There has also been some previous work on feature construction and then feature selection for networked data. Previously, (Popescul & Ungar 2004) suggested approaches for feature construction from database tables using refinement graphs. Then they selected features using a statistical model selection criteria. Perlich and Provost's relational learning system ACora (Automated Construction of Relational Attributes) (Perlich & Provost 2006) investigated many methods of feature construction, such as count, mode, max, using a node and its related entities. They outlined principles of feature aggregation, namely, aggregation should help

with classification and various aggregation methods should be considered. So, they considered distances to the class-conditional distributions and used standard aggregates for feature construction. Although (Perlich & Provost 2006) suggested that feature selection should be performed on the constructed features, they did not report results with feature selection, because it did not improve results for the datasets they used.

In (Senliol, Aral, & Cataltepe 2009), mRMR (Peng, Long, & Ding 2005) feature selection was used for classification of networked data using the node features. They showed that content only or collective classification using feature selection can achieve accuracies as high as using all the features. In (Rossi, McDowell, Aha, & Neville 2012), Rossi et.al. described a number of node feature construction methods and then possible dimensionality reduction methods on them.

1.1.6 Semi-supervised learning

In semi-supervised learning (Zhu 2008), the unlabeled instances are used to maximize the margin (Joachims 2003), complexity to place classifier boundaries around the low density regions between clusters in the data (Chapelle & Zien 2005). There are also co-training (Blum & Mitchell 1998) type algorithms which need different classifiers that are obtained through the use of different type of classifiers, different feature subspaces (Yaslan & Cataltepe 2010) or set of instances. When the classifiers produced are diverse and accurate enough, co-training may improve the final test accuracy (W. Wang & Zhou 2007). On the other hand, (Cozman, Cohen, & Cirelo 2002) has shown that unlabeled data can degrade the classification performance when there are discrepancies between modeling assumptions used to build the classifier and the actual model that generates the data. Therefore, both for the general semi-supervised and the transductive learning, the use of unlabeled data is not guaranteed to improve performance.

Transductive learning (Vapnik 1998) for networked data has been addressed in a number of studies, including (Macskassy & Provost 2007, Sen et al. 2008). A number of studies have imposed smoothness or regularity constraints (Culp & Michailidis 2008, Zhou, Scholkopf, & Hofmann 2005) on the classifier, which force the predicted

labels to be similar to each other on neighboring nodes. In addition to these, (Ji, Sun, Danilevsky, Han, & Gao 2010) has developed a framework called GNetMine that can be used for transductive classification on a heterogeneous network.

Recently, a novel semi-supervised algorithm for Collective Classification, ALFNET, was introduced in (Bilgic, Mihalkova, & Getoor 2010). The algorithm exploits links to select more informative examples. It first clusters the data using links. Within each cluster, the most informative nodes are selected from the unobserved nodes according to a score of disagreement between content only and collective classifiers. Also, a novel semi-supervised collective classification method was introduced. In this method, unobserved nodes are first labeled with CO classifier, then aggregation function is computed over the neighbors of the node. In addition, dimensionality reduction is applied as another case using PCA. The experiments on Cora and Citeseer datasets showed using dimensionality reduction and semi-supervised collective classification together was more accurate than using semi-supervised setting. The accuracy obtained using both of these methods were much higher than using CO and CC.

Another novel active learning method RAL (Relational Active Learning) which combines semi-supervised learning and relational resampling was introduced in (Kuwadekar & Neville 2011). The method uses across-network classification, to be able to separate the effects of label propagation and prediction. The superiority of the method proposed to ALFNET (Bilgic et al. 2010) is the applicability of the model to networks having a few content features. With the use of certainty instead of uncertainty, the accuracy of the model is increased by selecting the nodes having more consistent neighbors which provided more accurate labels to propagate over the network. On both synthetic and real-world datasets, namely IMDB and AddHealth datasets, the RAL method was able to learn faster than the other compared algorithms.

ICA (Iterative Classification Algorithm) (Macskassy & Provost 2007, Sen et al. 2008) can be related to other network diffusion algorithms, such as affinity propagation (Frey & Dueck 2007), which is used for clustering and where nodes propagate a degree of how they see the other node as their exemplars. In ICA algorithm, for each test node the classifier output is computed and then propagated to its neighbors. Previously in (Cataltepe, Sonmez, Baglioglu, & Erzan 2011), we have shown that instead of

combining content and link features into a single feature vector and training a single classifier, training separate classifiers for content and link features and then combining them can result in accuracies as well as collective classification.

1.1.7 Multi-class classification

There are different approaches for multi class classification in the literature. These are OAA (one-against-all scheme), OAO (one against one scheme) and DAG (directed acyclic graph scheme).

In OAA scheme (Bottou et al. 1994), which is sometimes also called as one-vs-all, OVA, for each class a classifier is trained which tries to discriminate the class from all other remaining classes. If there are K classes in the multi class dataset, then K classifiers are trained. When a test instance is to be classified, all classifiers output their class membership probabilities for that instance. Predicted class for the instance is assigned by taking into account the highest membership probability value.

In OAO scheme (Knerr, Personnaz, & Dreyfus 1990), for each class pair a different classifier is trained which tries to discriminate considered classes from each other. Obviously, if there are K classes in the multi class dataset, $K(K-1)/2$ classifiers are trained. Predicted class for a test instance is assigned by means of majority voting.

In DAG scheme (Solla, Leen, & Müller 2000), the training phase is the same as in the OAO scheme. However in testing phase, an acyclic graph is constructed to classify the test sample. For each node, classifier output probability for class i and class j are compared and the class that gets less probability is eliminated and is not used for later comparisons. This process is repeated for $K-1$ times until $K-1$ classes are eliminated. This method is faster than OAO scheme.

One-against-all (OAA) classification has been compared to one-against-one (OAO) classification for different types of classifiers, such as SVMs (Hsu & Lin 2002), Decision Trees (Polat & Güneş 2009) and Neural Networks (Ou & Murphey 2007), and different types of applications, such as for handwritten recognition with SVM and MLP (Milgram, Cheriet, Sabourin, et al. 2006), for land cover mapping with SVMs

(Anthony, Gregg, & Tshilidzi 2007), for fusion of multi temporal synthetic aperture radar data and optical imagery with SVM (Waske & Benediktsson 2007).

Different schemes in the literature, such as single machine scheme, error-correcting code scheme and OAA are reviewed in (Rifkin & Klautau 2004). It is shown that there is no evidence that they are superior to OAA. Also shown that when relatively weak classifiers are used, which are not tuned well, combining them could produce better results.

Multi-class schemes with SVM were applied in (Nguyen & Rajapakse 2003). These include one-against-all, one against one, directed acyclic graph and two approaches for protein protein secondary structure (PSS) prediction problem on two datasets. Two-stage SVMs performed better than single-stage SVM techniques for the PSS problem.

1.2 Notation and Background

We assume that there is a networked dataset represented by a graph $G = (V, E)$ with nodes (vertices) V and undirected links (edges) $E \subseteq \{\{u, v\} | u, v \in V\}$.

Each node $u \in V$ can belong to only one of C classes and the label is denoted by $\mathbf{r}(u) \in \{0,1\}^C$ which contains 1 at location i and 0 everywhere else if node u belongs to class i. Some of the vertices are in the training set V_{train} whose labels are known, while the rest are in the test set V_{test} whose labels will be predicted. Note that, $V_{train} \cap V_{test} = \emptyset$ and $V_{train} \cup V_{test} = V$.

If the validation set V_{val} is used then it is separated from V_{train}, and in this case $V_{train} \cap V_{val} = \emptyset$ and $V_{train} \cup V_{val} \cup V_{test} = V$.

Each node $u \in V$ (whether it is in the training or test set) also has a d dimensional node content feature vector $\mathbf{x}(u) \in \{0,1\}^d$. Later, we will use $\mathbf{f}_1, ..., \mathbf{f}_d$ to refer to the column vectors which are realizations of each feature in training set. $F_1, ..., F_d$ and R will denote the discrete random variables for each feature and the class label respectively.

In the pattern recognition scenario that we are interested in, given feature vectors of the training nodes and their labels, $\mathbf{x}(u)$ and $\mathbf{r}(u)$, $u \in V_{train}$, we need to train a mapping

(classifier) $g(\mathbf{x}(u)) : \{0,1\}^d \rightarrow \{0,1\}^C$ which achieves the best accuracy on the specific given test set:

$$acc(g, V_{test}) = \frac{1}{|V_{test}|} \sum_{v \in V_{test}} 1 - \delta[g(\mathbf{x}(v)), \mathbf{r}(v)]. \tag{1.1}$$

Here $\delta[\mathbf{p}, \mathbf{q}]$ returns 1 if vectors $\mathbf{p}$ and $\mathbf{q}$ differ in at least one position. Since the classifier $g(\mathbf{x}(u))$ uses only the input features, we will call it the content only classifier $g(\mathbf{x}(u)) = g_{CO}(\mathbf{x}(u))$.

When not only the training node features, but also their links are given, the link information can also be used for classification. Usually link information of neighbors of a specific node are taken into account, therefore we need to define the concept of a neighborhood. The neighborhood function $N(u)$ returns a set of nodes which are immediate neighbors of node u according to the links L: $N(u) = \{v : \{u, v\} \in L\}$.

We also define neighborhood in terms of neighbors that are more than one hop away. Let $SP_G(u, v)$ denote the number of edges (hops) on the shortest path between two nodes u and $v \in V$, and assign $SP_G(u, u) = 0$ and if u and v are not connected, then $SP_G(u, v) = \infty$. For each node $u \in V$, the h-neighborhood function $N_h(u)$ returns a set of nodes which according to the links L are neighbors of the node u that are at most h hops away from u:

$$N_h(u) = \{v : SP_G(u, v) \leq h\}. \tag{1.2}$$

Whenever we omit the subscript h, the neighborhood function denotes immediate neighborhood, i.e., $N(u) = N_1(u)$.

Classifiers need fixed dimensional inputs. So, we need to aggregate labels of neighbors of a node into a fixed dimensional vector. There are many different aggregation methods such as count, average, exists, weighted average (Lu & Getoor 2003, Perlich & Provost 2006) and the aggregation method which summarizes the neighbor labels to be most correlated with node label depends on the specific dataset (Sen & Getoor 2007). Among different aggregation methods, in this thesis, since it was used in many other studies on link-based classification, we use the count aggregation and define the

aggregated neighbor labels for a node u as:

$$\mathbf{r}_N(u) = \sum_{v \in N(u)} \mathbf{r}(v). \tag{1.3}$$

Based on the labels of the neighbors only, a classifier, which we call the link only (LO) classifier $g_{LO}(\mathbf{r}_N(u))$ can be trained on the training data. When a test node needs to be classified, if it has neighbors in test set, inputs to the classifier need to be determined iteratively, based on the current label assignment of the neighbors using a collective classification algorithm such as the Iterative Classification Algorithm (ICA) (Macskassy & Provost 2007, Sen et al. 2008).

When both node features and links are known, a classifier that uses both the content features of the node and labels of the neighbors have been used in (Sen et al. 2008). We will call this classifier with $d + C$ features, the content and link classifier:

$$g_{COLO}([\mathbf{x}(u)\ \mathbf{r}_N(u)]). \tag{1.4}$$

When there are $C > 2$ classes, we assume that the classifier outputs for all types of classifiers g_{CO}, g_{LO}, g_{COLO} are C dimensional vectors and each dimension corresponds to a class.

1.2.1 Sampling

The test data in social networks may contain a bunch of nodes that are connected to each other, in which case the training-validation partitioning process needs to take this dependency into account. It is also possible that the test data are randomly distributed among the training nodes. Two different sampling mechanisms, snowball sampling and random sampling, are used to handle these two situations.

1.2.1.1 Random sampling

When random sampling is used, nodes in training, validation, and test sets are selected. It is important to preserve class distribution of the dataset as much as possible during selection of the nodes. One method to achieve this, is to partition nodes from every class among themselves randomly proportional to the required train, validation and test

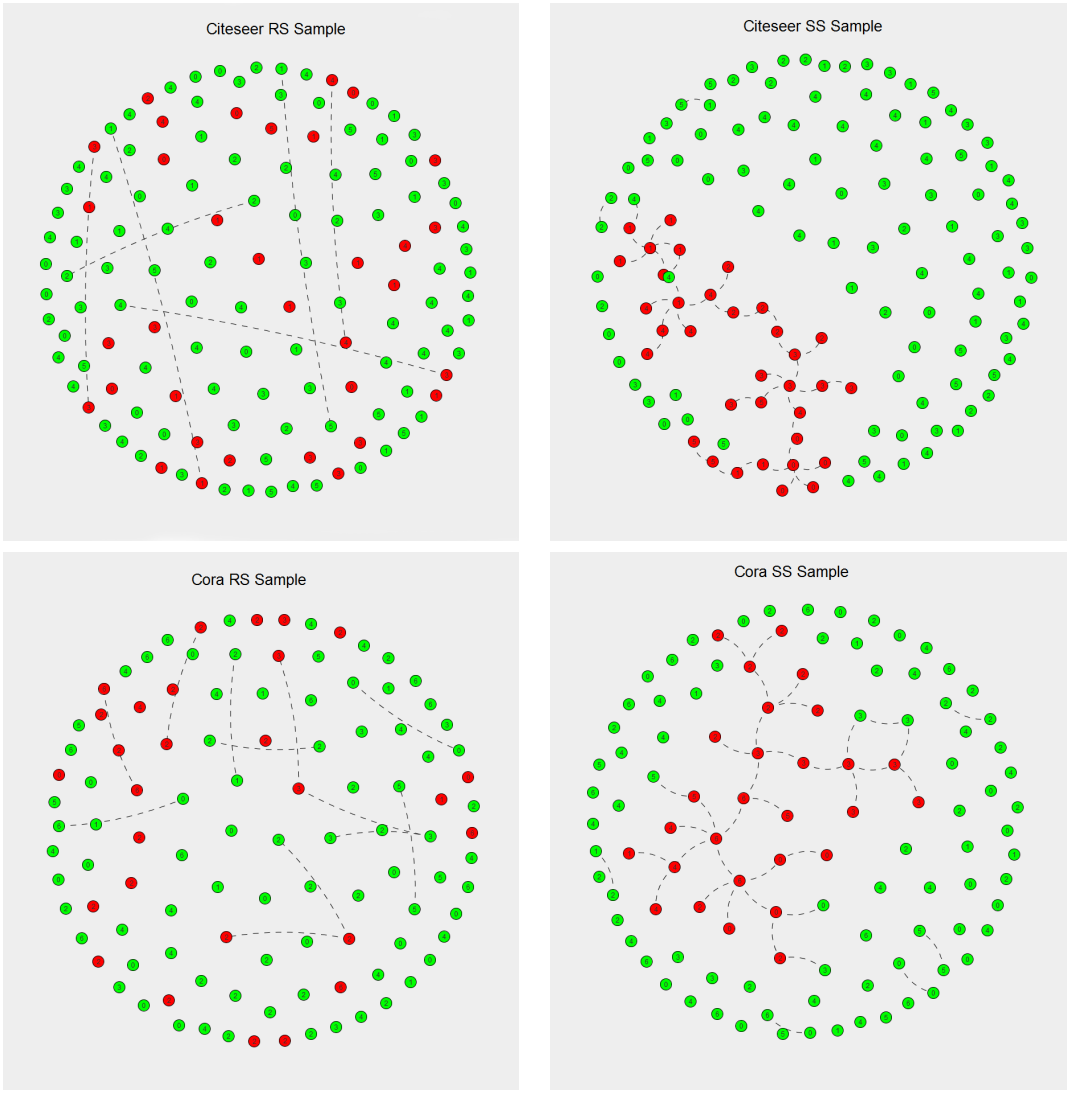


Figure 1.1 : Random sampling vs snowball sampling.

set sizes and then combine the nodes from every class in the training set to produce the training set and do the same for validation and test sets. While random sampling is a simple method, even if care is taken to preserve the class ratios, the sampled graph is likely to have very different topological properties than the original graph (Ahmed, Neville, & Kompella 2012), therefore, the classification algorithms trained/tested on the sampled graph may not perform similarly on the test set or the original dataset.

1.2.1.2 Snowball sampling

When the usual k-fold cross-validation training and test sets are obtained on networked data, especially when the number of links per node is low, k-fold random sampling may generate almost disconnected graphs (Sen et al. 2008), making learning through

the links in the networked data impossible. To overcome the issue of disconnected graphs in k-fold cross validation, snowball sampling is used. In snowball sampling, first of all, different starting nodes are selected. Then new nodes are selected among the nodes which are accessible through the selected nodes. Thus, the selected nodes grow like a snowball and as a result selected nodes are connected. It is important to preserve the class ratios in the selected set of nodes, therefore, at every point during sampling, if a class is underrepresented, it is given higher probability. This sampling procedure continues until there are enough nodes in the selected subset.

In Figure 1.1, visualization of train/test partitions for Cora and Citeseer datasets are given for both random sampling and snowball sampling. Red nodes are nodes in the test partition while green ones are in the train partition. Nodes' actual labels are also displayed in the circles representing the nodes. In order to be able to visualize these two sets a subsample of 4% of the actual data is used.

When random sampling is used, there is no order or dependency between the selection of training, validation and test sets. However, when snowball sampling is used, whether the snowball is selected and taken to be the test set or the training set may give different results. Taking the snowball to be the test set (Sen et al. 2008), generating k disjoint snowballs and using them for training-validation set formation (McDowell et al. 2007), using temporal sampling and past data for training and generating snowball samples with some portion of provided labels for validation (Neville & Jensen 2008) are all different uses of snowball sampling for classification of networked data. While some authors mention that snowball sampling causes bias toward highly connected nodes and may be more suitable to infer about links than to infer about nodes in a social network (Snijders 1992), others suggest that since snowball is not guaranteed to reach individuals with high connectivity and would not reach disconnected individuals, snowball's starting nodes should be chosen carefully (Hanneman & Riddle 2005).

When random sampling is used to generate k-fold cross validation training and validation (and test) sets, there are no overlaps between different test sets. However, when snowball sampling is used to generate the k test sets, the test snowballs created may overlap. Since linked instances in a snowball are correlated, errors made on them may also be correlated. The statistical tests, such as the paired t-test, which is used

for model selection, may not give reliable results when test sets are not independent (Neville, Gallagher, & Eliassi-Rad 2009). Forest Fire Sampling (FFS) (Leskovec & Faloutsos 2006) method may be considered as an alternative to snowball sampling, in this algorithm, as in snowball sampling, a breadth first search technique is used but some of the followed links are burned according to a probability distribution.

1.2.1.3 Sampling on streaming graphs

When the network has many nodes or the nodes are observed as a stream (as in the case of twitter for example), it is not possible to consider all of the graph when sampling. For streaming or large graphs the sampling algorithm needs to be space and time efficient. In (Ahmed et al. 2012) a sampling algorithm for streaming graphs called Partially-Induced Edge Sampling (PIES) is introduced. PIES algorithm always keeps a constant number of nodes in the sample and drops old nodes and adds new observed ones as the network keeps being observed. Note that the same idea can also be used when the graph is too large to fit in the memory, as new nodes are explored, they can be considered as a stream. (Ahmed et al. 2012) considers sampling algorithms based on network nodes, edge, and topology-based sampling for three different types of networks, static-small, static-large and streaming. Snowball sampling is a topology based sampling method. In (Ahmed et al. 2012), the authors' objective is to ensure that the sampled graph is a representative subgraph which matches the topological properties of the original graph.

1.2.2 Aggregation

Each node in a graph may have a different degree and therefore different number of neighbors. On the other hand, most classifiers need the input dimensionality for each instance to be the same. Therefore, in order to take advantage of neighbor link or feature information, a mechanism to make them the same dimensional, regardless of the identity of a node is needed. Aggregation methods (also called propositionalization methods or flattening methods (Preisach & Schmidt-Thieme 2008)) are often used for this purpose. In this section we give common aggregation methods used for aggregation of neighbor labels so that they can be used for classifier training.

The main objective of aggregation in relational modeling is to provide features which improve the generalization performance of the model (Perlich & Provost 2006). However aggregation usually causes loss of information, therefore one needs to be careful about not losing predictive information. Perlich and Provost proposed general guidelines for designing aggregation operators, suggesting that aggregation should be performed keeping the class labels under consideration, aggregated features should cause instances of the same class to be similar to each other and different aggregation operators should be experimented with. Below, we will present performances of different aggregation methods on different datasets. In (Perlich & Provost 2006) authors considered both simple aggregators and new more complex aggregators in the context of the relational learning system ACORA (Automated Construction of Relational Attributes). ACORA computes class-conditional distributions of linked object identifiers, and for an instance that needs to be classified, it creates new features by computing distances from these distributions to the values linked to the instance (Perlich & Provost 2006). Lu and Getoor considered various aggregation methods: existence (binary), mode and value counts. The count method performed best in their study (Perlich & Huang 2005).

1.2.3 Neighbor label aggregation methods

In the following sections, the notation introduced in Section 1.2 is used. Note that, although the neighborhood function $N(u)$ is usually defined to include the immediate neighbors of a node, it could be extended to include neighbors which are at most a number of links away.

- Count Method: The count aggregation method (Preisach & Schmidt-Thieme 2008) determines the frequency of the neighbors having the same class as the node:

$$\mathbf{r}_N^{count}(u) = \sum_{v \in N(u)} \mathbf{r}(v). \tag{1.5}$$

  The count method does not consider any uncertainty with the labels or links, neither does it consider the edge weights (Preisach & Schmidt-Thieme 2008).

- Mode Method: This aggregation method considers the mode of the neighbor labels:

$$\mathbf{r}_N^{mode}(u) = mode_{v \in N(u)} \mathbf{r}(v). \tag{1.6}$$

- Binary Existence Method: This aggregation method only considers whether a certain label exists among the neighbors or not, it does not take into account the number of occurrences, as count or mode aggregation do. For the jth class, the binary existence of neighbor labels' aggregation is computed as:

$$\mathbf{r}_N^{exist}(u,j) = [\mathbf{r}_N^{count}(u,j) > 0] \tag{1.7}$$

- Weighted Average Method: The weighted average aggregation method (Preisach & Schmidt-Thieme 2008) sums the weights of the neighbors of the node belonging to each class and then normalizes it with the sum of the weights of all edges to the neighbors. Similar to the count method, it does not consider uncertainty.

$$\mathbf{r}_N^{wavg}(u) = \frac{1}{Z} \sum_{v \in N(u)} w(u,v)\mathbf{r}(v). \tag{1.8}$$

Here $w(u,v) \in \mathscr{R}$ is the weight of the link between nodes u and v, Z is a normalization constant:

$$Z = \sum_{v \in N(u)} w(u,v) \tag{1.9}$$

- Probabilistic Weighted Average Method: This aggregation method is the probabilistic version of the Weighted Average method. It is based on the weighted arithmetic mean of class membership probabilities of neighbors of a node. This method was introduced by Macskassy and Provost and was used as a probabilistic Relational Classifier (PRN) classifier (Preisach & Schmidt-Thieme 2008).

$$\mathbf{r}_N^{pwavg}(u,c) = \frac{1}{Z} \sum_{v \in N(u), c \in C} w(u,v).P(c|v) \tag{1.10}$$

where Z is defined as in Eq. 1.9 and c denotes a certain class.

1.3 Classification with Networked Data

We consider two types of classification with networked data. The content only classification can be used whether the test nodes are known or not. On the other hand, when the test nodes or some unlabeled nodes are known, then semi-supervised classification algorithms can be used.

1.3.1 Supervised, content only classification

This model consists of a (learned) model, which uses only the local features of the nodes whose class label will be estimated. The local models can also be used to generate priors for the initial state for the relational learning and collective inference components. They also can be used as one source of evidence during collective inference. These models typically are produced by traditional machine learning methods (Macskassy & Provost 2007).

1.3.2 Semi-supervised and transductive classification

Since social network data usually come in huge sizes, in addition to labeled instances, there are, usually, a huge number of unlabeled instances. In such cases, it could be possible to use the information other than labels that exist in the unlabeled data, which leads to use of semi-supervised learning algorithms. When the test nodes whose class will need to be predicted are known, then we have a transductive learning scenario.

In contrast to the non-relational (local) model, the relational model use the relations in the network as well as the values of attributes of related entities, even possibly long chains of relations. In relational models, a relational classifier determines the class label or estimates the class conditional probabilities. The relational classifier might combine local features and the labels of neighbors using a naive Bayes model or a logistic regression (Macskassy & Provost 2007).

1.3.3 Collective classification

Collective classification methods, which are sometimes also called collective inference methods, are iterative procedures, which classify related instances simultaneously (Preisach & Schmidt-Thieme 2008, Yonghong, Tiejun, & Wen 2006). In collective classification, the content and link information for both training and test data are available. First, based on the available training content, link and label information, models are trained. Then, those models are used to label the test data simultaneously and iteratively where each test sample is labeled based on its neighbors.

Collective classification exploits relational autocorrelation. Relational autocorrelation is a very important property of relational data and is used as a measure of how an attribute for an instance is correlated with the same variable from a related instance (Preisach & Schmidt-Thieme 2008).

However, sometimes, the advantage of exploiting the relationships can become a disadvantage since it is possible to make incorrect predictions about a particular node which propagates in the network and may lead to incorrect predictions about other nodes. Bilgic and Getoor proposed an acquisition method which learns the cases when a given collective classification algorithm makes mistakes, and suggests label acquisitions to correct those mistakes (Bilgic & Getoor 2008).

Iterative classification algorithm (ICA) and Gibbs sampling algorithm (GS), Mean field relaxation labeling (MF), Loopy belief propagation (LBP), are popular approximate inference algorithms used for collective classification (Sen et al. 2008). In this thesis, we explain the ICA algorithm and use it in our experiments. Iterative classification algorithm (ICA) is a popular and simple approximate collective inference algorithm (Macskassy & Provost 2007, Sen et al. 2008). Despite its simplicity, ICA was shown to perform as well as the other algorithms such as Gibbs Sampling (Sen & Getoor 2007). Please see (Macskassy & Provost 2007, Sen et al. 2008) for details on the other collective classification algorithms.

1.3.3.1 Iterative classification algorithm (ICA)

To determine the label of a node, Iterative Classification Algorithm (ICA) assumes that all of the neighbors' attributes and labels of that node are already known. Then, it calculates the most likely label with a local classifier which uses node content and neighbors' labels. However, most nodes have neighbors which are not in training data and hence are not labeled, therefore the label assignment on one test instance may affect the label assignment on a related test instance. ICA repeats the labeling process iteratively until all of the label assignments are stabilized. Neighbor label information is summarized using an aggregation operator (See Section 1.2.3).

Pseudocode for the ICA algorithm (based on (Sen et al. 2008)) is given in Algorithm 1. In the pseudo code, $\tilde{\mathbf{r}}(u)$ stands for temporary label assignment of instance u in the

Table 1.1 : Summary information about real world datasets.

	Citeseer	Cora	WebKb	HepTh
Number of Nodes	3312	2708	877	1559
Number of Features	3703	1433	1703	2295
Number of Classes	6	7	5	4
Number of Links (Link)	4536	5278	1388	2500
Number of Links (Content)	6643	5945	2159	3868

test set. $g_{CL}([\mathbf{x}(u)\ \mathbf{r}_N(u)])$ is the base classifier which is first trained on training nodes and their neighbors from the training set. The base classifier uses the estimated labels of the neighbors if they are test nodes. O is a random ordering of test nodes.

Algorithm 1 $\tilde{\mathbf{r}}(V_{test}) = \text{ICA}(G, V_{train}, V_{test}, g_{CL}())$

```
for all u ∈ V_test do
    Compute r̃_N(u) using only neighbors in V_train
    Set r̃(u) ← g_CL([x(u) r̃_N(u)])
end for
repeat
    Generate ordering O over nodes in V_test
    for all u ∈ O do
        Compute r̃_N(u) using current label assignments to nodes in N(u)
        Set r̃(u) ← g_CL([x(u) r̃_N(u)])
    end for
until all labels are stabilized or threshold number of iterations
```

The ICA algorithm starts with a bootstrapping to assign initial temporary labels to all nodes by using only the content features of the nodes. Then, it starts iterating and updating labels according to the both relational and content features (Sen & Getoor 2006).

1.4 Datasets

We used three datasets that have been used in network classification research (Macskassy & Provost 2007, McDowell et al. 2007, Sen et al. 2008) and prepared one dataset for network classification. We give their graph properties in Table 1.1. We also created synthetic datasets, whose graph properties are given in Table 1.2.

1.4.1 Cora dataset

Cora (McCallum, Nigam, Rennie, & Seymore 2000) dataset consists of information on 2708 Machine Learning papers. Every paper in Cora cites or is cited by at least one other paper in the dataset. There are 1433 unique words that are contained at least 10 times in these papers. There are also 7 classes assigned to the papers according to their topics. For each paper, whether or not it contains a specific word, which class it belongs to, which papers it cites and which papers it is cited by are known. Citation connections and paper features (class and included words) are contained in two separate files. Total number of connections between the papers is 5278. There are 3.898 links per paper.

1.4.2 Citeseer dataset

Citeseer (Giles, Bollacker, & Lawrence 1998, Sen & Getoor 2007) dataset consists of information on 3312 scientific papers. There are 3703 unique words that are contained at least 10 times in these papers. There are 6 classes assigned to the papers according to their topics. Just as in the Cora dataset, word, class and cites and cited by information are given in two separate files. Total number of connections between the papers is 4536. There are 2.74 links per paper.

1.4.3 WebKB dataset

WebKB (DiPasquo et al. 1998) dataset consists of sets of web pages from four computer science departments, with each page manually labeled into 5 categories: course, project, staff, student, or faculty.

Link structure of WebKB dataset is different from Cora and Citeseer datasets since co-citation links are useful for WebKB dataset. The reason for that can be explained based on the observation that a student is more likely to have a hyperlink to her adviser or a group/project page rather than to one of her peers (Macskassy & Provost 2007).

Table 1.2 : Summary information about synthetic datasets.

	Synthms0	Synthms16	Synthms32
Number of Nodes	1000	1000	1000
Number of Features	32	32	32
Number of Shared Features	0	16	32
Number of Classes	2	2	2
Number of Links (Link)	2072	2207	2230
Number of Links (Content)	2065	2070	2230

1.4.4 HepTh dataset

HepTh is a dataset which consists of papers in theoretical high-energy physics. The dataset is downloaded from Knowledge Discovery Laboratory Group web site (https://kdl.cs.umass.edu/display/public/HEP-Th) at the University of UMass Amherst. The original dataset consists 42319 objects and 532429 links. It is composed of different type of objects, namely Journal, Paper, Author and Email-Domain. We preprocessed the dataset to include only the papers and the citation links between them. Since only 2542 of 29555 papers have at least one assigned label, we first filtered only these labeled instances and the links between them. The filtered dataset has still 92 different labels, which is reduced to 54 different labels after a few preprocessing such as revision of typos, replacing blanks, etc. After that, we omitted the labels and thus the instances which have only a few instances. The resulting dataset has 1559 nodes and 2516 links. For content features, we found the matching abstracts of those papers. After stop words removal and stemming by filtering the words at least 50 times occurred in all of the documents, we created a bag of words of size 2294. We also included "isPaperPublished" property as content feature, resulting with 2295 content features per node. The resulting set has 4 different types of labels.

1.4.5 Synthetic datasets

To create synthetic networked data, we used a method that allows varying content and link relevances with the class label and varying dependence (redundancy) between content and link. As in the "content based" networks of (Balcan & Erzan 2004), we generated content and link bits, and based on their link similarity we connected the nodes in the network. Content features $\mathbf{x}(u)$ are produced for each node u. In order to

Table 1.3 : Acc. comp. of aggregation methods (ICA, SS) (Cora, Citeseer, WebKB).

	Cora	CiteSeer	WebKB
Count Method	75.39 ± 1.02	68.86 ± 1.22	84.75 ± 0.67
Mode Method	75.42 ± 0.98	69.79 ± 1.00	81.68 ± 0.43
Binary Existence Method	71.33 ± 1.65	68.58 ± 0.71	80.32 ± 0.86
Proportion Method	76.46 ± 1.31	66.81 ± 0.85	76.45 ± 0.93
Pr. Weighted Average Method	64.43 ± 1.15	66.81 ± 0.85	77.82 ± 0.99

Table 1.4 : Acc. comp. of aggregation methods (ICA,RS) (Cora, Citeseer, WebKB).

	Cora	CiteSeer	WebKB
Count Method	87.78 ± 0.48	77.43 ± 0.67	87.01 ± 1.01
Mode Method	87.78 ± 0.59	78.13 ± 0.66	85.17 ± 1.93
Binary Existance Method	85.19 ± 0.61	77.58 ± 0.78	86.78 ± 1.11
Proportion Method	86.48 ± 0.63	77.76 ± 0.89	86.32 ± 1.17
Pr. Weighted Average Method	70.15 ± 0.74	70.76 ± 0.93	86.32 ± 1.04

produce links between nodes, a similarity measure between them is needed. We used an integer power of inverse normalized hamming distance as the similarity measure. Class labels are determined according to the mode of the complete feature vector. We used the same datasets produced as in (Cataltepe et al. 2011). Please see (Cataltepe et al. 2011) for more details.

1.5 ICA Performance on the Datasets

In this section, we compare the performance of different sampling and aggregation methods on our datasets. In Tables 1.3 and 1.4, we show the average test accuracies over 10 folds, of using iterative classification algorithm (ICA). We used logistic regression as the base classifier, during these experiments.

As it can be seen from both tables, the count method outperforms the other methods both in terms of its simplicity and the accuracies obtained. The test accuracies are higher when random sampling is used. This is due to the fact that when random sampling is used independent instances are more likely to be selected, therefore the effective number of training instances is higher than snowball sampling.

2. GRAPH PROPERTIES AND COLLECTIVE CLASSIFICATION

2.1 Background and Purpose

Network data consist of a graph with nodes, node features and labels and links between the nodes. Graph properties, such as homophily, degree distribution help us understand the data at hand. Graph properties can also be used as guidelines in finding out if a sampling method (Section 1.2.1) used for evaluation of algorithms is a good one or not, or which type of aggregation (Section 1.2.2) of neighbor labels should be used for classification. In this section, we define some of the important graph properties from the literature (for example (M. E. J. Newman 2003b) or (Dorogovtsev & Mendes 2002)) that we use in the thesis. We also introduce some new graph properties such as the local alpha which is the average accuracy of a node's neighbors in training set, and the local beta which is the average homophily of the node's neighbors in training set.

Some of the graph properties such as degree distribution, clustering coefficient or betweenness centrality can be calculated when only links are known, without a need for the labels. We name these graph properties the unlabeled graph properties. Homophily, entropy and local beta, on the other hand, are labeled graph properties. Accuracy and local alpha are graph properties which are related to the classifiers which have been trained on the available dataset.

In this section, certain graph properties of Citeseer, Cora, WebKb, HepTh and Synthetic datasets are compared. Visualization of these properties with respect to each other and correlations between graph properties are presented. Graph properties are computed on the original graph, link graph (G_{LO}), that is available with the dataset. We also produce the content graph (G_{CO}) by means of linking nodes whose contents' cosine similarity (Salton 1989) is above a certain threshold.

2.2 Methodology

2.2.1 Homophily

A classification algorithm in a social network aims to use both content and link information. Whether the link information will be useful or not depends on whether linked objects have similar features and/or labels. These similarities have been quantified using a number of different criteria.

Neville and Jensen defined two quantitative measures of two common characteristics of relational datasets: *concentrated linkage* and *relational autocorrelation*. Concentrated linkage occurs when many entities are linked to a common entity like the citation links to the key papers. Relational autocorrelation occurs when the features of the entities are similar among entities that share a common neighbor (Jensen & Neville 2002). As pointed out by Neville and Jensen, most of the models (e.g., PRMs, RMNs) do not automatically identify which links are the most relevant to the classification task. In their method, they defined links which are most relevant to the classification task by using concentrated linkage and relational autocorrelation, and explored how to use these relational characteristics to improve feature selection in relational data (Yonghong et al. 2006).

Yang et al. identified five hypertext link regularities that might (or not) hold in a particular hypertext corpus (Yang, Slattery, & Ghani 2002, Yonghong et al. 2006). We list three of them here.

- *Encyclopedia regularity:* The class of a document is the same as the class of the majority of the linked documents.
- *Co-referencing regularity:* Documents with the same class tend to link to documents not of that class, but which are topically similar to each other.
- *Partial co-referencing regularity:* Documents with the same class tend to link to documents that are topically similar to each other, but also link to a wide variety of other documents without semantic reason. The presence (or absence) of these

regularities may significantly influence the optimal design of a link-based classifier. Most of link analysis methods and link-based classification models are built upon the "encyclopedia" or "co-referencing" regularity. As a result, the models do not automatically identify which links are most relevant to the task (Yonghong et al. 2006).

Assortativity index defined by (M. E. J. Newman 2003b) is also a measure of how similar are the labels of connected nodes. Assortativity index is defined using the number of links which connect nodes of same and different classes and it is proportional to the number of links that connect nodes of the same class.

Homophily (Macskassy & Provost 2007, Provost, Perlich, & Macskassy 2003) or label autocorrelation can be defined as the tendency of entities to be related to other similar entities, i.e. linked entities generally have a tendency to belong to the same class. High homophily usually implies that using link information helps with classification while for low homophily datasets using a content only classifier could do a better job.

Using the notation that we introduced in Section 1.2, we define homophily of a node u as the proportion of neighbor nodes of u which have the same label as u:

$$H(u) = \frac{1}{N(u)} \sum_{v \in N(u)} [\mathbf{r}(u) == \mathbf{r}(v)] \tag{2.1}$$

In this equation [] is the Iverson bracket and $[p]$ is equal to 1 if p is true, it is 0 otherwise. The graph homophily is then:

$$H(G) = \frac{1}{|V|} \sum_{u \in V} H(u) \tag{2.2}$$

Note that although homophily is defined as label "sameness", labels could be related to each other, and for example could just be the opposite of each other for a binary classification problem (i.e. heterophily). The classifiers that we use would be able to take advantage of homophily, heterophily or any other correlation between the label of a node and labels its neighbors.

2.2.2 Degree distribution

Degree, $k(u)$, of a node u is the total number of its connections to the other nodes in the network. Although the degree of a node seems to be a local quantity, degree distribution of the network often may help to determine some important global characteristics of networks. A scale-free network is type of a network whose degree distribution follows a power law (Dorogovtsev & Mendes 2002). It was shown that whether an epidemic spreads in a scale-free network or stops can be computed based on the scale free exponent in (Hein, Schwind, & Konig 2006).

2.2.3 Clustering coefficient

Clustering coefficient is a measure of degree to which nodes tend to cluster together in a graph. Clustering coefficient property can give information, for example, on how close are two nodes in the graph, and therefore how much their predicted labels would affect each other during prediction of labels. Different types of clustering coefficients can be defined as follows:

2.2.3.1 Global clustering coefficient

The global clustering coefficient, which is a measure of indication of the clustering in the whole network is based on triplets of nodes. A triplet is called an open triplet when three nodes are connected with two links or it is called a closed triplet when all three nodes are tied together. Three closed triplets, one centered on each of the nodes, form triangle. The global clustering coefficient is defined as the ratio of the number of closed triplets to the total number of triplets (open and closed) (Dorogovtsev & Mendes 2002):

$$CC_G(G) = \frac{Number\,of\,closed\,triplets}{Number\,of\,connected\,triplets} \tag{2.3}$$

2.2.3.2 Local clustering coefficient

Local clustering coefficient is defined in (Dorogovtsev & Mendes 2002) as the ratio of the actual number of edges between the neighbors of a node u to all the possible

numbers of edges is the local clustering coefficient for node u:

$$CC_L(u) = \frac{\sum_{v_1,v_2 \in N(u), v_1 \neq v_2} [\{v_1, v_2\} \in E]}{|N(u)|(|N(u)|-1)/2} \tag{2.4}$$

Here [] is the Iverson bracket. The number of all possible undirected links between neighbors $N(u)$ of node u is $|N(u)| * (|N(u)|-1)/2$.

2.2.3.3 Average clustering coefficient

The network average clustering coefficient is defined by Watts and Strogatz as the average of the local clustering coefficients of all the nodes in the graph (Watts & Strogatz 1998):

$$CC_L(G) = \frac{1}{|V|} \sum_{u \in V} CC_L(u) \tag{2.5}$$

If the average clustering coefficient is zero, then the graph is a tree. If the average clustering coefficient is higher than a random graph with the same degree distribution, then the network may show the small world phenomenon, i.e. any two random nodes can be connected using much smaller number of links than $O(|V|)$. It should be noted that during calculation of the average clustering coefficient, the nodes having less than two neighbors, which naturally have a clustering coefficient of zero, are not considered.

2.2.4 Rich club coefficients

Let $V_k \subseteq V$ denote the nodes having degree higher than a given value k and $E_{>k}$ denote the edges among nodes in V_k. The rich-club coefficient is defined as:

$$\phi_k(G) = \frac{|E_{>k}|}{|V_{>k}|(|V_{>k}|-1)/2} \tag{2.6}$$

In Equation 2.6 $|V_{>k}|(|V_{>k}|-1)/2$ represents the maximum possible number of undirected links among the nodes in $V_{>k}$. Thus, $\phi(k)$ measures the fraction of edges actually exist between those nodes to the the maximum number of edges they may have. The rich club coefficient helps to understand important information about the underlying architecture revealing the topological correlations in a complex network.(Colizza, Flammini, Serrano, & Vespignani 2006)

2.2.5 Degree-degree correlation

Degree-degree correlation is the correlation between the number of neighbors (minus 1) of neighboring nodes. Both homophily and degree-degree correlation can be considered as special case of assortatitivy (M. E. J. Newman 2003a), the correlation between a certain property (label, degree, etc.) of two neighboring nodes, average degree of the neighbors' of the nodes having degree k. (Kahng, Oh, Kahng, & Kim 2003) found out that among scale-free networks authorship and actor networks are assortative (i.e. nodes with large degree connect to nodes with large degree) while protein-protein interaction and world wide web networks are disassortative (i.e. large degree nodes tend to connect to small degree nodes).

2.2.6 Graph radius and diameter

These two notions are related to each other. We use the definitions from (Erdos, Pach, Pollack, & Tuza 1989).

Let $d_G(u,v)$ be the distance (i.e. the minimum number of edges that connect u and v) between nodes u and v. The eccentricity $\varepsilon(u)$ of a node u is defined as the greatest distance between u and any other node in the graph:

$$\varepsilon(u) = \max_{v \in V} d_G(u,v) \tag{2.7}$$

The diameter of a graph is defined as the length of the longest of the shortest paths between any two nodes or equivalently as the maximum eccentricity of any vertex:

$$diam(G) = \max_{u,v \in V} d_G(u,v) \tag{2.8}$$

The radius of the graph is defined as the minimum eccentricity among all the nodes in the graph:

$$rad(G) = \min_{u \in V} \varepsilon(u) = \min_{u \in V} \max_{v \in V} d_G(u,v) \tag{2.9}$$

2.2.7 Average path length

The average path length of a graph is defined as the average of all shortest paths between nodes:

$$APL(G) = avg_{u \in V} \max_{v \in V} d_G(u, v) \tag{2.10}$$

(Fronczak, Fronczak, & Holyst 2004) has computed the average path length based on whether the graph is a scale free one or not and the scale free exponent.

2.2.8 Graph density

The density of a graph is defined as the ratio of the number of edges to the number of possible edges:

$$D(G) = \frac{|E|}{|V|(|V|-1)/2} \tag{2.11}$$

2.2.9 Entropy

The entropy of a node is defined as the entropy of the probability distribution of its aggregated neighbor labels. Using the notation that we introduced in Section 1.2, let $\mathbf{r}_{\mathbf{N}(\mathbf{u})}$ be the aggregated neighbor labels given in equation 1.3, and C be the number of classes. Let $r_i(v) \in \{0,1\}$ denote whether node v is in class i or not. Let $r_{N,i}(u)$ denote the ith entry of the aggregated neighbors label for node u.

We define the probability of class i in the aggregated neighbor labels as:

$$p_i = r_{N,i}(u) = \sum_{v \in N(u)} r_i(v). \tag{2.12}$$

The entropy of a node u is defined as:

$$E(u) = \sum_{i=1}^{C} -p_i \log_2 p_i \tag{2.13}$$

The entropy of the graph is then:

$$E(G) = \frac{1}{|V|} \sum_{u \in V} E(u) \tag{2.14}$$

If a node has no neighbors, its entropy is defined as 0.

2.2.10 Local alpha

In order to be able to take into account classifier's performance for each node separately, we use the Local alpha, which is the local average of the classifier accuracy on the neighbors from training data. We use this node property as a means to weigh each classifier for node classification. Based on different types of classifiers, we introduce $\alpha_{g_{CO}}(u) \in R$, $\alpha_{g_{LO}}(u) \in R$ and $\alpha_{g_{ICA}}(u) \in R$ (Cataltepe et al. 2011).

The weights $\alpha_{g_{LO}}(u)$ and $\alpha_{g_{ICA}}(u)$ can be determined locally, based on the correct classification rate of those classifiers in the neighborhood of u according to edges in the link graph $G_{LO} = G$. In order to compute $\alpha_{g_{LO}}(u)$ and $\alpha_{g_{ICA}}(u)$, we first find nodes which are in the h-neighborhood of u in graph G_{LO}, $N_{h,G_{LO}}(u)$. (See Equation 1.2). We introduce G_{LO} in $N_{h,G_{LO}}(u)$, so that it is clear that the neighborhood is according to the G_{LO} graph. Then we compute the local average accuracy of the classifier g_{LO} and g_{ICA} within $N_{h,G_{LO}}(u)$ as:

$$\alpha_{g_x}(u) = \overline{acc(g_x,u)} = \frac{1}{|N_{h,G_{LO},train}(u)|} \sum_{v \in N_{h,G_{LO},train}(u)} acc(g_x,v), x = LO, ICA. \quad (2.15)$$

Here $N_{h,G_{LO},train}(u)$ denotes nodes from training data which are in h-neighborhood of node u in G_{LO} graph.

$$\alpha_{g_{CO}}(u) = \overline{acc(g_{CO},u)} = \frac{1}{|N_{h,G_{CO},train}(u)|} \sum_{v \in N_{h,G_{CO},train}(u)} acc(g_{CO},v). \quad (2.16)$$

Similarly, in order to determine $\alpha_{g_{CO}}(u)$, we first create a graph G_{CO} based on the content similarities of the nodes. We use cosine similarity in this work, however other similarity measures could also be used. In the content graph, we join nodes whose similarity are above a threshold whose value is chosen so that the average degree of the content graph G_{CO} is as close as possible to the average degree of the link only graph G. Once the content graph is produced, $\alpha_{g_{CO}}(u)$ is produced as in Equation 2.16.

2.2.11 Local beta

Local beta for a node is the local average of homophily of the neighbors of the node from training data. In order to compute $\beta_i(u)$, first the nodes, which are in the

Table 2.1 : Graph properties of real world datasets.

	Citeseer	Cora	WebKb	HepTh
Number of Nodes	3312	2708	877	1559
Number of Features	3703	1433	1703	2295
Number of Classes	6	7	5	4
Number of Links (Link)	4536	5278	1388	2500
Number of Links (Content)	6643	5945	2159	3868
Average Degree (Link)	2,74	3,90	3,17	3,21
Average Degree (Content)	4,01	4,39	4,92	4,96
Homophily (Link)	0,71	0,83	0,13	0,58
Homophily (Content)	0,39	0,08	0,19	0,12
Clustering Coefficient (Link)	0,24	0,29	0,29	0,41
Clustering Coefficient (Content)	0,69	0,84	0,82	0,83
Diameter (Link)	28	19	8	19
Diameter (Content)	10	5	5	5
Average Path Length (Link)	9,3	6,3	3,1	7,3
Average Path Length (Content)	3,3	2,1	2,1	2,2

h-neighborhood of u in the corresponding graph G_i, $N_{h,G_i(u)}$ are found. (See Equation 1.2.)

G_i in $N_{h,G_i}(u)$ is the neighborhood according to the G_i graph. Then the local beta is computed by

$$\beta_{g,i}(u) = \frac{1}{|N_{h,G_i,train}(u)|} \sum_{v \in N_{h,G_i,train}(u)} homophily(v). \tag{2.17}$$

Here $N_{h,G_i,train}(u)$ denotes nodes from training data which are in h-neighborhood of node u in G_i graph.

2.3 Experiments

We used four datasets Citeseer, Cora, WebKb and HepTh that have been used in network classification research (See Section 1.4). In Table 2.1, we show the graph properties computed on these datasets. We also created synthetic datasets (See Section 1.4). In Table 2.2, we show the graph properties computed on synthetic datasets.

Table 2.2 : Graph properties of synthetic datasets.

	Synthms0	Synthms16	Synthms32
Number of Nodes	1000	1000	1000
Number of Features	32	32	32
Number of Shared Features	0	16	32
Number of Classes	2	2	2
Number of Links (Link)	2072	2207	2230
Number of Links (Content)	2065	2070	2230
Average Degree(Link)	4,14	4,41	4,46
Average Degree(Content)	4,13	4,14	4,46
Homophily(Link)	0,73	0,76	0,80
Homophily(Content)	0,75	0,75	0,80
Clustering Coefficient(Link)	0,0967	0,1024	0,1136
Clustering Coefficient(Content)	0,0873	0,0961	0,1136
Diameter(Link)	13	12	11
Diameter(Content)	12	13	11
Average Path Length(Link)	5,4	5,2	5,2
Average Path Length(Content)	5,4	5,4	5,2

2.3.1 Analysis of average local accuracy values

In Figure 2.1, we show the correlation between the accuracy $acc(u)$ of a node u and the average accuracy of its neighbors $\alpha(u)$ for the Cora dataset. Since accuracy depends on the classifier used, we show the correlations when logistic regression and Bayes net classifiers are used. For link only classification, the average local accuracy values are more correlated with accuracy for logistic regression classifier. On the other hand, for content only classification, the correlation is higher for the Bayes net classifier. As it can be seen in the figures, usually the correlation between the accuracy of a node and its neighbors decreases as the size of the neighborhood (h in $N_h(u)$) increases. Except, with the content only classifier, the correlation is maximized for a neighborhood of size 2.

2.3.2 Content similarity

Cosine similarity is used to build content graph from content features. Average degree versus similarity graphs give us an idea about average degree of the resulting content graph in case of selection of that similarity threshold. Average degree vs. similarity

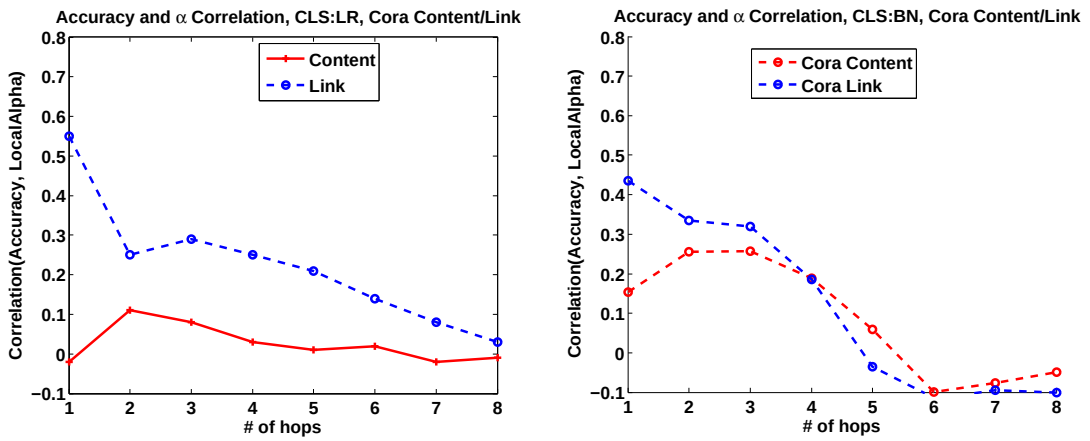


Figure 2.1 : Correlation between accuracy of the classifier at a node and its accuracy within the h-hop-neighborhood of the node for Cora dataset and using Logistic Regression (left side) and Bayes Net (right side) classifiers.

graphs are given in Figure 2.2. According to the figure, similarity between nodes are much lower on Cora and Citeseer datasets compared to Synthetic datasets. To obtain a similar graph in terms of average degree to the corresponding datasets' link graphs, a similarity threshold value about 0.2 is enough for Citeseer dataset, while it is about 0.25 on Cora dataset. For Synthetic datasets the similarity threshold value should be about 0.95 for the same case. The behavior of WebKB dataset is similar to Cora and Citeseer datasets', however it is a bit relaxed version of them. It is also observed that content of HepTh dataset is similar to Synthetic datasets'. For these datasets, most of the content features of the nodes are similar to each other and only a few content features differ, which is contrast to Cora and Citeseer datasets.

2.4 Correlations between Graph Properties

2.4.1 Correlations between graph properties of content graph

Correlations between graph properties of the content graph of the datasets are calculated and given in Tables 2.3, 2.4, 2.6, 2.5, 2.7, 2.8 and 2.9. The results show that betweenness centrality is highly correlated with degree, entropy is negatively correlated with homophily and highly correlated with degree on all datasets. Homophily and accuracy are correlated. On HepTh dataset, degree is highly negatively correlated with clustering coefficient. See Figure 2.3.

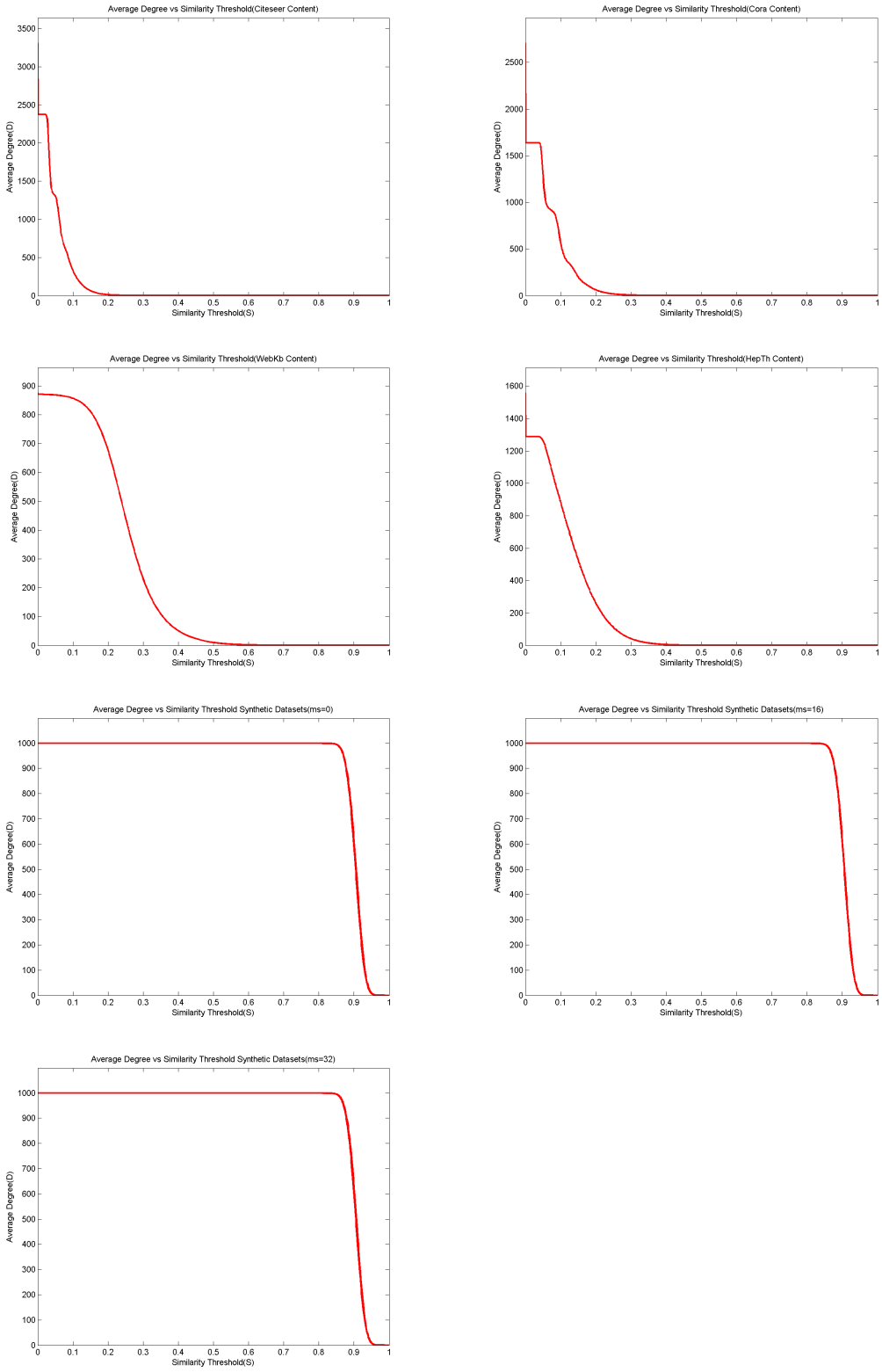


Figure 2.2 : Average degree vs similarity.

Table 2.3 : Citeseer content only correlations results.

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	acc (CO)
k (CO)	-0,06	0,01	0,39	0,65	0,13	-0,10	-0,03
c (CO)	-	0,18	-0,27	-0,34	-0,08	0,11	-0,01
h (CO)	-	-	-0,49	-0,08	0,03	0,07	0,11
ent (CO)	-	-	-	0,38	0,04	-0,16	-0,07
betw (CO)	-	-	-	-	0,04	-0,02	-0,03
alpha (CO)	-	-	-	-	-	-0,88	0,09
beta (CO)	-	-	-	-	-	-	0,04

Table 2.4 : Cora content only correlations results.

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	acc (CO)
k (CO)	-0,08	-0,11	0,41	0,81	-0,18	0,37	0,02
c (CO)	-	0,11	-0,18	-0,14	0,03	0,06	0,00
h (CO)	-	-	-0,56	-0,09	0,30	0,11	0,12
ent (CO)	-	-	-	0,24	-0,38	0,07	-0,07
betw (CO)	-	-	-	-	-0,15	0,16	0,01
alpha (CO)	-	-	-	-	-	0,25	0,18
beta (CO)	-	-	-	-	-	-	0,05

Table 2.5 : WebKb content only correlations results.

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	accuracy (CO)
k (CO)	-0,25	-0,07	0,48	0,48	0,22	0,28	0,08
c (CO)	-	-0,10	-0,12	-0,42	-0,27	-0,06	-0,08
h (CO)	-	-	-0,51	-0,08	-0,10	-0,11	0,11
ent (CO)	-	-	-	0,25	0,26	0,23	-0,06
betw (CO)	-	-	-	-	0,12	0,07	0,05
alpha (CO)	-	-	-	-	-	0,78	-0,11
beta (CO)	-	-	-	-	-	-	-0,02

Table 2.6 : HepTh content only correlations results.

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	accuracy (CO)
k (CO)	-0,80	-0,02	0,53	0,67	-0,37	-0,16	0,02
c (CO)	-	-0,06	-0,40	-0,51	0,39	-0,00	0,03
h (CO)	-	-	-0,24	-0,00	-0,06	0,40	0,11
ent (CO)	-	-	-	0,21	-0,23	-0,57	0,01
betw (CO)	-	-	-	-	-0,13	-0,05	0,03
alpha (CO)	-	-	-	-	-	0,03	0,10
beta (CO)	-	-	-	-	-	-	0,03

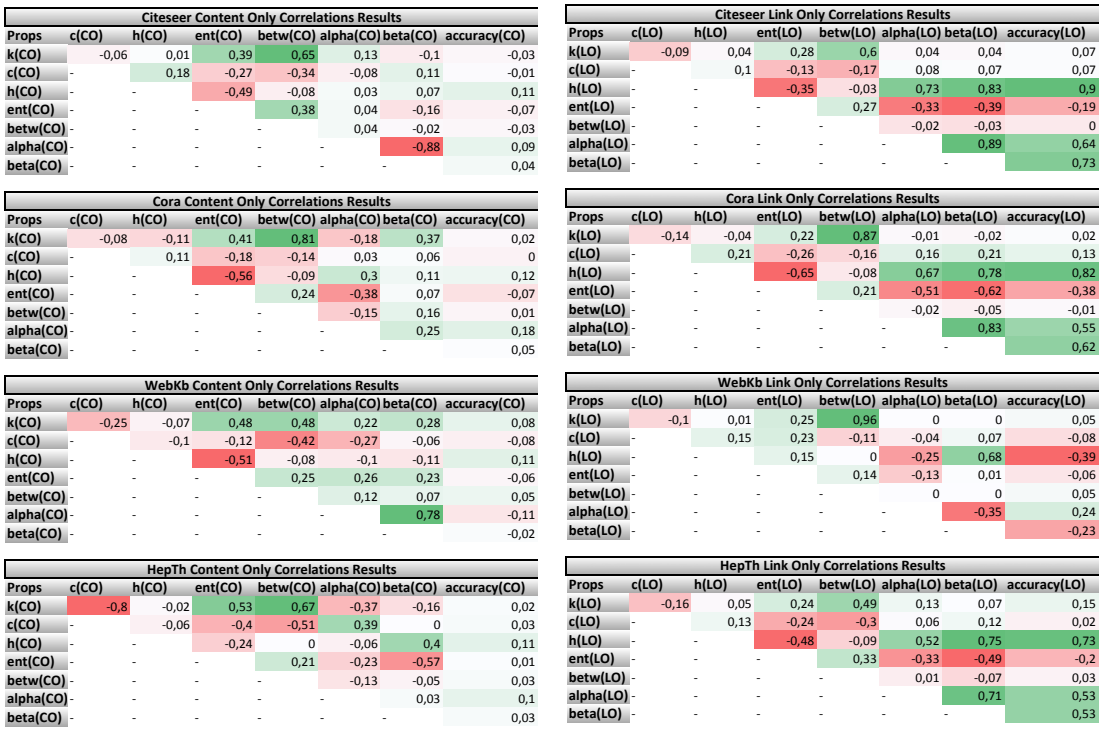


Figure 2.3 : Correlations highlight.

Table 2.7 : Synthetic datasets meta correlations (CO) results (ms=0).

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	accuracy (CO)
k (CO)	-0,08 ± 0,01	0,32 ± 0,01	0,14 ± 0,01	0,90 ± 0,00	-0,05 ± 0,11	-0,04 ± 0,20	0,13 ± 0,00
c (CO)	-	-0,02 ± 0,03	-0,23 ± 0,02	-0,12 ± 0,00	0,08 ± 0,03	0,08 ± 0,05	-0,04 ± 0,02
h (CO)	-	-	0,04 ± 0,02	0,20 ± 0,01	-0,02 ± 0,06	0,04 ± 0,12	0,16 ± 0,01
e (CO)	-	-	-	0,13 ± 0,01	0,07 ± 0,02	0,05 ± 0,03	0,01 ± 0,01
b (CO)	-	-	-	-	-0,11 ± 0,08	-0,04 ± 0,13	0,09 ± 0,00
alpha (CO)	-	-	-	-	-	-0,20 ± 0,14	0,13 ± 0,01
beta (CO)	-	-	-	-	-	-	-0,00 ± 0,04

2.4.2 Correlations between graph properties of link graph

Correlations between graph properties of the link graph of the datasets are calculated and given in Tables 2.10, 2.11, 2.13, 2.12, 2.14, 2.15 and 2.16. Results show that homophily and accuracy are highly correlated as expected. Homophily and entropy are negatively correlated as expected. Local alpha and accuracy are highly positively correlated for Citeseer, Cora and HepTh datasets, while as weakly positive correlated for WebKb dataset. Local alpha and local beta are positively highly correlated. Local beta and homophily are also correlated as expected. See Figure 2.3.

Table 2.8 : Synthetic datasets meta correlations (CO) results (ms=16).

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	accuracy (CO)
k (CO)	-0,08 ± 0,01	0,34 ± 0,01	0,12 ± 0,01	0,90 ± 0,01	-0,02 ± 0,14	-0,17 ± 0,21	0,12 ± 0,00
c (CO)	-	-0,02 ± 0,02	-0,21 ± 0,02	-0,12 ± 0,00	0,07 ± 0,02	0,00 ± 0,04	-0,03 ± 0,01
h (CO)	-	-	0,01 ± 0,02	0,22 ± 0,01	-0,02 ± 0,08	-0,02 ± 0,13	0,16 ± 0,01
e (CO)	-	-	-	0,13 ± 0,01	0,05 ± 0,03	0,01 ± 0,03	-0,01 ± 0,01
b (CO)	-	-	-	-	-0,06 ± 0,10	-0,12 ± 0,15	0,08 ± 0,01
alpha (CO)	-	-	-	-	-	-0,54 ± 0,06	0,14 ± 0,03
beta (CO)	-	-	-	-	-	-	-0,02 ± 0,04

Table 2.9 : Synthetic datasets meta correlations (CO) results (ms=32).

Props	c (CO)	h (CO)	ent (CO)	betw (CO)	alpha (CO)	beta (CO)	accuracy (CO)
k (CO)	-0,07 ± 0,01	0,35 ± 0,01	0,09 ± 0,01	0,91 ± 0,01	-0,01 ± 0,13	-0,09 ± 0,20	0,05 ± 0,01
c (CO)	-	0,01 ± 0,02	-0,24 ± 0,01	-0,12 ± 0,01	0,03 ± 0,02	0,06 ± 0,05	0,00 ± 0,01
h (CO)	-	-	0,02 ± 0,02	0,22 ± 0,01	-0,01 ± 0,09	0,07 ± 0,14	0,05 ± 0,00
e (CO)	-	-	-	0,12 ± 0,01	-0,00 ± 0,01	0,02 ± 0,02	0,00 ± 0,01
b (CO)	-	-	-	-	-0,03 ± 0,10	-0,08 ± 0,14	0,04 ± 0,00
alpha (CO)	-	-	-	-	-	-0,47 ± 0,05	0,17 ± 0,02
beta (CO)	-	-	-	-	-	-	-0,02 ± 0,02

Table 2.10 : Citeseer link only correlations results.

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	-0,09	0,04	0,28	0,60	0,04	0,04	0,07
c (LO)	-	0,10	-0,13	-0,17	0,08	0,07	0,07
h (LO)	-	-	-0,35	-0,03	0,73	0,83	0,90
ent (LO)	-	-	-	0,27	-0,33	-0,39	-0,19
betw (LO)	-	-	-	-	-0,02	-0,03	0,00
alpha (LO)	-	-	-	-	-	0,89	0,64
beta (LO)	-	-	-	-	-	-	0,73

Table 2.11 : Cora link only correlations results.

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	-0,14	-0,04	0,22	0,87	-0,01	-0,02	0,02
c (LO)	-	0,21	-0,26	-0,16	0,16	0,21	0,13
h (LO)	-	-	-0,65	-0,08	0,67	0,78	0,82
ent (LO)	-	-	-	0,21	-0,51	-0,62	-0,38
betw (LO)	-	-	-	-	-0,02	-0,05	-0,01
alpha (LO)	-	-	-	-	-	0,83	0,55
beta (LO)	-	-	-	-	-	-	0,62

Table 2.12 : WebKb link only correlations results.

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	-0,10	0,01	0,25	0,96	0,00	-0,00	0,05
c (LO)	-	0,15	0,23	-0,11	-0,04	0,07	-0,08
h (LO)	-	-	0,15	-0,00	-0,25	0,68	-0,39
ent (LO)	-	-	-	0,14	-0,13	0,01	-0,06
betw (LO)	-	-	-	-	0,00	0,00	0,05
alpha (LO)	-	-	-	-	-	-0,35	0,24
beta (LO)	-	-	-	-	-	-	-0,23

Table 2.13 : HepTh link only correlations results.

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	-0,16	0,05	0,24	0,49	0,13	0,07	0,15
c (LO)	-	0,13	-0,24	-0,30	0,06	0,12	0,02
h (LO)	-	-	-0,48	-0,09	0,52	0,75	0,73
ent (LO)	-	-	-	0,33	-0,33	-0,49	-0,20
betw (LO)	-	-	-	-	0,01	-0,07	0,03
alpha (LO)	-	-	-	-	-	0,71	0,53
beta (LO)	-	-	-	-	-	-	0,53

Table 2.14 : Synthetic datasets meta correlations (LO) results (ms=0).

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	0,10 ± 0,02	0,38 ± 0,01	-0,35 ± 0,01	0,89 ± 0,00	0,24 ± 0,01	0,39 ± 0,01	0,23 ± 0,01
c (LO)	-	0,18 ± 0,01	-0,22 ± 0,01	-0,10 ± 0,01	0,14 ± 0,01	0,22 ± 0,01	0,09 ± 0,01
h (LO)	-	-	-0,69 ± 0,01	0,21 ± 0,01	0,43 ± 0,01	0,64 ± 0,00	0,77 ± 0,00
e (LO)	-	-	-	-0,13 ± 0,01	-0,48 ± 0,01	-0,66 ± 0,00	-0,30 ± 0,01
b (LO)	-	-	-	-	0,12 ± 0,01	0,21 ± 0,01	0,15 ± 0,01
alpha (LO)	-	-	-	-	-	0,78 ± 0,00	0,24 ± 0,02
beta (LO)	-	-	-	-	-	-	0,39 ± 0,01

Table 2.15 : Synthetic datasets meta correlations (LO) results (ms=16).

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	0,10 ± 0,01	0,41 ± 0,01	-0,40 ± 0,01	0,88 ± 0,00	0,26 ± 0,01	0,43 ± 0,01	0,24 ± 0,01
c (LO)	-	0,18 ± 0,01	-0,25 ± 0,01	-0,11 ± 0,01	0,16 ± 0,01	0,24 ± 0,01	0,08 ± 0,01
h (LO)	-	-	-0,73 ± 0,01	0,22 ± 0,01	0,44 ± 0,01	0,67 ± 0,01	0,75 ± 0,00
e (LO)	-	-	-	-0,15 ± 0,01	-0,46 ± 0,01	-0,67 ± 0,01	-0,31 ± 0,01
b (LO)	-	-	-	-	0,13 ± 0,01	0,22 ± 0,01	0,15 ± 0,01
alpha (LO)	-	-	-	-	-	0,77 ± 0,01	0,25 ± 0,02
beta (LO)	-	-	-	-	-	-	0,40 ± 0,01

Table 2.16 : Synthetic datasets meta correlations (LO) results (ms=32).

Props	c (LO)	h (LO)	ent (LO)	betw (LO)	alpha (LO)	beta (LO)	accuracy (LO)
k (LO)	0,12 ± 0,02	0,41 ± 0,01	-0,40 ± 0,02	0,88 ± 0,00	0,23 ± 0,01	0,44 ± 0,01	0,22 ± 0,01
c (LO)	-	0,24 ± 0,01	-0,27 ± 0,01	-0,10 ± 0,02	0,14 ± 0,01	0,27 ± 0,01	0,10 ± 0,01
h (LO)	-	-	-0,82 ± 0,01	0,21 ± 0,01	0,40 ± 0,01	0,68 ± 0,01	0,69 ± 0,01
e (LO)	-	-	-	-0,15 ± 0,01	-0,40 ± 0,01	-0,67 ± 0,01	-0,32 ± 0,01
b (LO)	-	-	-	-	0,12 ± 0,01	0,22 ± 0,01	0,14 ± 0,01
alpha (LO)	-	-	-	-	-	0,69 ± 0,01	0,19 ± 0,02
beta (LO)	-	-	-	-	-	-	0,36 ± 0,01

2.5 Detailed Analysis of Graph Properties

2.5.1 Degree distribution

Degree distributions for the content and link graphs of Citeseer, Cora, WebKb, HepTh and Synthetic datasets are given in figure 2.4. The corresponding semi-log and log-log plots are given in figures 2.5 and 2.6. According to these figures, Citeseer, Cora, WebKb, HepTh datasets' both content and link graphs and Synthetic datasets' content graphs are exponential graphs (M. E. Newman 2000). Link graphs of the Synthetic datasets are random graphs.

The reason why content graphs of the Synthetic datasets are random graphs, could be because cosine similarity was used for the generation of the content graphs while in order to produce the Synthetic datasets match similarity (Cataltepe et al. 2011) was used.

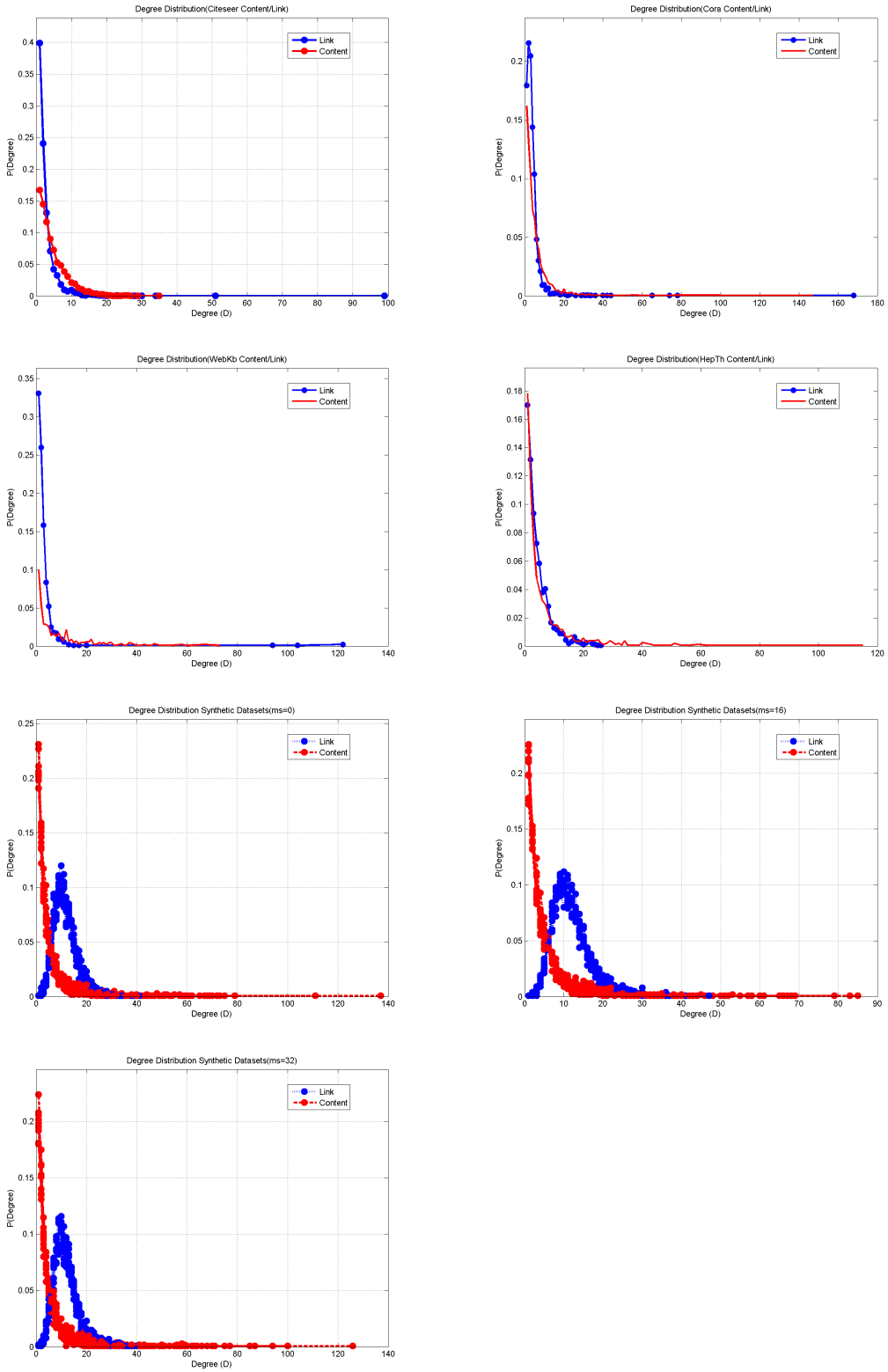


Figure 2.4 : Degree distributions (linear plots).

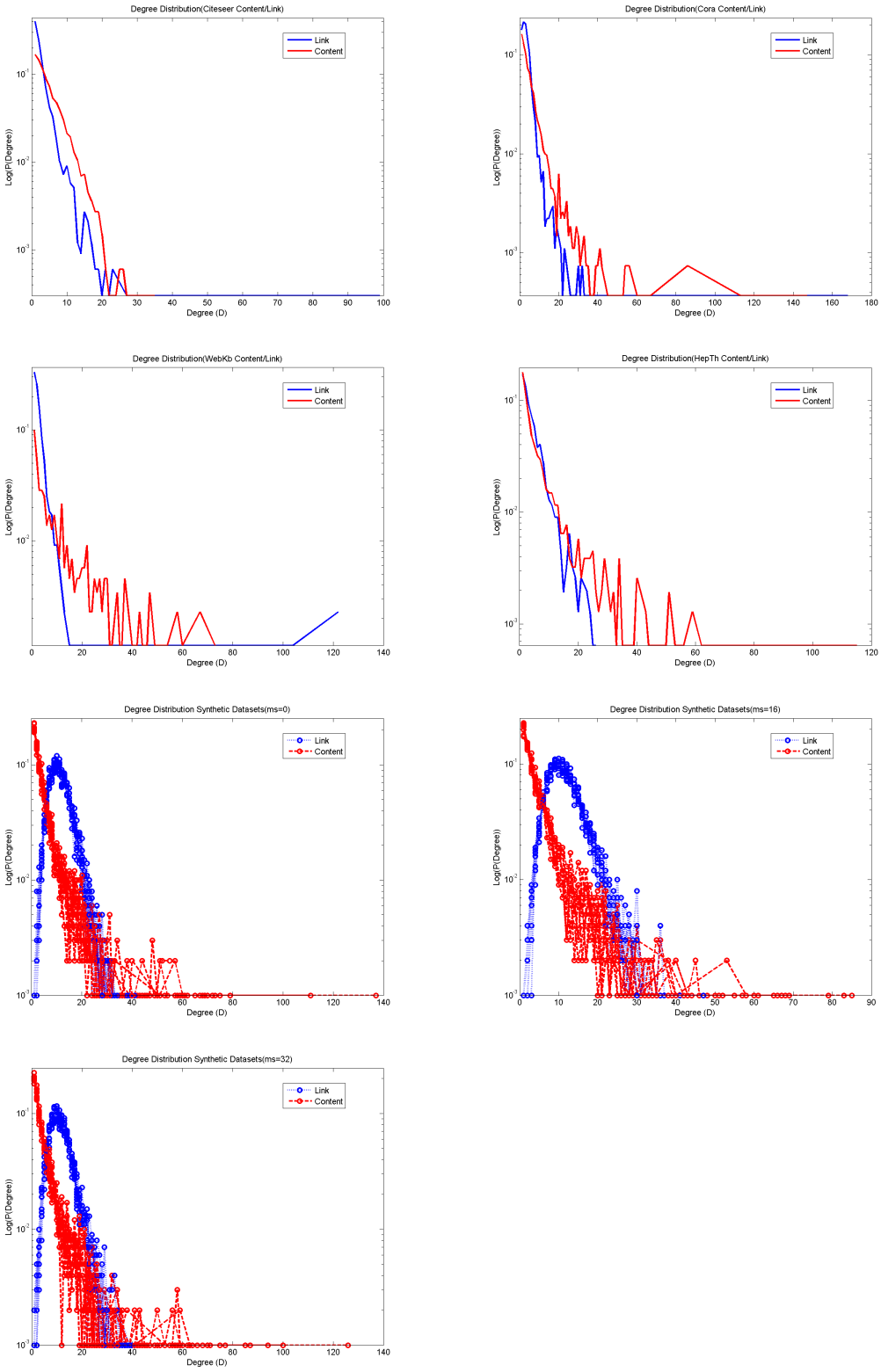


Figure 2.5 : Degree distributions (semiLog plots).

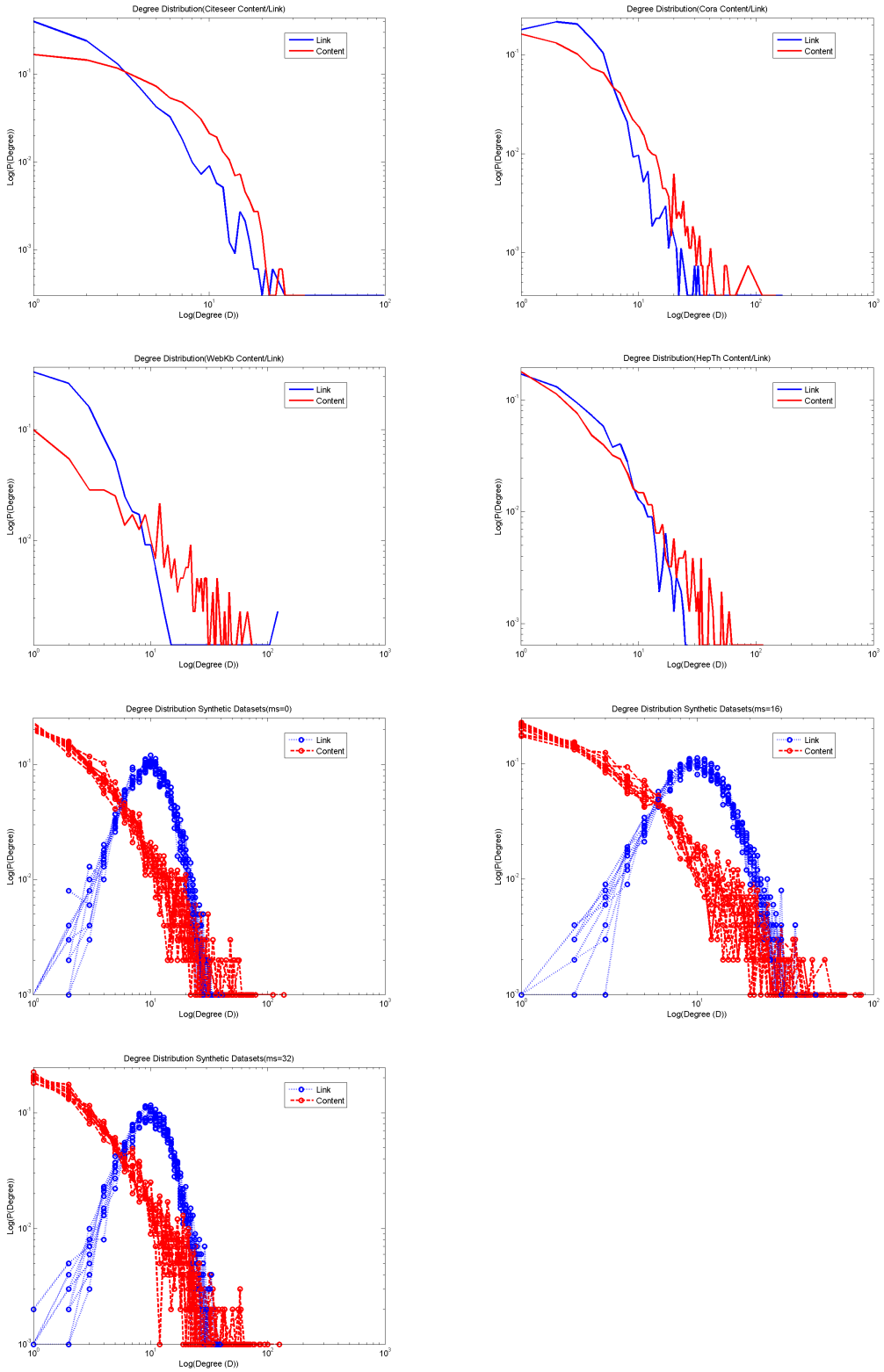


Figure 2.6 : Degree distributions (logLog plots).

2.5.2 Homophily versus degree

Homophily versus degree graphs for the datasets are calculated and given in Figures 2.7 and 2.8. X axis of the graphs are given in log-scale to make the graphs more clear for low degrees in figure 2.7. According to the figures, homophily has a high variance on nodes with higher degree for Cora and Citeseer datasets, while it has a low variance on Synthetic datasets. For Cora dataset, homophily decreases slowly as degree increases for both content and link graphs. For WebKb dataset, homophily is low for low degree and there are some hubs whose homophily is around 0.7. For higher degrees of content graph, homophily has a high variance and not possible to trust. For HepTh dataset, homophily is more stable for lower degrees and have a value around 0.8. Homophily for content graph behaves similar to link graph. There are some hubs and homophily is quite low.

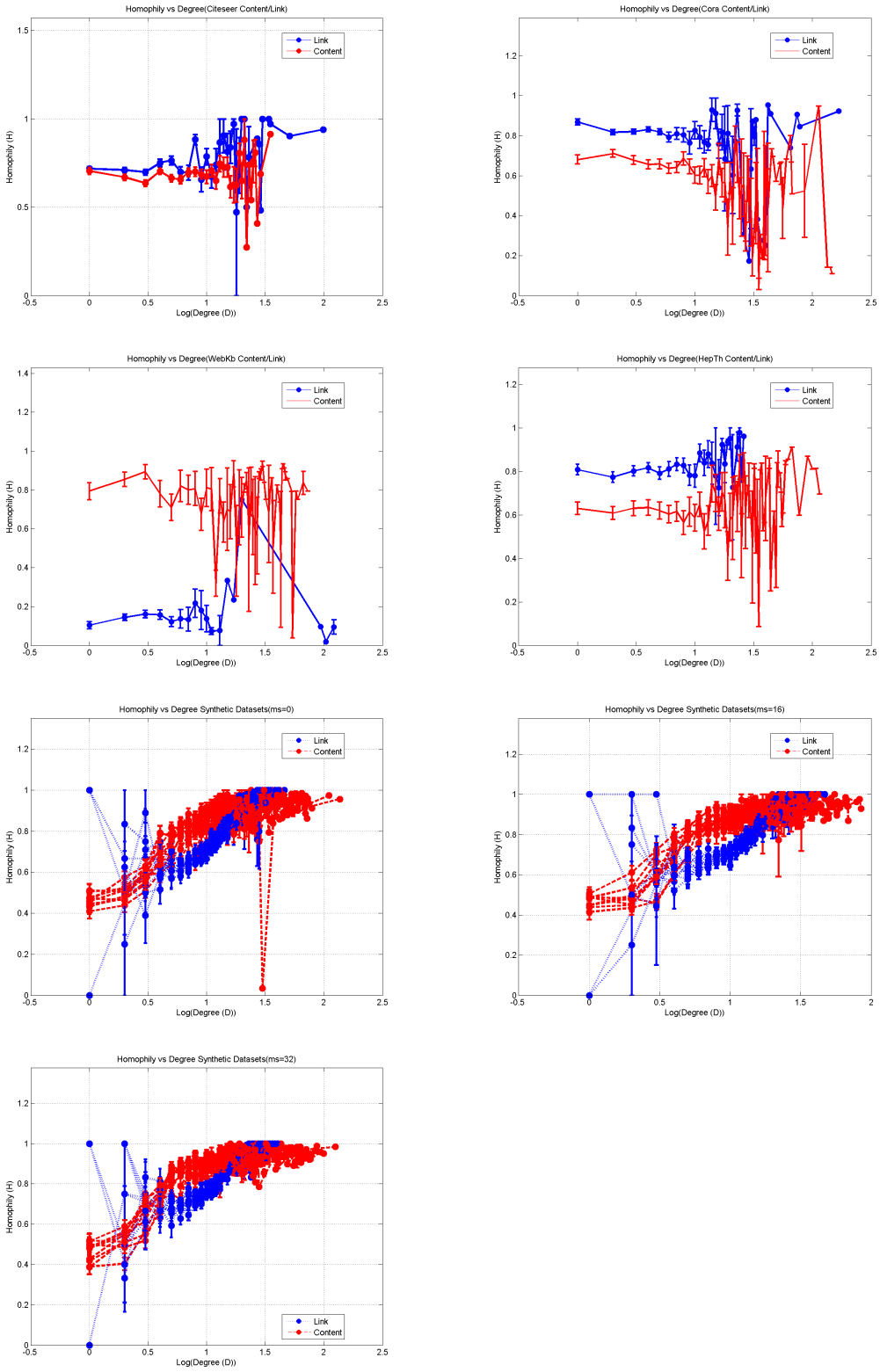


Figure 2.7 : Homophily vs degree.

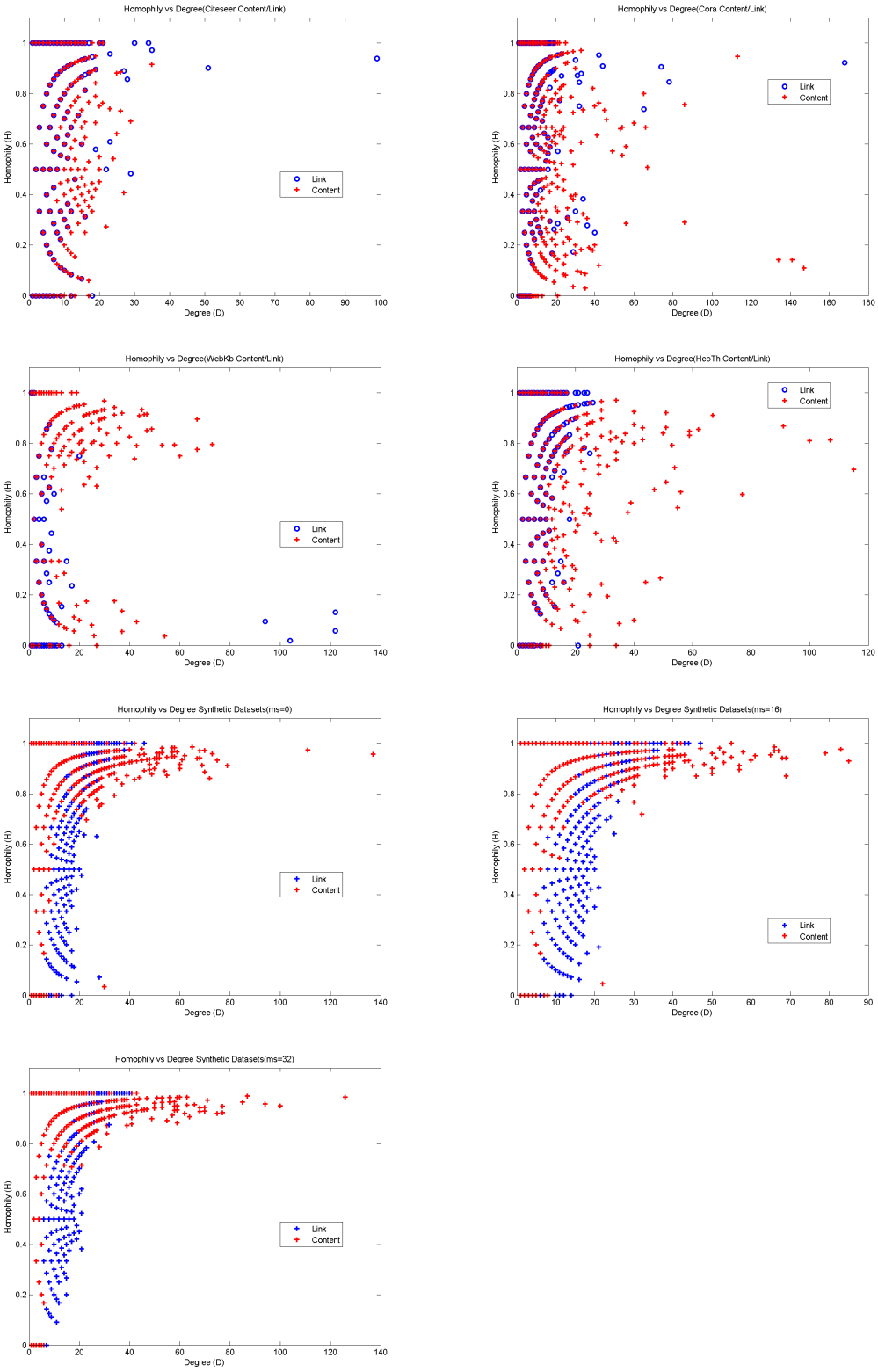


Figure 2.8 : Homophily vs degree dot plot.

2.5.3 Entropy versus degree

Entropy versus degree graphs for the datasets are calculated and given in Figures 2.7 and 1. X axis of the graphs are given in log-scale to make the graphs more clear for low degrees in figure 2.7. According to the figures, for higher degrees entropy has a high variance. For Citeseer, Cora, WebKb and HepTh datasets, entropy increases by the degree for lower degrees. For HepTh dataset, after some threshold degree, it starts decreasing. For Synthetic datasets entropy first increases and after some threshold degree level, it decreases.

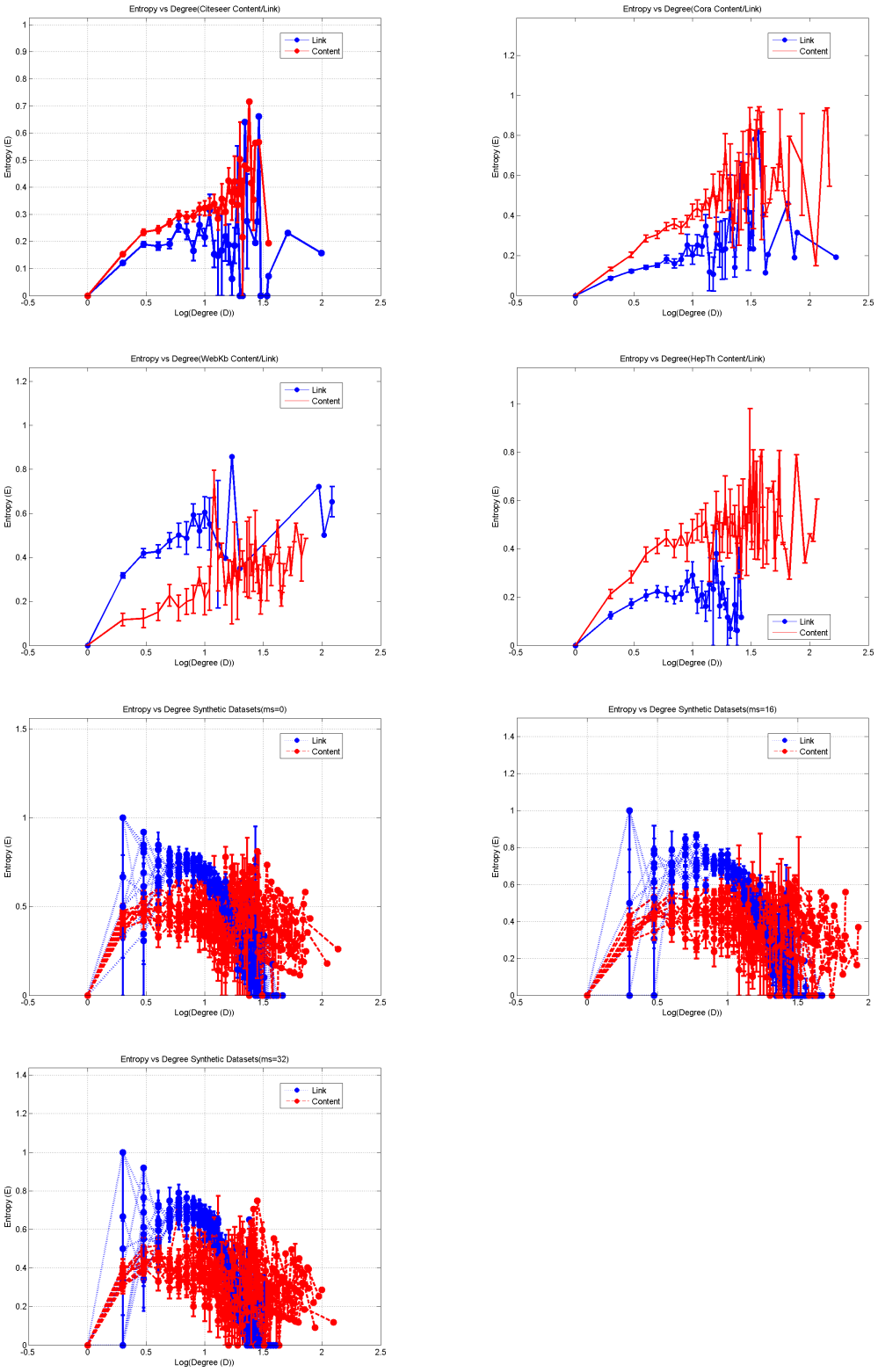


Figure 2.9 : Entropy vs degree.

2.5.4 Clustering coefficient versus degree

Clustering coefficient versus degree graphs for the datasets are calculated and given in Figure 2.10. According to the figure, for Cora and Citeseer datasets, clustering coefficient decreases as degree increases and generally clustering coefficient of the content graph is higher than the link graph's. For WebKb and HepTh datasets, the behavior in the link graphs are similar to Cora and Citeseer datasets', but on WebKb with lower clustering coefficient values and on HepTh with higher clustering coefficient values. When we consider the content graph of WebKb dataset, clustering coefficient values are much higher than link graph. For HepTh dataset, clustering coefficient values of the content graph are lower than the link graph. The behavior on Synthetic datasets is totally different. Clustering coefficient values for their content graphs are close to their link graphs'.

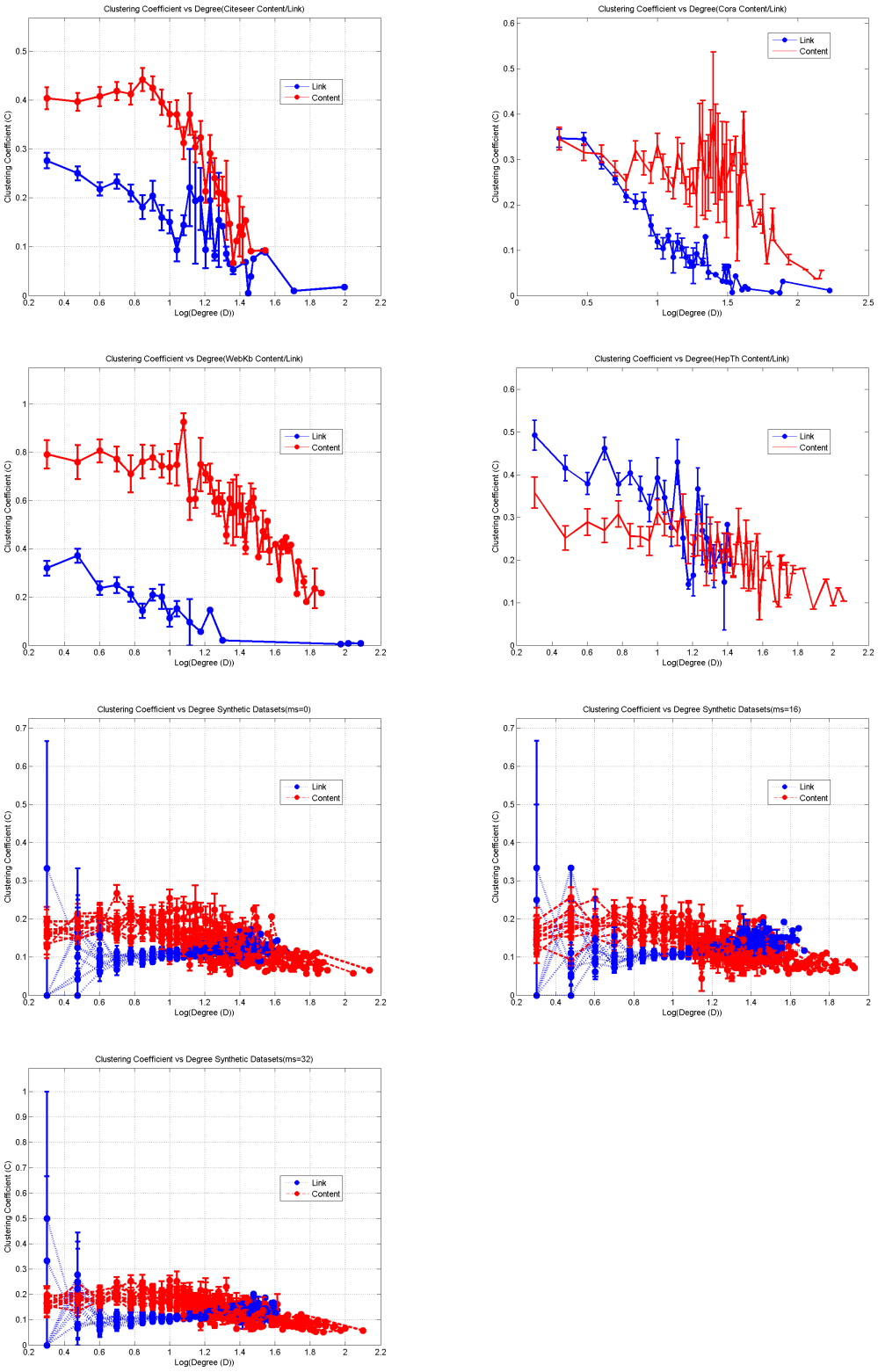


Figure 2.10 : Clustering coefficient vs degree.

2.5.5 Homophily versus clustering coefficient

Homophily versus clustering coefficient graphs for the datasets are calculated and given in Figure 2. According to the figure, Citeseer, Cora and Hepth datasets show almost the same behavior. Homophily and clustering coefficient are not correlated on these datasets. Link graph of the WebKb has much lower values than its content graph. It is totally different on synthetic datasets.

2.5.6 Local alpha versus degree

Local alpha versus degree graphs for the datasets are calculated and given in Figure 2.11. X axis of the graphs are given in log-scale to make the graphs more clear for low degrees in Figure 2.11. According to the figure, local alpha is not correlated with the degree for the content graph. For higher degrees on the link graph, high variance in local alpha is observed.

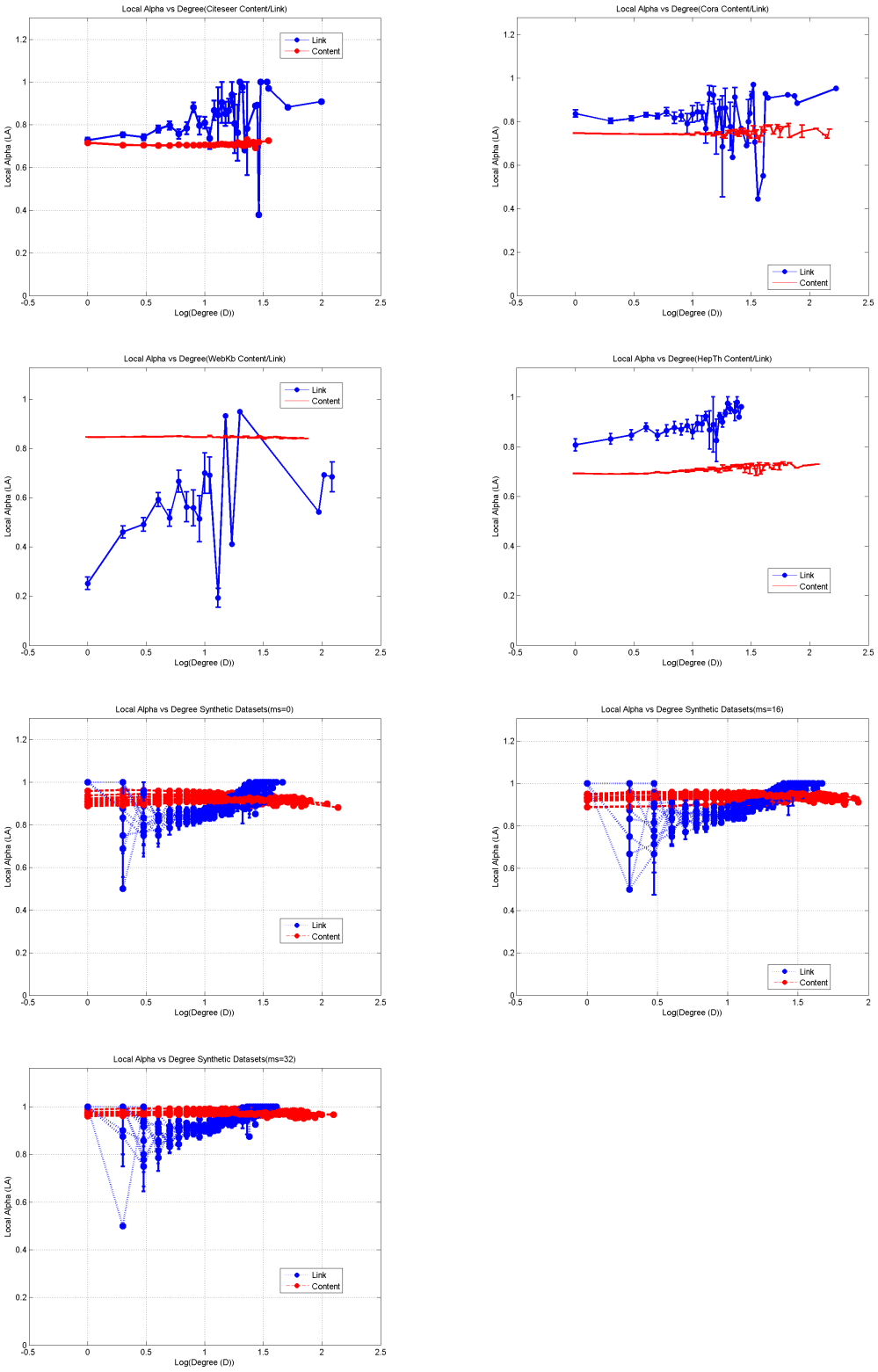


Figure 2.11 : Local alpha vs degree.

2.5.7 Local alpha versus homophily

Local alpha versus homophily graphs for the datasets are calculated and given in Figures 2.12 and 4. According to the figures, for Cora, Citeseer and HepTh datasets, local alpha increases as homophily increases on the link graphs. For WebKb dataset, there is no correlation between homophily and local alpha. Local alpha is not correlated with homophily on the content graph for these datasets. For Synthetic datasets' link graphs, variance in local alpha decreases as homophily increases. Again, local alpha is not correlated with homophily on content graphs.

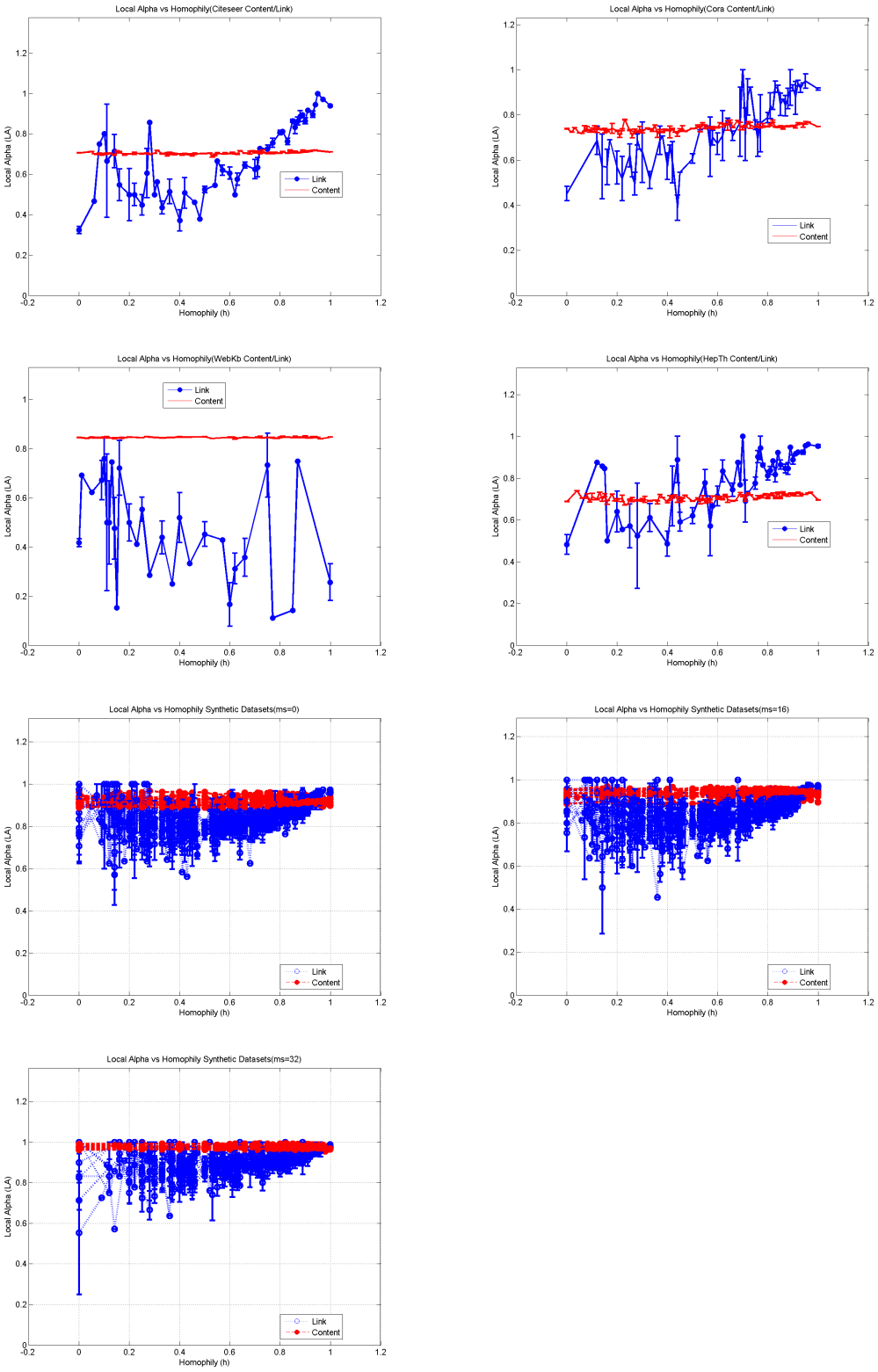


Figure 2.12 : Local alpha vs homophily.

2.5.8 Local beta versus degree

Local beta versus degree graphs for the datasets are calculated and given in Figure 2.13. X axis of the graphs are given in log-scale to make the graphs more clear for low degrees in figure 2.13. According to the figure, local beta has a high variance on nodes with higher degree for Cora, Citeseer, HepTh and WebKb datasets, like in homophily degree graphs, while it is more stable on Synthetic datasets. For Synthetic datasets, local beta increases as degree increases on the link graph. Almost the same behavior is observed on the content graphs.

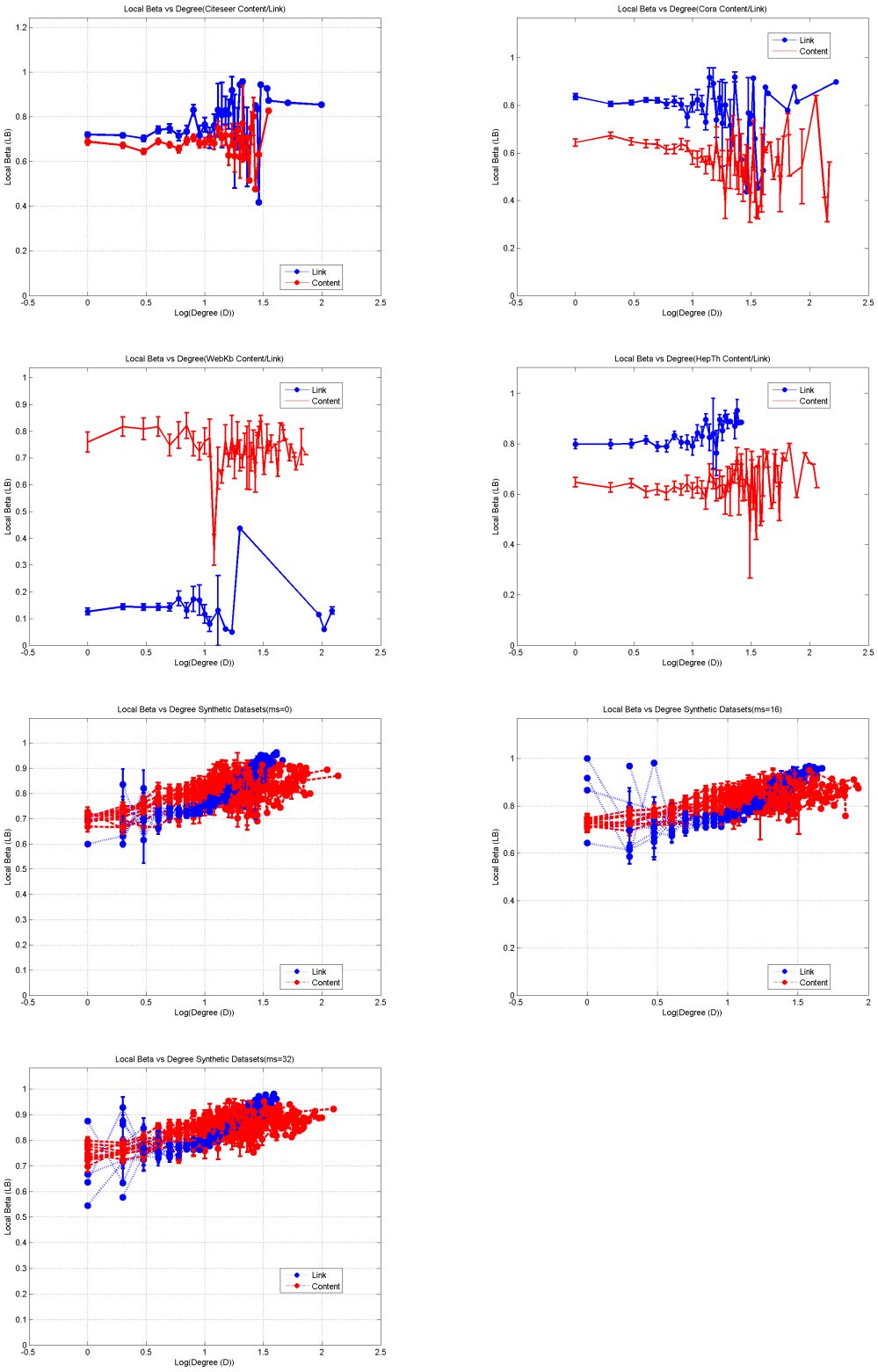


Figure 2.13 : Local beta vs degree.

2.5.9 Local beta versus homophily

Local Beta versus homophily graphs for the datasets are calculated and given in Figures 2.14 and 5. According to the figures, local beta is directly correlated with homophily on the content graphs for all datasets. For all datasets, the behavior of content graph show us that the nodes having high homophily tend to connect to the nodes with high homophily friendships. Similar to behavior on the content graphs, local beta also increases as homophily increases on all datasets' link graphs.

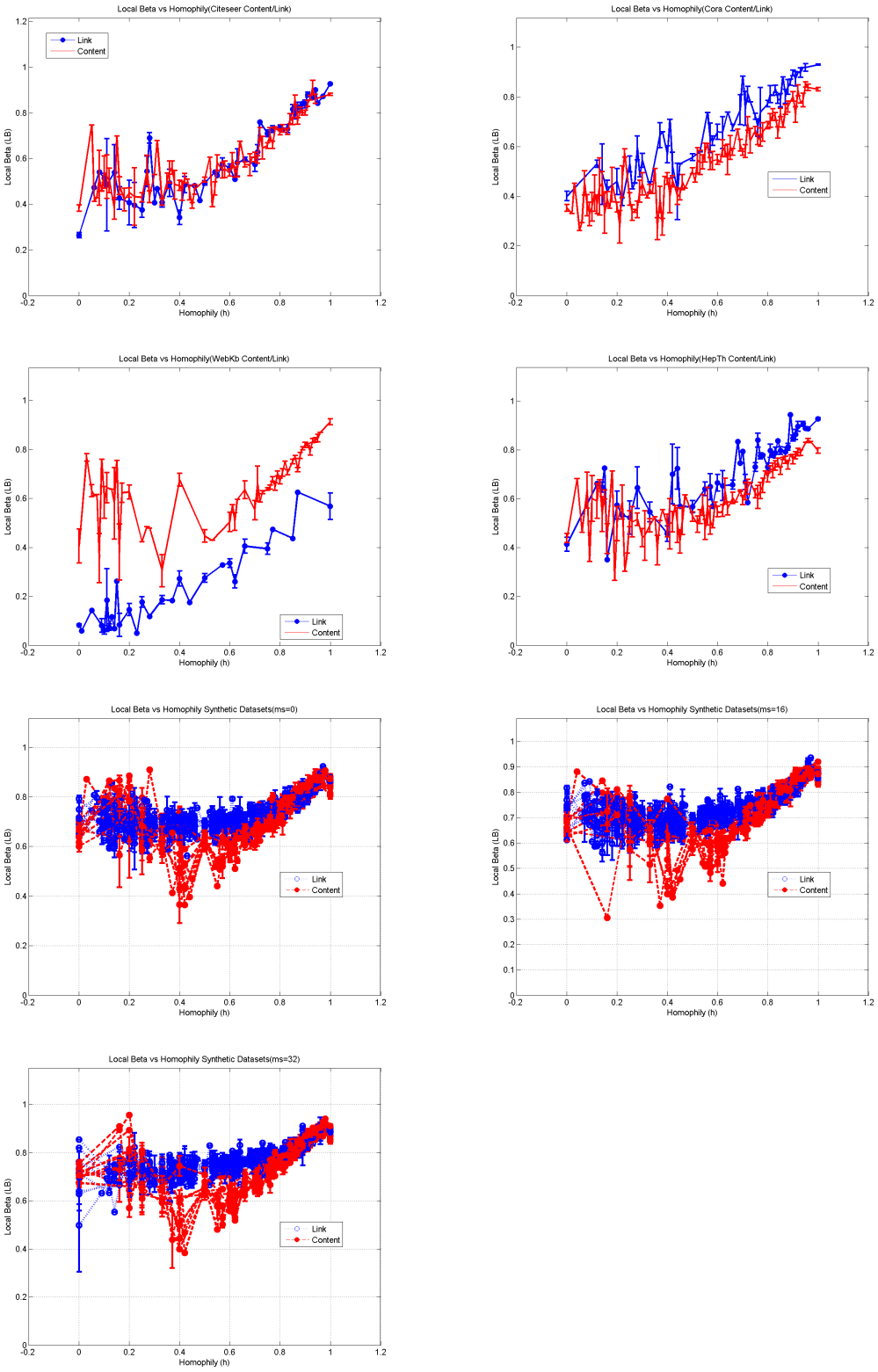


Figure 2.14 : Local beta vs homophily.

2.5.10 Local alpha versus clustering coefficient

Local alpha versus clustering coefficient graphs for the datasets are calculated and given in Figures 6 and 7. According to the figures, there is no correlation between local alpha and clustering coefficient on both content and link graphs for all datasets.

2.5.11 Local beta versus clustering coefficient

Local beta versus clustering coefficient graphs for the datasets are calculated and given in Figures 8 and 9. According to the figures, there is no correlation between local beta and clustering coefficient on both content and link graphs for all datasets. Local beta values are higher on the link graph except for WebKb dataset.

2.5.12 Accuracy versus degree

Accuracy versus degree graphs for the datasets are calculated and given in Figures 2.15 and 10. X axis of the graphs are given in log-scale to make the graphs more clear for low degrees in figure 2.15. According to the figures, high variance in accuracy is observed on the nodes with higher degrees. For the Synthetic datasets, high variance is observed on the lower degrees and accuracy increases as the degree increases.

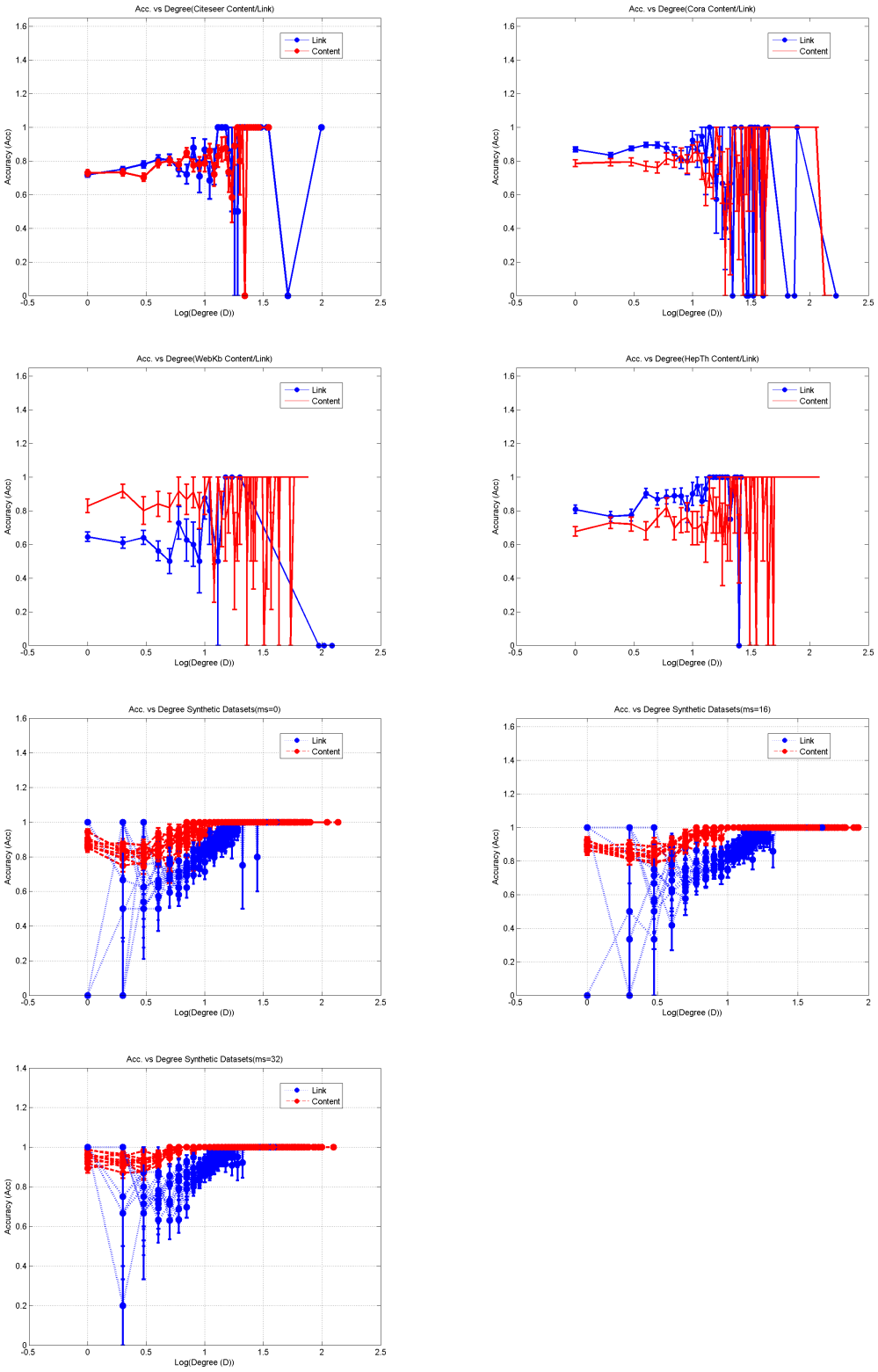


Figure 2.15 : Accuracy vs degree.

2.5.13 Accuracy versus clustering coefficient

Accuracy versus clustering coefficient graphs for the datasets are calculated and given in Figures 11 and 12. According to the figures, there is no correlation between accuracy and clustering coefficient.

2.5.14 Accuracy versus homophily

Accuracy versus homophily graphs for the datasets are calculated and given in Figures 2.16 and 13. According to the figures, accuracy is highly correlated with homophily on the link graphs. On content graphs, accuracy has a high variance for low homophily nodes.

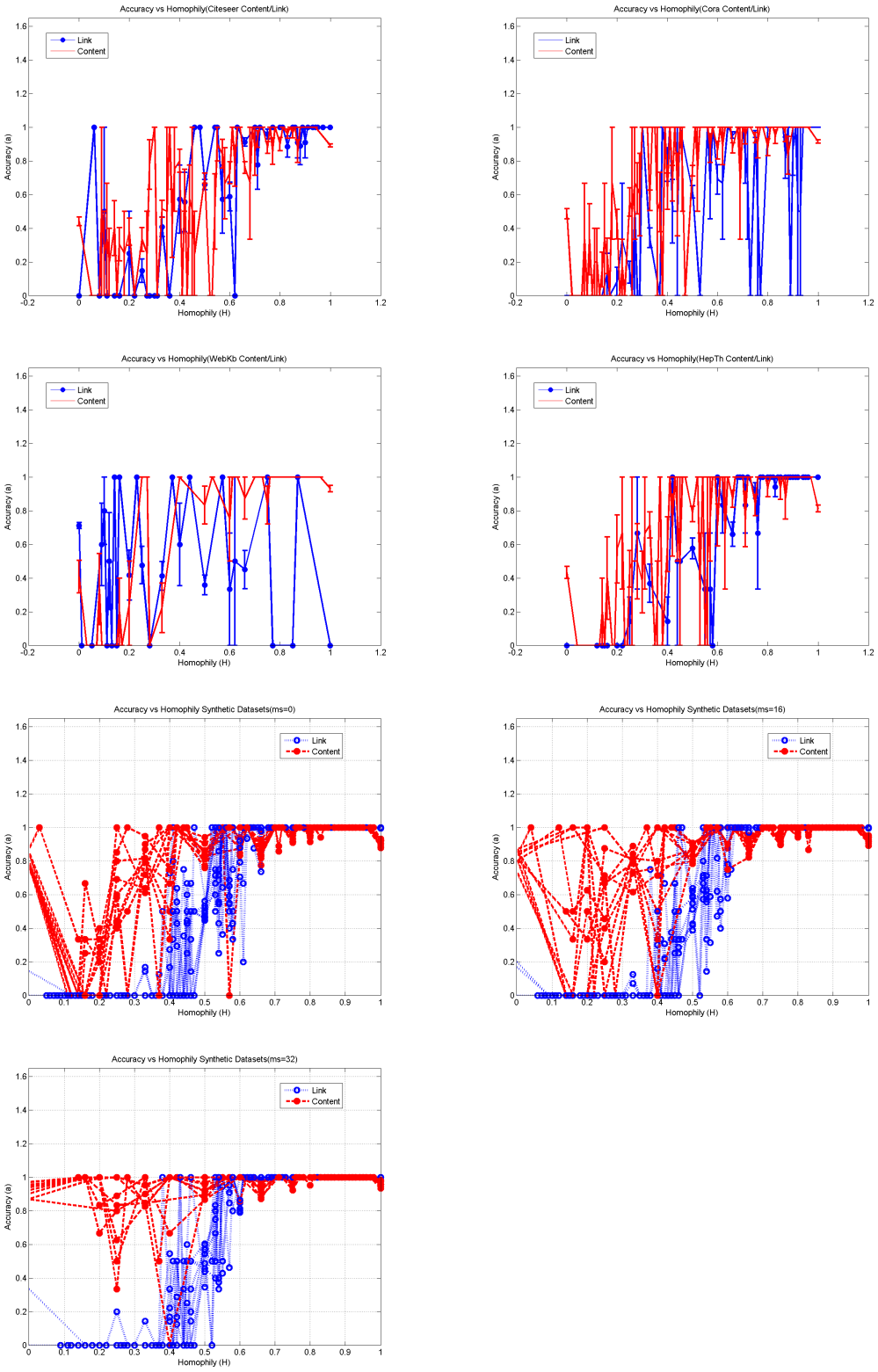


Figure 2.16 : Accuracy vs homophily.

2.5.15 Accuracy versus entropy

Accuracy versus entropy graphs for the datasets are calculated and given in Figures 2.17 and 14. According to the figures, for content graphs, variance of the accuracy increases by entropy. The same behavior also applies to the link graphs of Cora and Citeseer datasets. For the WebKb, HepTh and Synthetic datasets, accuracy is not correlated with entropy on their link graphs.

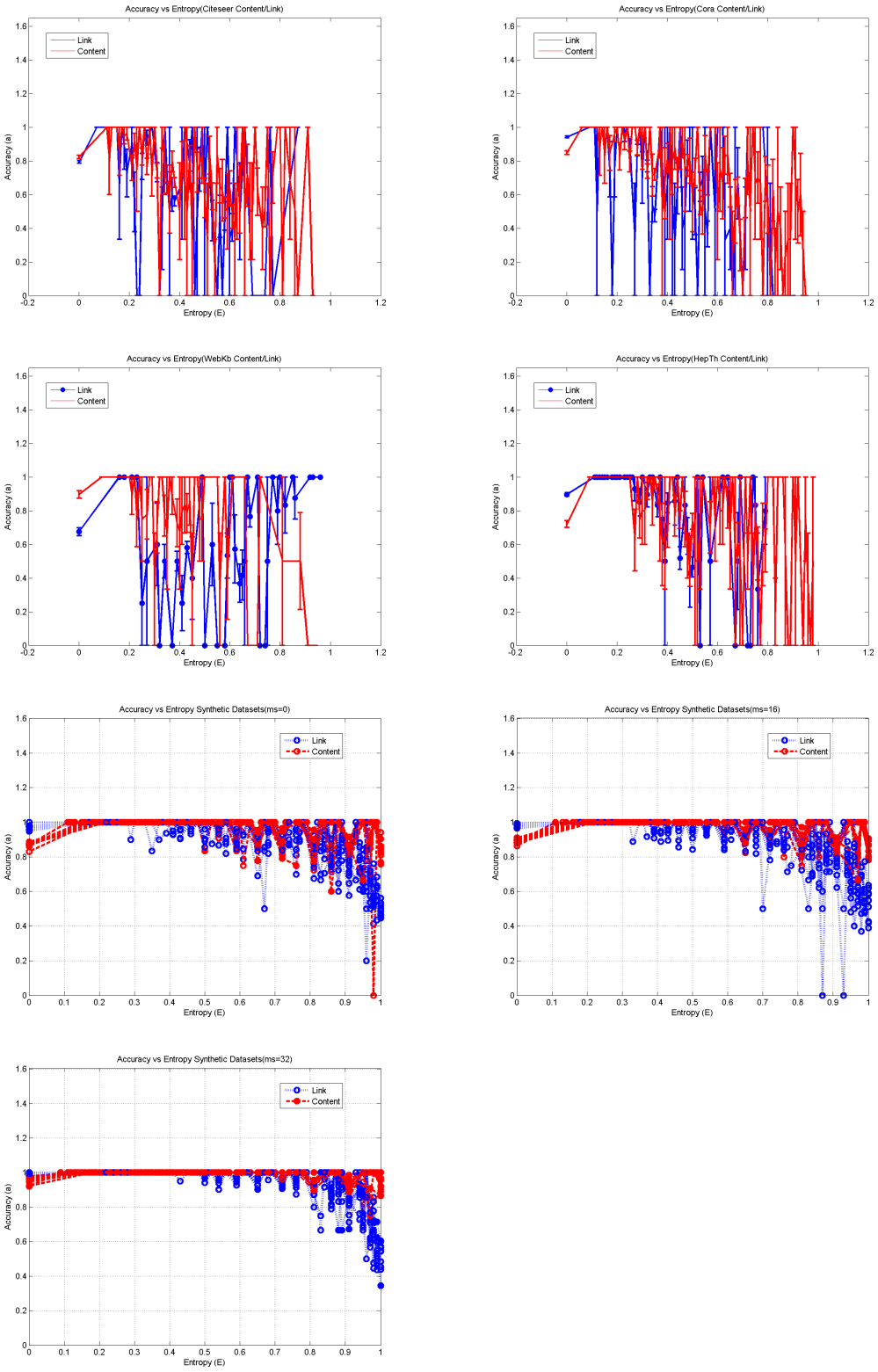


Figure 2.17 : Accuracy vs entropy.

2.5.16 Accuracy versus local alpha

Accuracy versus local alpha graphs for the datasets are calculated and given in Figures 2.18 and 15. According to the figures, for the content graphs local alpha of the nodes are too populated between 0.8 and 1.0 which is totally different from link graph. Accuracy and local alpha are highly correlated.

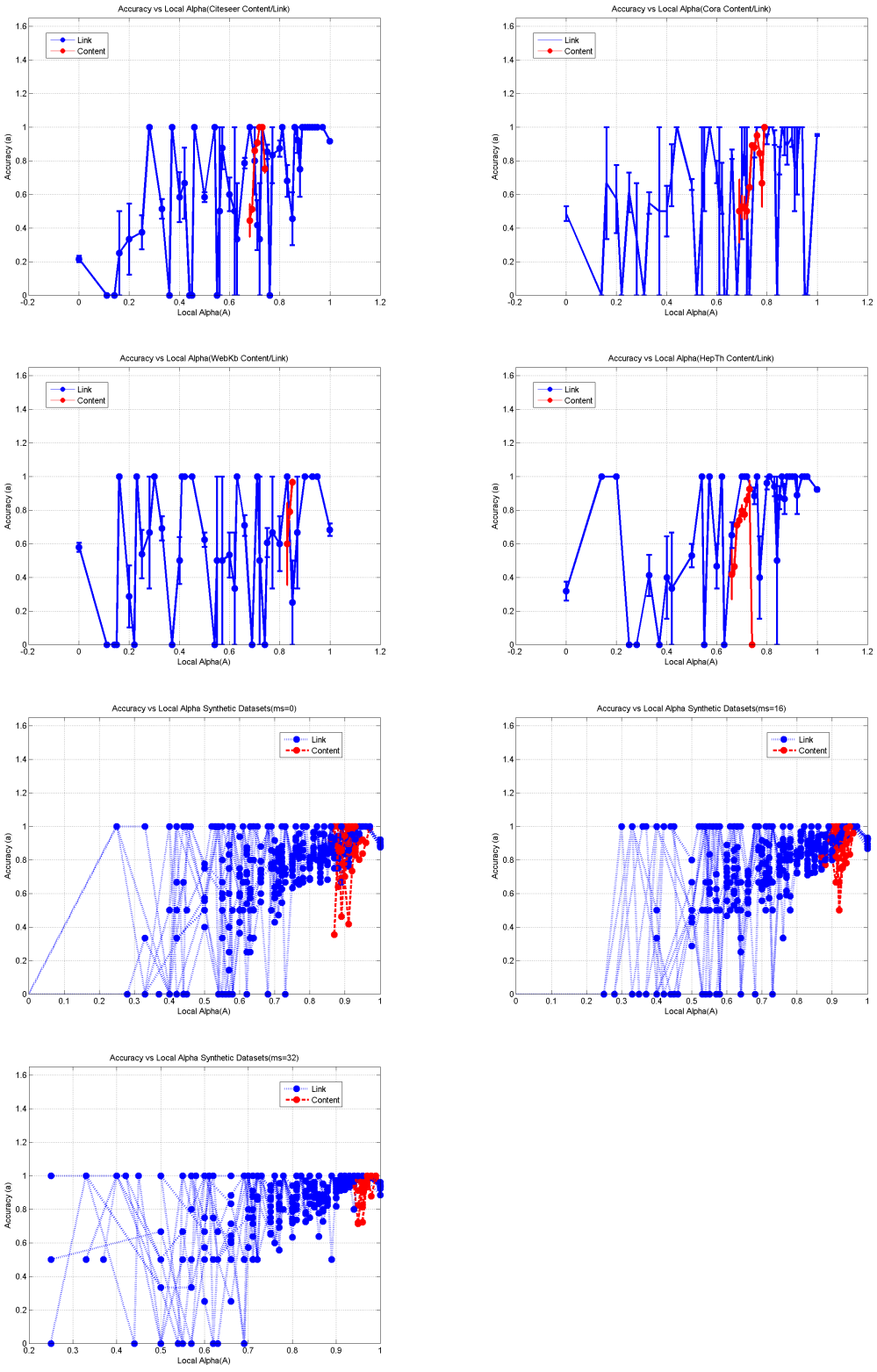


Figure 2.18 : Accuracy vs local alpha.

2.5.17 Accuracy versus local beta

Accuracy versus local beta graphs for the datasets are calculated and given in Figures 2.19 and 16. According to the figures, accuracy increases as local beta increases for both on content and link graphs. On Synthetic datasets, high variance in accuracy is observed on nodes with low degree.

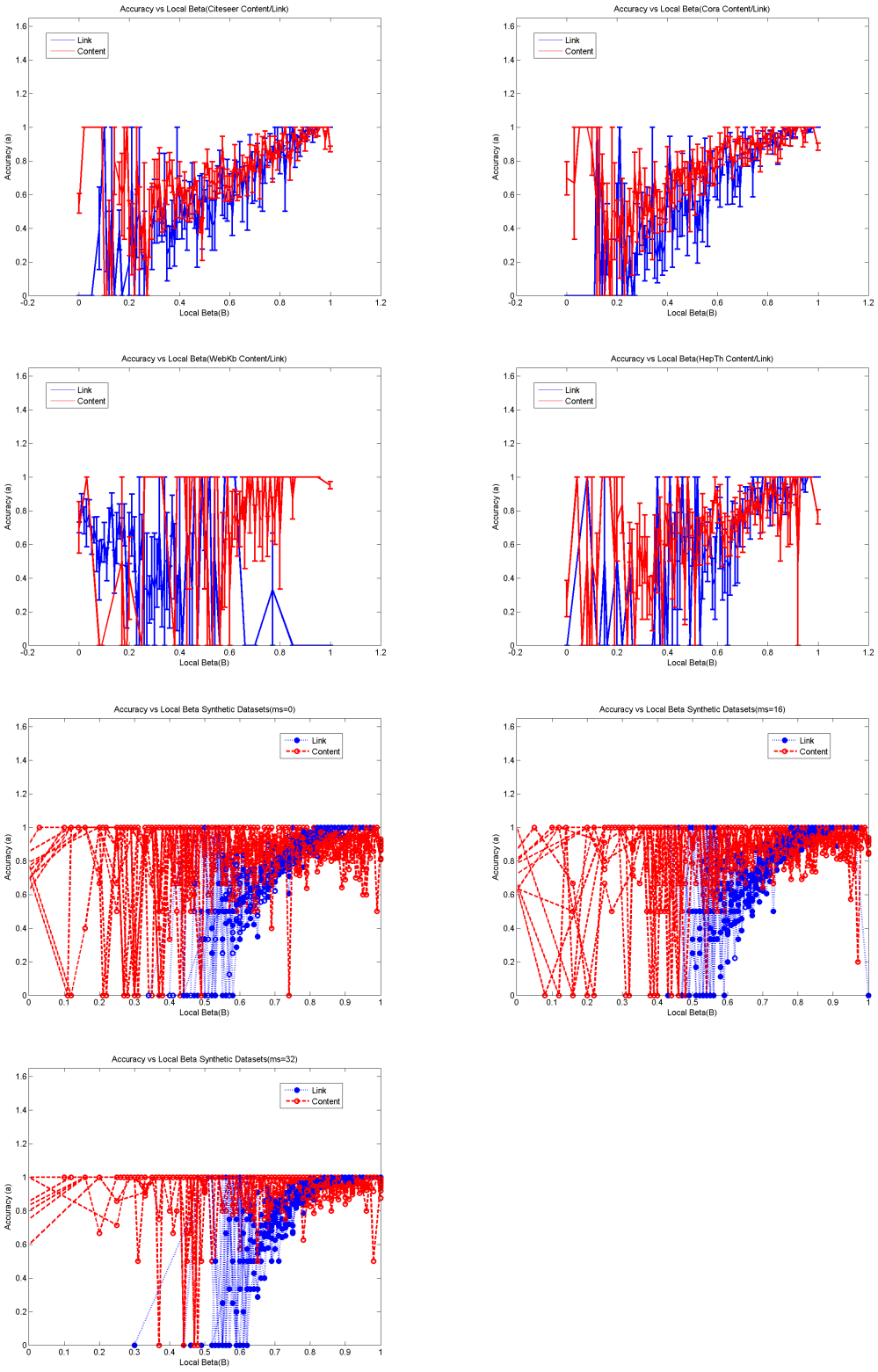


Figure 2.19 : Accuracy vs local beta.

2.6 Discussion

Graph properties' plots and calculated correlations show us that homophily and accuracy are highly correlated as expected. Homophily and entropy are negatively correlated as expected. Local alpha and accuracy are highly correlated on some datasets. Local alpha and local beta are positively highly correlated for the link graphs. Local beta and homophily are also correlated as expected.

When homophily is low, as in the case of WebKb dataset, links are not useful and thus link only classifier just using links can not achieve a good test accuracy. This situation also affects the test accuracy performance of collective classification. During iterations, if homophily is low, flow of information through the network in collective classification is not as beneficial as in the high homophily case.

Test accuracy obtained with snowball sampling is generally lower than test accuracy obtained using random sampling on high homophily datasets. On the other hand on low homophily datasets, the sampling method used does not make a big difference on test accuracy.

In Section 3, we will use local alpha and local beta for deriving new heterogeneous classifier combination methods.

We have seen that the graph produced from content features has different characteristics than the link graph. The homophily of the content graph is generally too low, compared to link graph. Since the content graph is constructed using local content features and since there are thousands of content features most of which are zero, instead of using all content features for construction of content graph, first a feature selection algorithm can be applied and most useful features may be determined. Then, the content graph may be constructed using only these selected features.

3. HETEROGENEOUS CLASSIFIERS FOR COLLECTIVE CLASSIFICATION

3.1 Background and Purpose

Collective classification (Chakrabarti et al. 1998, Macskassy & Provost 2007, Sen et al. 2008) algorithms aim to classify networked data when the test nodes and their links to other test nodes and training nodes are known. In collective classification, first a base classifier is trained using both content and link information in training data. Then, using a collective inference method, test nodes are iteratively labeled, based on their content and neighbor information. Especially when there is class autocorrelation among the neighboring nodes in the network, test nodes are able to take advantage of their neighbors' class information and collective classification improves classification accuracy (Jensen et al. 2004). Iterative Classification Algorithm (ICA), Gibbs Sampling and Relaxation Labeling (Macskassy & Provost 2007, Sen et al. 2008) are common methods of collective inference.

Different choices of base classifiers that are able to use content and neighbors' link information, such as naive Bayes, logistic regression, decision trees, k-nearest neighbors, have been used in the literature (Jensen et al. 2004, McDowell et al. 2009; 2007, Neville & Jensen 2000, Sen et al. 2008). The base classifier takes as input, usually, the content features of the node being classified and relational features, which are usually an aggregation of the class labels of the other linked instances (McDowell et al. 2009; 2007, Sen et al. 2008). However, when content and relational features show different characteristics it may not be optimal to have a single classifier to combine all of those features. For example, when the homophily in the network is low, one may want to give more weight to the information that the content has. On the other hand, if homophily is high, the content information may need to be given less weight. Also, if there are multiple content types, such as text, images and audio on a web page or link types such as direct or co-citation links on a web page; citation or bibliography links

on scientific papers; SMS or call links in call detailed record (CDR) data or family, work, friend links on a social web site, it is hard to input all those features into a single classifier while still obtaining a good generalization performance. Certain content or link types may be better suited to identify certain classes and putting all of them into a single feature vector would harm discrimination ability. Moreover, for certain types of content, certain choices of local classifier may be known to perform better, for example SVMs have been known to perform well for text categorization (Joachims 1998), while hidden Markov models are used for speech (Rabiner 1989), and therefore it could be better to use different choices of classifiers for each content type. When a different classifier is taught each type of content/link information, these classifier outputs need to be combined to come up with a final class prediction.

In this section, we consider the following factors that affect the performance of a classifier for a networked classification problem:

- **Identity of the dataset:** If a dataset has high homophily, and high average node degree, then considering network information helps with classification.
- **Snowball vs random sampling:** Based on the classification problem, the test data may either be randomly distributed or may come in a connected chunk. Random or snowball sampling, respectively, results should be used for these two cases. Snowball sampling accuracy results are usually lower and have more variance over different test sets.
- **Proportion of test data over the whole data set:** If the test data set size increases over the available nodes (and hence the training set size decreases), the test classification accuracy usually decreases. In addition to the decreasing number of training instances, the increased instability of ICA/LO classification for a larger test set could be the reason for this.
- **Identity of the training and test instances:** While a classifier may perform well on a test set, it may perform poorly on another test set.

First of all, for a given training and test split of the dataset, we train different classifiers, such as SVM, Naive Bayes, Bayes Network, on content only, link only and content and

link information (CO, LO, COLO respectively). We propose seven different classifier combination methods to combine different classifiers. In order to select the most diverse and useful classifiers, we use a genetic algorithm based selection method. We use the validation sets to determine which classifiers should be combined to achieve better performance for the given train/test partition of the dataset. After we determine the classifier set to be combined, we use the same classifier set for the test set. Our experiments on four different datasets, namely Citeseer, Cora, WebKb and HepTh, show that our new method outperforms best of the base classifiers on the datasets used. Our method can also be extended to collective classification scenarios with multiple types of content and link.

3.2 Methodology

3.2.1 Classifier combination methods

We use seven different classifier combination methods. Some of these methods, such as average and maximum, are classifier combination methods that have frequently been used in the literature (Kuncheva 2004). We also introduce some classifier combination methods that are specific to classification of networked data and that use some of the network properties described in Section 2.

In this section, we consider different classifiers, such as SVM, Naive Bayes, Bayes Network, trained on content only (g_{CO}), link only (g_{LO}) or content and link (g_{COLO}). We assume that there are a total of K such classifiers being considered and denote the output of each one of these classifiers on a node u as $g_k(u), k = 1, \ldots, K$. If the classification problem consists of C classes, the classifier outputs $g_k(u)$ are C dimensional vectors. For each dimension c, $g_{k,c}(u)$ contains a number in $[0:1]$ which indicates the posterior probability of the cth class for the particular node. We use $h_c()$ to denote a particular classifier combination method. For a particular node u, given the base classifiers $g_1, \ldots, g_K$, the classifier combination method h_c returns a label, denoted as $\widehat{r(u)} = h_{comb}(g_1(u), \ldots, g_K(u))$. If the classification problem consists of C classes, then for the node u, the classifier combination output $\widehat{r(u)}$ is also a C dimensional

vector. We denote the cth dimension of the combined output as $\widehat{\mathbf{r}(u)_c}$. The predicted class is taken to be the class corresponding to the maximum value of the $\widehat{\mathbf{r}(u)}$ vector.

- *AVE* (Average) method returns the average of the outputs of the classifier set on a node.

$$\widehat{\mathbf{r}(u)} = h_{AVE}(g_1(u), \ldots, g_K(u)) = \frac{1}{K}\sum_{k=1}^{K} g_k(u) \tag{3.1}$$

- *MAX* (Maximum): For each class c, the *MAX* method computes the probability of that class to be the maximum of the class probabilities produced by each classifier for class c.

$$\widehat{\mathbf{r}(u)_c} = h_{MAX}(g_1(u), \ldots, g_K(u)) = max_{k=1,\ldots,K}\, g_{k,c}(u) \tag{3.2}$$

- *ENT* (Minimum Entropy). When a classifier outputs similar probabilities for each class, then this indicates uncertainty of the classifier. We measure the uncertainty of classifier k at node u using $Ent(g_k(u))$ which is the entropy of the C dimensional classifier output $g_k(u)$. The Minimum Entropy method selects the classifier for which the classifier outputs have the minimum entropy:

$$\widehat{\mathbf{r}(\mathbf{u})} = g_{k*}(u) \quad \text{where} \quad k* = \arg\min_{k=1,\ldots,K} Ent(g_k(u)) \tag{3.3}$$

- *AVG − LA* (Alpha Weighted Average) method computes a weighted average of the classifier outputs. The weight of a classifier g_k on a particular node u is taken to be $\alpha_{g_k}(u) \in R$, which is the classifier's accuracy on the training nodes which are in the neighborhood of the node (See Section 2.2.10).

  If a test node is too far away from training data in G, there may not be any training nodes within its h neighborhood. The same could happen for a test node too far away from training data in the content graph G_{CO}. When that is the case, instead of the local weights $\alpha_{g_k}(u)$, their averages over the whole corresponding G_{CO} or G_{LO} graphs are used.

In order to compute the $\alpha_{g_k}(u)$ values, we need to determine the size of the neighborhood. In our previous study (Cataltepe et al. 2011), we determined the best values of the number of hops, h^*_{CO} and h^*_{LO}, based on the training data as follows: We compute the correlation (Pearson Correlation Coefficient, to be precise) between accuracy of a classifier on a node and also the local average accuracy of the classifier within the h neighborhood of the node. We choose the best number of hops, $h*$, as the number of hops which maximizes this correlation.

$$\widehat{\mathbf{r}(u)} = h_{AVG-LA}(g_1(u),\ldots,g_K(u)) = \frac{1}{K}\sum_{k=1}^{K}\alpha_{g_k}(u)g_k(u) \tag{3.4}$$

- $MAX-LA$ (Alpha Weighted Max) method takes the maximum of the classifier outputs weighted with the α_{g_k} values:

$$\widehat{\mathbf{r}(u)_c} = h_{MAX-LA}(g_1(u),\ldots,g_K(u)) = max_{k=1,\ldots,K}\ \alpha_{g_k}(u)g_{k,c}(u) \tag{3.5}$$

- $AVG-BETA$ (Beta Weighted Average) method also computes a weighted average of the classifier outputs for each class. The weights are determined as the $\beta_k(u)$ values computed for the node according to the graph used for the classifier g_k. As explained in Section 2.2.11, the local beta property is defined for each node and a graph (G_{CO} or G_{LO}) as the homophily of the neighbors of a node within the h-neighborhood and in the training set. For CO classifiers β_{CO}, for LO and ICA classifiers β_{LO} is used.

  If a test node is too far away from training data in G, there may not be any training nodes within its h neighborhood. When that is the case, instead of the local weights $\beta(u)$, global β values, which are namely their averages over the whole graph, are used.

$$\widehat{\mathbf{r}(u)} = h_{AVG-BETA}(g_1(u),\ldots,g_K(u)) = \frac{1}{K}\sum_{k=1}^{K}\beta_k(u)g_k(u) \tag{3.6}$$

- $MAX-BETA$ (Beta Weighted Max) method takes the maximum of the classifier outputs weighted with the $\beta(u)$ values:

$$\widehat{\mathbf{r}(u)_c} = h_{MAX-BETA}(g_1(u), \ldots, g_K(u)) = max_{k=1,\ldots,K}\ \beta_k(u) g_{k,c}(u) \tag{3.7}$$

3.2.2 Genetic algorithm

Genetic algorithms (GAs) (Mitchell 1996), inspired from biological evolution and based on the evolutionary ideas of natural selection and genetics, are one of the most powerful stochastic methods that solve optimization problems with the ability to explore a very large solution space within a limited time. GAs are domain independent and can be applied to any problem in different application domains. For any given problem, GAs may not be able to find the optimal solution, but in most cases the solution found is good enough and acceptable.

GAs use intelligent and random search to solve optimization problems (Mitchell 1996). Randomness is introduced through the use of mutation, crossover and selection of individuals that would be allowed to pass their genes to the next population. Search is intelligent, because the selection of individuals for reproduction and survival is also based on a domain and representation dependent fitness function. Aproppriate representation of a problem so that it can be solved using genetic algorithms is an important problem.

We use GA for our optimization problem, to decide on which of the base classifiers to be included in the classifier set to optimize test accuracy performance of the particular classifier combination method. In other words, our fitness function value is the test accuracy. However since we do not know the labels of the test set, we determine the classifier set by using a validation set instead of the test set.

In our GA, we have a population that consists of 20 individuals. The length of the chromosome is the number of base classifiers to be selected. We generate and evaluate the population for 500 times. We use JGAP library (et al. 2014) with default configuration for crossover, mutation and selection. The default rate for crossover is 35%, that is, per generation total number of crossover operations are populationSize * 0.35 and each crossover produces 2 individuals. Crossover point is selected randomly. Mutation rate is 1/12. Mutation is applied to the individuals excluding the ones produced by crossover. The selector is elitist, that is, best 90% of the previous

population is inherited to the next generation. Our fitness function calculates and returns the validation accuracy performance with the particular classifier combination method using the selected classifiers. We give the genetic algorithm that we use in Algorithm 2.

Algorithm 2 Genetic algorithm.

Input: $G = (V,E), V_{train}, V_{validation}, V_{test}$
Input: X_{train}, R_{train} (training inputs and outputs)
Input: $X_{validation}$ (validation inputs)
Input: $NumberOfEvolutions$ = 500
Input: $PopulationSize$ = 20
Input: $ChromosomeSize$ = K (Number of base classifiers)
Input: $FitnessFunction$ = Accuracy of selected classifiers among $g_1, \ldots, g_K$ on $V_{validation}$, using a specific classifier combination method, h_{comb} where $comb \in \{AVE, MAX, ENT, AVG-LA, AVG-BETA, MAX-LA, MAX-BETA\}$.
Output: $SelectedClassifiersSet$

```
InitializePopulation()
fitnessMax = 0
evolution = 0
GenerateRandomInitialPopulation()
for evolution = 0 to NumberOfEvolutions do
    for individual = 0 to PopulationSize do
        fitnessOfTheIndividual = calculateFitness()
        if fitnessOfTheIndividual > fitnessMax then
            fitnessMax = fitnessOfTheIndividual
            fittestIndividual = individual
        end if
    end for
    applyCrossOver()
    applyMutation()
    selectNextGeneration()
end for
return convertFittestIndividualToSelectedClassfiersSet(fittestIndividual)
```

3.3 Experiments

3.3.1 Experimental setup

In this section, we present a number of experiments to compare different classifier combination methods' performance.

3.3.1.1 Datasets

Citeseer, Cora, HepTh scientific publication datasets and WebKb web pages dataset are used in the experiments. For details about these datasets please refer to Section 1.4.

3.3.1.2 Sampling

We produce test-train splits 10 times and report the average accuracies over these folds for all experiments. In our experiments, we used different train/test percentages, where test percentage is changed from 10 percent to 90 percent by an increment of 10 percent of the whole dataset. The validation set ratio is kept as fixed as 5 percent of the whole dataset and used with training partition when necessary. For all of these combinations, we used both random sampling and snowball sampling whose details are given in Section 1.2.1.

3.3.1.3 Classification methods

A base classifier which is trained on node features and local connectivity information is needed for collective classification.

In this section of the thesis, we use logistic regression (LR), Support Vector Machine (SVM), Naive Bayes (NB), Bayes Net (BN), k-Nearest Neighbor (kNN, k=1 and k=3), Random Forest (RF), Radial Basis Function (RBF) and J48 for the g_{CO}, g_{LO} and g_{COLO} classifiers. For all of the methods Weka implementations with default parameters (unless otherwise noted) have been used.

3.3.2 Experimental results

3.3.2.1 Base classifier experiments

First of all, we conducted the experiments with different base classifiers given in Section 3.3.1.3 by using content only, link only and ICA for all different train/test ratios mentioned in Section 3.3.1.2 for both random and snowball sampling for all datasets considered. Since a total of 10 different classifiers were used with two different sampling methods with 9 different train/test ratios for each fold, 180 content only

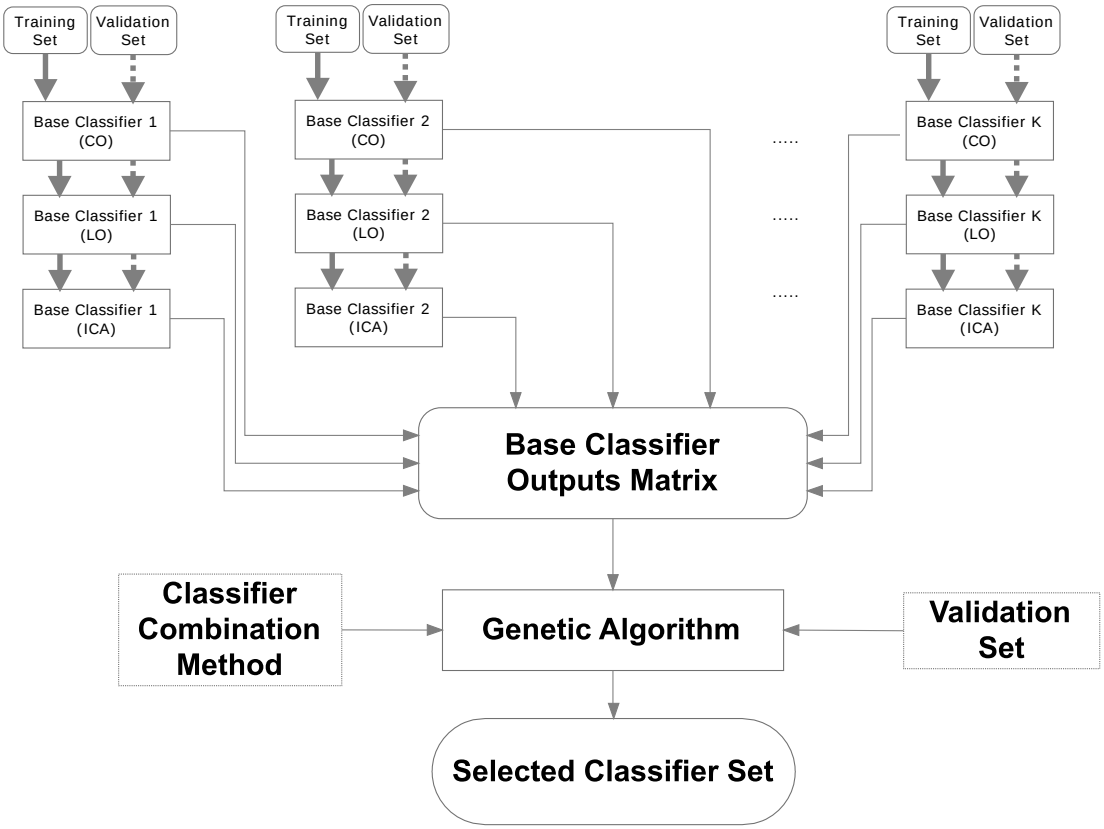


Figure 3.1 : Classifier selection procedure flow.

experiment results, 360 link only results, and 360 ICA results were obtained. For link only and ICA, we did not only generate the results obtained at the end of the iterations, but also the result obtained during start of the iterations. That's why the results obtained for link only and ICA were twice of the content only results. Since we used 10-fold cross validation, during the experiments, the number of results were multiplied by 10, resulting with a total of 9000 experiment results. Each result file consists the class membership probability distribution vector outputs generated by the corresponding classifier for all the nodes.

For each fold each experiment repeated twice differentiating with if the validation set is included or not in the training stage.

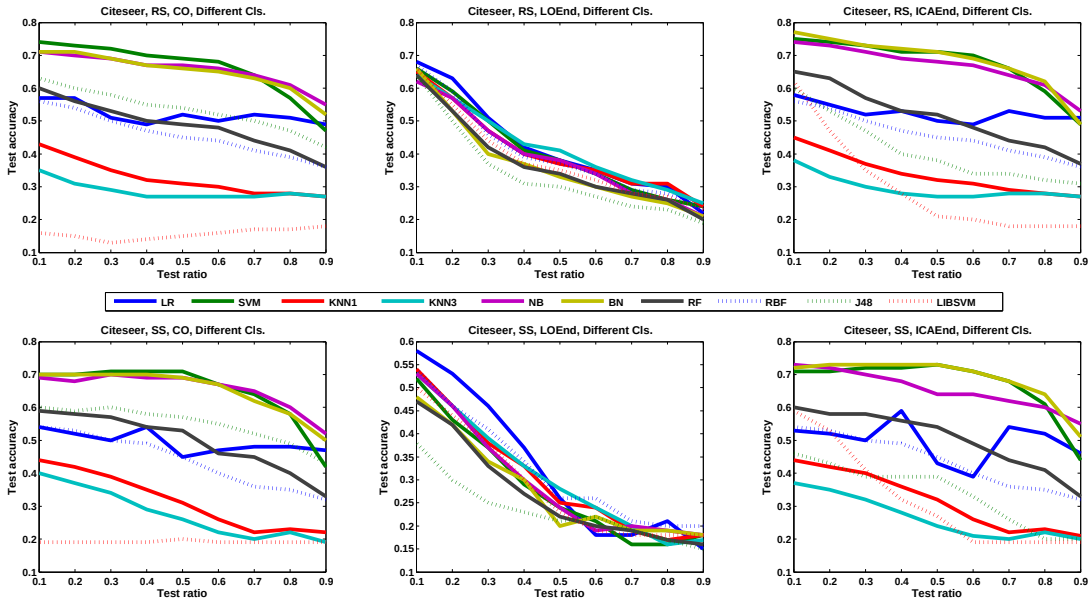


Figure 3.2 : Comparison of test accuracy performance of different classifier combination methods on Citeseer dataset.

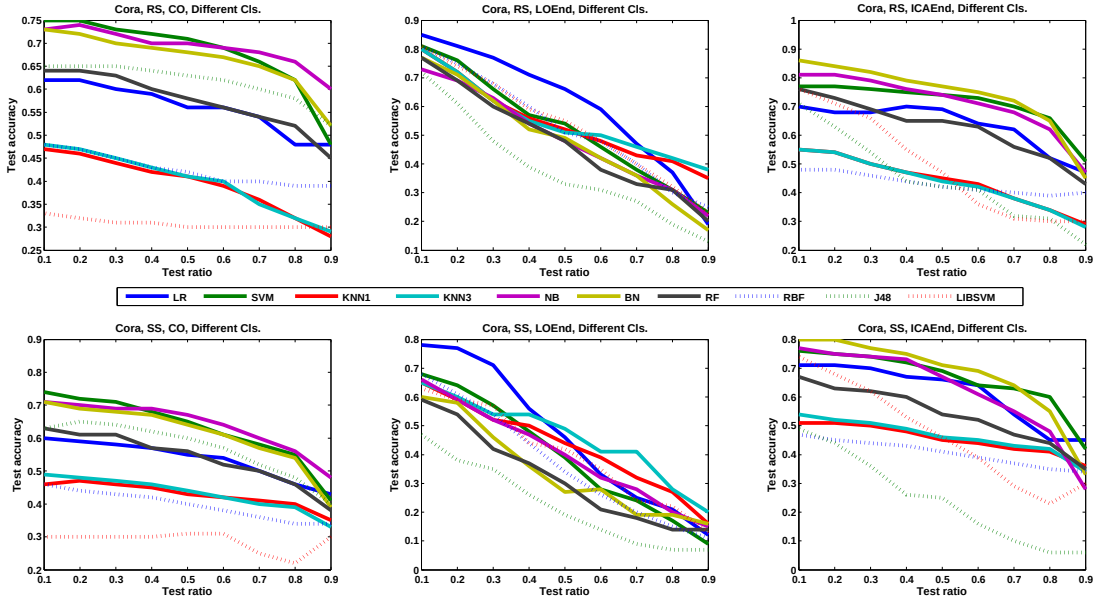


Figure 3.3 : Comparison of test accuracy performance of different classifier combination methods on Cora dataset.

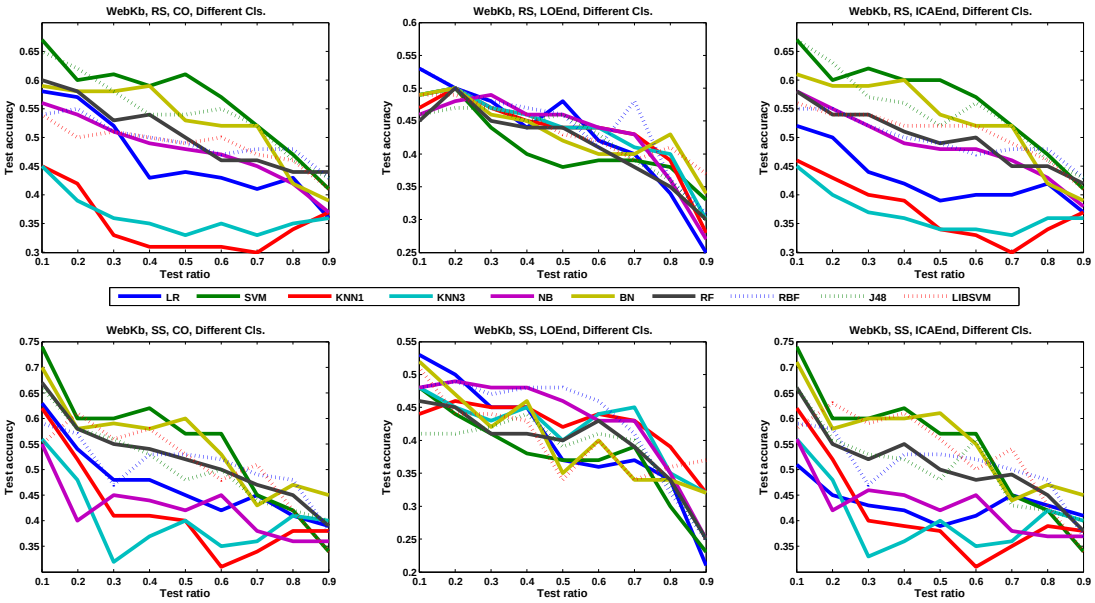


Figure 3.4 : Comparison of test accuracy performance of different classifier combination methods on WebKb dataset.

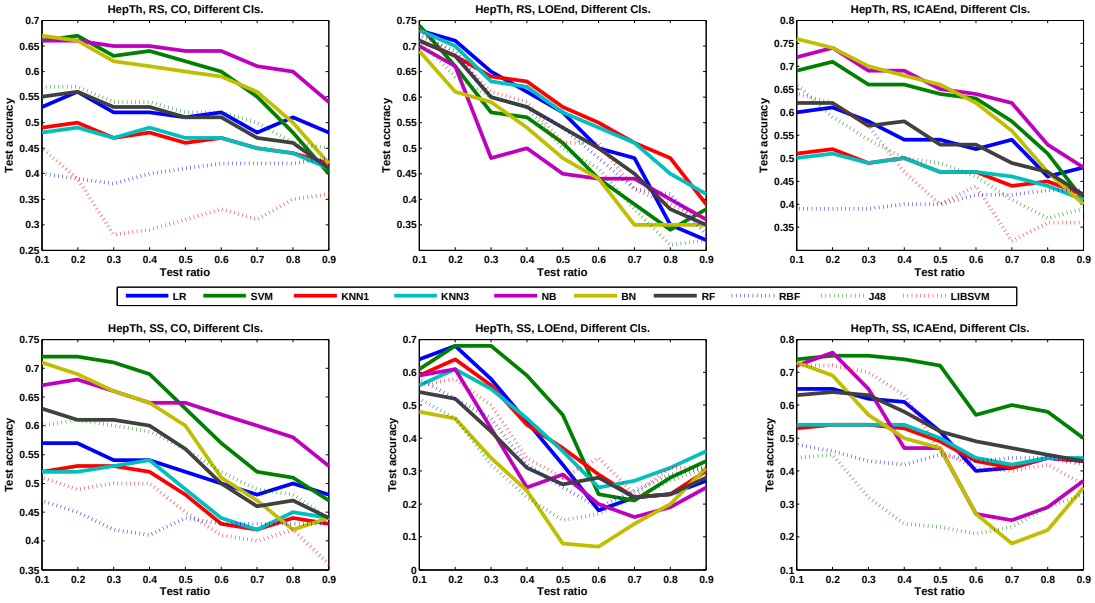


Figure 3.5 : Comparison of test accuracy performance of different classifier combination methods on HepTh dataset.

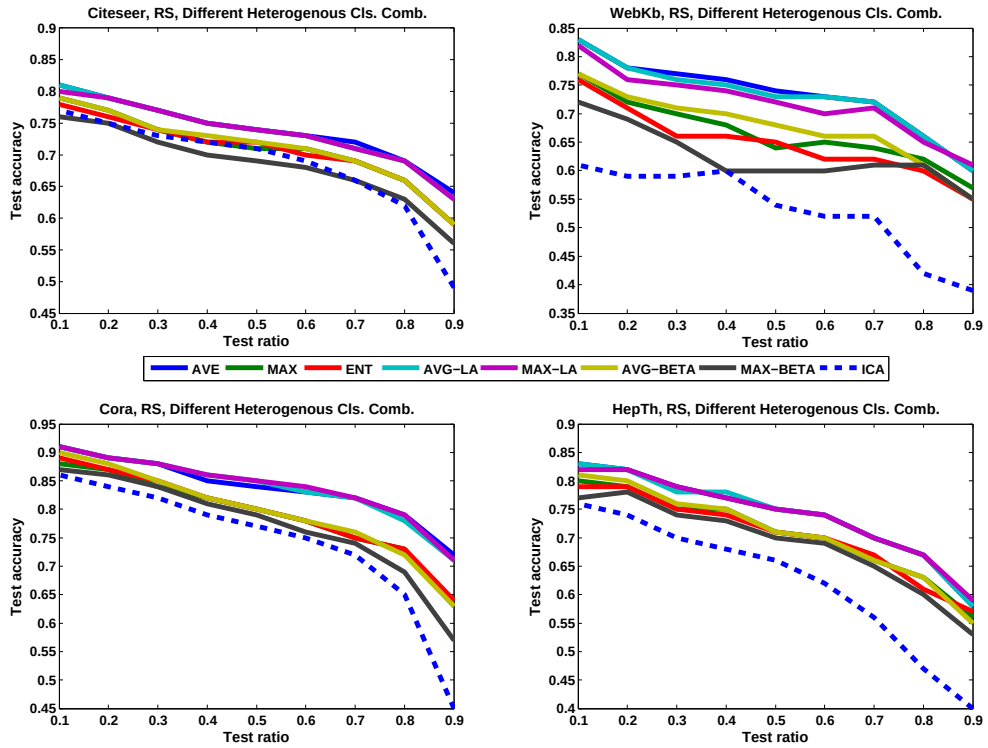


Figure 3.6 : Comparison of different classifier combination methods, with random sampling.

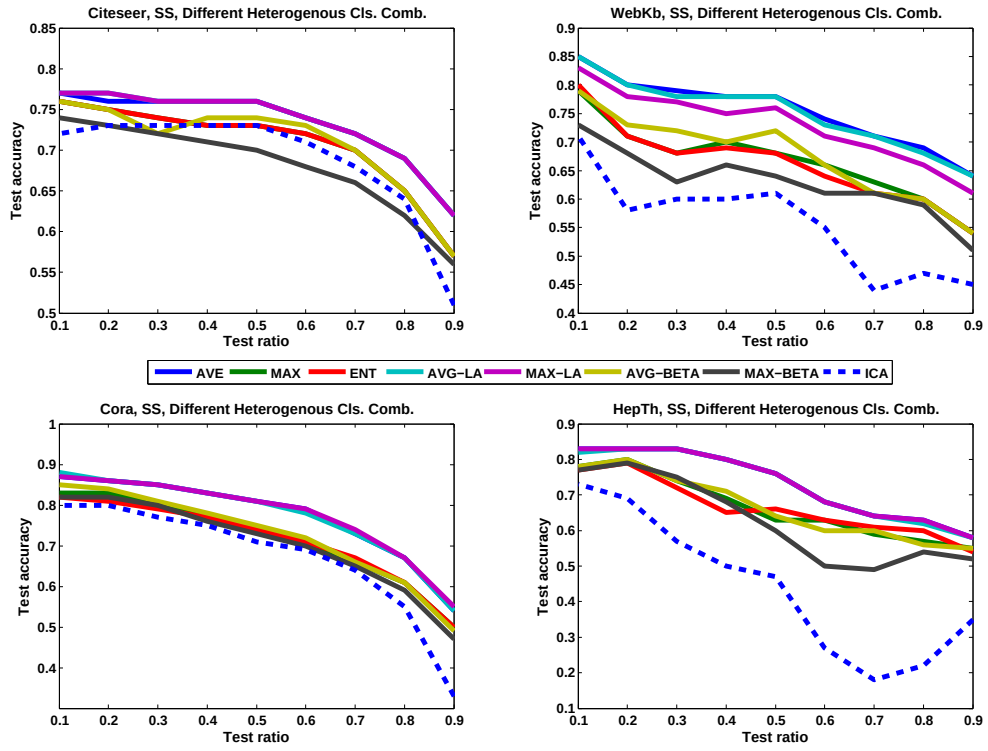


Figure 3.7 : Comparison of different classifier combination methods, with snowball sampling.

3.3.2.2 Performance of different classifiers

The results of the experiments mentioned in Section 3.3.2.1 are given in figures 3.2, 3.3, 3.4 and 3.5. They show the accuracies obtained when content only (CO), link only (LO) and ICA classifiers were used with a specific classification method for each dataset. Since for the base classifiers, standard deviations of the results are at most 0.01 for Citeseer and Cora datasets and at most 0.03 for WebKb and HepTh datasets, we did not plot error bars to keep the figures more clear.

For CO classification, SVM, BN and NB performed best for almost all different test ratios for Citeseer and Cora datasets. For WebKb dataset, SVM and BN performed best. When we consider HepTh dataset, SVM, BN and NB were the best performers, noting that NB was better from the others especially when test data increased.

For LO classification, LR performed best when train/test ratio is high and kNN classifiers performed best when train/test ratio is low. For WebKb dataset, almost all of the classifiers except LR and SVM performed similarly. Again, the behavior of LR was similar to the one in Citeseer and Cora's. Classifiers also behaved similarly on HepTh dataset when random sampling is considered. However when snowball sampling is considered, SVM had a superior behavior to the others when test ratio is between 0.2 and 0.6.

For collective classification, ICA, BN and SVM were the best performers. BN was a bit superior to SVM for Citeseer dataset. When test ratio is above 0.8, their performances decreased below NB. BN performed best when Cora dataset is considered, however still if the test data increases 70% of the dataset, SVM performed better than BN. When we consider WebKb dataset, SVM was the best performer almost for all test ratios. BN followed it, approaches more to it when snowball sampling is considered. For HepTh dataset, BN was the best performer when random sampling is considered. For test ratios above 0.6, NB performed better than BN. When we consider snowball sampling, SVM was the best performer, and BN, surprisingly failed to achieve good performance.

The overall picture shows us that ICA performed better than CO methods. The results of these experiments also show us that for different train/test ratios, with different sampling methods, on different datasets, different classifiers may perform better. Using the same classifier for all the domain, should lead to loss of information.

3.3.2.3 Heterogeneous classifier combination

Based on the results obtained, the question arose: How can we best combine the classifiers to obtain the best result? For combination, we used content only, link only iterations end and ICA iterations end experimental results. There were a total of 30 classifiers for this case. To combine these results, we used the proposed methods in Section 3.2.1. Our main objective was to maximize the test accuracy performance with each method. To accomplish this, we used the validation set for the fold, to decide which of the classifiers should be selected for the corresponding fold with the corresponding classifier combination method. However, since there were 30 classifiers to be considered, we had a total of 2^{30} subsets of the classifiers could be selected. Since each calculation takes about 8 milliseconds on laptop having a 8 cores, 2.2 Ghz. i7 processor with 16 GB. of RAM with 480GB. SSD hard disk, on average, 2^{30} calculations for 4 datasets, 9 test ratios, 7 different methods for 10 different folds execution, it was almost impossible to calculate and check all combinations. Therefore, we used a genetic algorithm to be able to find a good enough and an acceptable solution for our combination problem. The objective function for the genetic algorithm we used, returned the accuracy obtained on the validation set using corresponding classifier combination method. That is, whenever we use AVE method for combination, we use the classifier set that maximizes validation test accuracy on the fold with AVE combination method. Then, we selected the classifier set that maximizes the validation accuracy. We used the same classifier set to assess accuracy on the test set. However in this case, we also used validation set in training the corresponding classifier (See figure 3.1).

Table 3.1 : A sample of selected classifiers for each fold (Cora, SS, Test = 0.5, AVE).

	# Sel.	LR			SVM			KNN1			KNN3			NB			BN			RF			RBF			J48			LIBSVM		
		CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA	CO	LO	ICA
F0	16			x		x		x			x	x	x	x	x	x		x	x				x			x			x	x	x
F1	7			x					x						x	x		x	x												x
F2	13			x	x	x	x		x			x		x		x			x	x		x		x							x
F3	11			x		x						x		x	x				x	x		x			x				x		x
F4	7			x					x					x		x			x			x									x
F5	11		x	x							x	x		x		x	x		x	x	x					x					
F6	7			x		x						x		x	x	x			x												
F7	11		x	x		x			x			x		x		x			x			x						x			x
F8	11	x	x	x								x				x			x	x				x	x			x			x
F9	13		x	x			x					x			x	x	x		x	x			x			x		x		x	

3.3.2.4 Performance of heterogeneous classifier combination

The results of the experiments mentioned in Section 3.3.2.3 is given in figure 3.6 for random sampling and in figure 3.7 for snowball sampling. Each figure shows the accuracies obtained with the proposed classifier combination methods for each dataset. To be able to show the increase in performance, we also included our baseline best performer base classifier, BN-ICA, in these figures.

Almost for all the datasets, all the proposed classifier combination methods achieved better performance than best performer base classifier BN-ICA. However, especially AVE, AVG-LA and MAX-LA were the best performers among the others for both random sampling and snowball sampling on all datasets considered. ENT, AVG-BETA and MAX methods performed similarly, still they performed well from our baseline ICA, but worse than AVE, AVG-LA, MAX-LA methods. MAX-BETA method was the worst performer in the proposed combination methods, still had good performance than our baseline method. We see that using local alpha for normalization is beneficial, since LA-MAX method was one of the best performers and had significant better performance than MAX method. However it is difficult to say the same thing for beta normalization. Since beta normalized versions of AVE and MAX methods, both had significant performance decrease. It is better to note that for these experiments, for α_{CO} and α_{LO} computations the number of hops to explore for neighborhood was chosen to be $h^*_{CO} = 2$ and $h^*_{LO} = 1$, while as for β_{CO} and β_{LO} computations the number of hops was chosen to be $h^*_{CO} = 1$ and $h^*_{LO} = 1$.

Due to the complexity of the proposed methods, we prefer and suggest use of AVE method among AVG-LA and MAX-LA methods. That's why we present a new figure, Figure 3.9, which compares our baseline BN-ICA with AVE classifier combination method for both sampling methods at once to be able to visualize and summarize the increase in performance by using our proposed heterogeneous classifier combination methods. In figure 3.9, straight lines show the test accuracy performances obtained by AVE combination method and dashed lines show the baseline BN-ICA. The test accuracy increase in performance is significant and consistent for all different train/test ratios for the datasets.

Just as our claim in the beginning, we showed that for a classifier combination method even on different portions of train/test data, a set of different classifiers combination may perform better. Due to lack of space, we gave the selected classifiers just for ten folds on Cora dataset, for AVE classifier combination method with a test ratio of 0.5 (See Table 3.1).

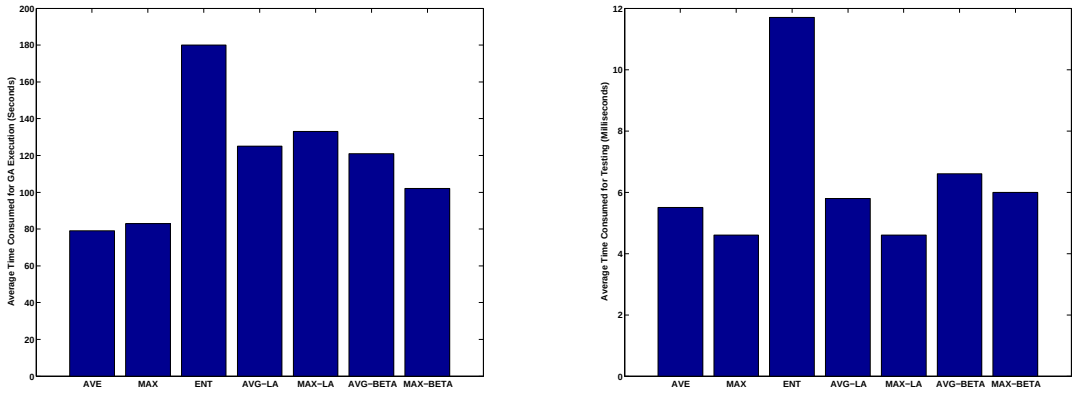


Figure 3.8 : Average genetic algorithm execution time (left side) and average testing time of classifier combination methods (right side) for Cora dataset, snowball sampling, test ratio = 0.5.

When we consider the time consumed, we see that almost all proposed classifier combination methods perform extremely fast in terms of milliseconds for one fold execution. However until testing, we have the another time consuming process which is the classifier selection phase with the genetic algorithm. The execution of classifier selection phase takes a few minutes for one classifier combination method per fold (See Figure 3.8). The most time consuming part is the base classifiers training time. We should recall that for any classification problem, generally a bunch of classifiers

are trained. So, this phase can be assumed in regular cycle of the classification problem solving process.

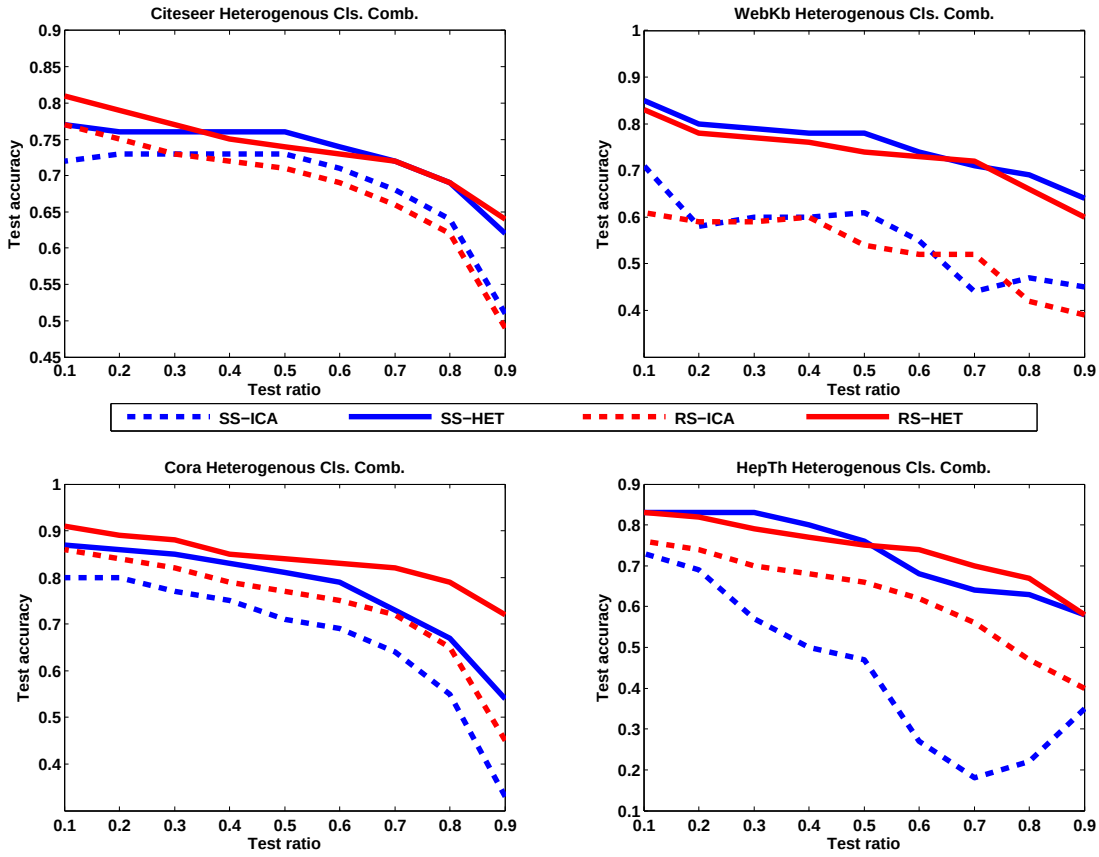


Figure 3.9 : Test accuracy comparison of ICA with AVE heterogeneous combination method.

3.4 Discussion

In this section, we presented content only, link only and ICA classification results for different train/test ratios on four datasets, namely Cora, Citeseer, WebKb and HepTh datasets. We prepared HepTh dataset for collective classification. Although HepTh dataset was also used previously in collective classification by McDowell and colleagues (McDowell et al. 2009), they used only the words in title or name of the corresponding journal to construct the content features. However, we found the matching abstracts of papers and used the words in the abstracts to construct the content features. The number of non-relational features was 387 for their case, while it is 2295 for our case.

For the datasets used, we have shown that for CO classification, SVM, BN and NB performed best, while LR performed best when LO classification is considered. BN and SVM were the best performers for ICA. So, for content only, link only or ICA classification, based on the characteristics of inputs, different classifiers may perform better than the others. Using the same classifier for all the domain, should lead to loss of information. To overcome this problem, we proposed seven different heterogeneous classifier combination methods (See Section 3.2.1). One of the major problems, we faced, was to find the classifier set which should participate to combination process to get better classification performance. We proposed a genetic algorithm based selection method for this problem. We used the validation set with the corresponding classifier combination method to find the classifier set to be selected. Our methods outperformed best base classifiers. The test accuracy increase in performance is significant and consistent for all different train/test ratios for the datasets. Especially AVE, AVG-LA and MAX-LA combination methods were the best performers among others. Due to the complexity of the proposed methods, we prefer and suggest use of AVE method among AVG-LA and MAX-LA methods.

We showed that using local alpha (average accuracy of the neighbors in training set), for normalization is beneficial, while it is difficult to say the same thing for beta (average homophily of the neighbors in training set) normalization. Since for content only classifiers, generated content only graph is used for calculation of local alpha and beta, the structure of the content graph is important. We think that the main reason not to be able to get better performance with local alpha based methods is the low homophily of the content graph generated. Generating a high homophily content graph is an interesting problem that we are planning to investigate in the near future.

The proposed method can also be extended to collective classification scenarios with multiple types of content and link. This is another possible future research direction.

4. COLLECTIVE CLASSIFICATION ON A NETWORK WITH DIRECTED LINKS

4.1 Background and Purpose

Up to this point in the thesis, for the sake of simplicity, links in the networks are assumed to be undirected. However, direction information was originally available for the datasets used. Not to take into account direction information, when available, may cause useful information for classification to be lost. That's why, in this section, the effect of using directed link information on classifier performance in collective classification is explored in detail.

Performance of collective classification algorithms using different local classifiers are evaluated on Citeseer, Cora, WebKb and HepTh datasets with random and snowball sampling, using directed and undirected links for different train/test ratios. It has been shown that by using directed graphs, significant performance increase is obtained when link only classifier is used. This performance increase is also reflected in the collective classification (ICA) results. Whenever directed links are used with snowball sampling, the performance of the content only classifier decreased slightly. The reason for this is, snowball sampling is related to links and in case of usage of directed links, a growing snowball has less directions to grow compared to an undirected graph. Content only classifiers' performance for random sampling was not affected by directed links, since random sampling is independent of the links.

4.2 Methodology

4.2.1 Collective classification with directed links

First of all, directed graphs were constructed for the datasets. The main question should be answered here was how to construct the relational features so as to use the direction

information. Traditional, undirected relational features, are constructed usually by aggregating the class labels of the other linked instances (McDowell et al. 2009; 2007, Sen et al. 2008). However, when direction information is available, the concept of neighborhood needs to be redefined. For each node in the directed graph, there are in links and out links. Considering only the out links for neighborhood, again, might cause useful information to be lost. So, for each node, we construct the relational features by concatenating two vectors, namely in neighbors vector, and out neighbors vector. In neighbor vector is constructed by aggregating the class labels of the other linked instances, which connect to the node. Out neighbor vector is constructed as in the traditional one by aggregating the class labels of the other linked instances, which the node connects. In Figure 4.1, the construction of relational features is summarized. The nodes in the graph belong to 3 different classes A, B and C. On the left side relational features are constructed assuming the graph is undirected, while on the right side the graph is taken to be a directed graph. In this section, as in the previous sections, the Count aggregation method is used. Directed relational feature vector's size is double size of the undirected one. This is a consequence of using in links in the first part and out links in the latter part of the vector.

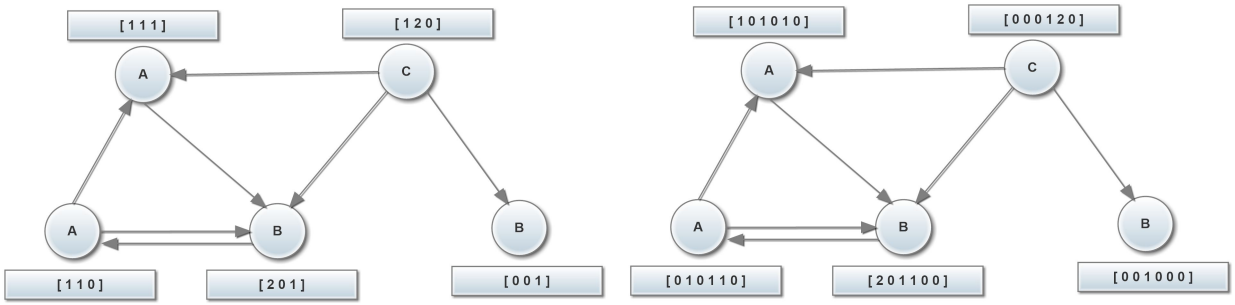


Figure 4.1 : Construction of relational features, left: undirected graph, right: directed graph.

4.3 Experiments

4.3.1 Experimental setup

4.3.1.1 Datasets

Citeseer, Cora, HepTh scientific publication datasets and WebKb web pages dataset are used in the experiments. For details about these datasets please refer to Section 1.4.

4.3.1.2 Sampling

Test-train splits are produced 10 times and average accuracies over these folds are reported for all experiments. In the experiments, different train/test percentages are used, where test percentage is changed from 10 percent to 90 percent by an increment of 10 percent of the whole dataset. For all of these combinations, both random sampling and snowball sampling are used whose details are given in Section 1.2.1.

4.3.1.3 Classification methods

A base classifier which is trained on node features and local connectivity information is needed for collective classification.

In the experiments, logistic regression (LR), Support Vector Machine (SVM), Naive Bayes (NB), Bayes Net (BN), k-Nearest Neighbor (kNN, k=1 and k=3), Random Forest (RF), Radial Basis Function (RBF) and J48 are used as base classifiers for the g_{CO}, g_{LO} and g_{COLO} classification. Again, unless otherwise noted, for all of the methods Weka implementations with default parameters have been used.

4.3.1.4 Details of the experiments

First of all, we conducted the experiments with different classifiers given in Section 4.3.1.3 by using content only, link only and ICA for all different train/test ratios mentioned in Section 4.3.1.2 for both random and snowball sampling for all datasets considered using directed links and undirected links. Since a total of 10 different

classifiers were used with two different sampling methods with 9 different train/test ratios for each fold, 180 content only experiment results, 360 link only results, and 360 ICA results were obtained. For link only and ICA, we did not only generate the results obtained at the end of the iterations, but also the result obtained during start of the iterations. That's why the results obtained for link only and ICA were twice of the content only results. Since we used 10-fold cross validation, during the experiments, the number of results were multiplied by 10, resulting with a total 18000 experiment results composed of 9000 undirected and 9000 directed experiment results.

4.3.2 Experimental results

4.3.2.1 Performance of different classifiers

The results of the experiments mentioned in Section 4.3.1.4 are given in Figures 4.2 and 4.3 for Citeseer dataset, in Figures 4.4 and 4.5 for Cora dataset, in Figures 4.6 and 4.7 for WebKb dataset and in Figures 4.8 and 4.9 for HepTh dataset. To be able to visualize the results better, the results obtained with directed links versus undirected links, are presented in two different figures for each dataset. The figures are separated according to different set of classifiers, which of each includes 5 different classifiers. The first classifier set includes LR, SVM, BN, NB and RF, while the second classifier set includes kNN1, kNN3, RBF, J48 and libSVM. The test accuracy differences for the cases of directed graphs versus undirected graphs are given in Figures 4.10, 4.11, 4.12 and 4.13.

According to the Figures 4.2, 4.3 and 4.10, on Citeseer dataset, almost all classifiers get benefit from directed links regardless of the sampling method used, when link only classification is considered. Especially they benefit more when snowball sampling is used. While J48, BN and libSVM get benefit more with random sampling, BN, NB, kNN3, LR and especially SVM get benefit more with snowball sampling when link only classification is considered. Directed links are also useful for ICA. Especially J48 benefits best when directed links are used with ICA.

According to the Figures 4.4, 4.5 and 4.11, on Cora dataset, almost all classifiers get benefit from directed links regardless of the sampling method used when link only

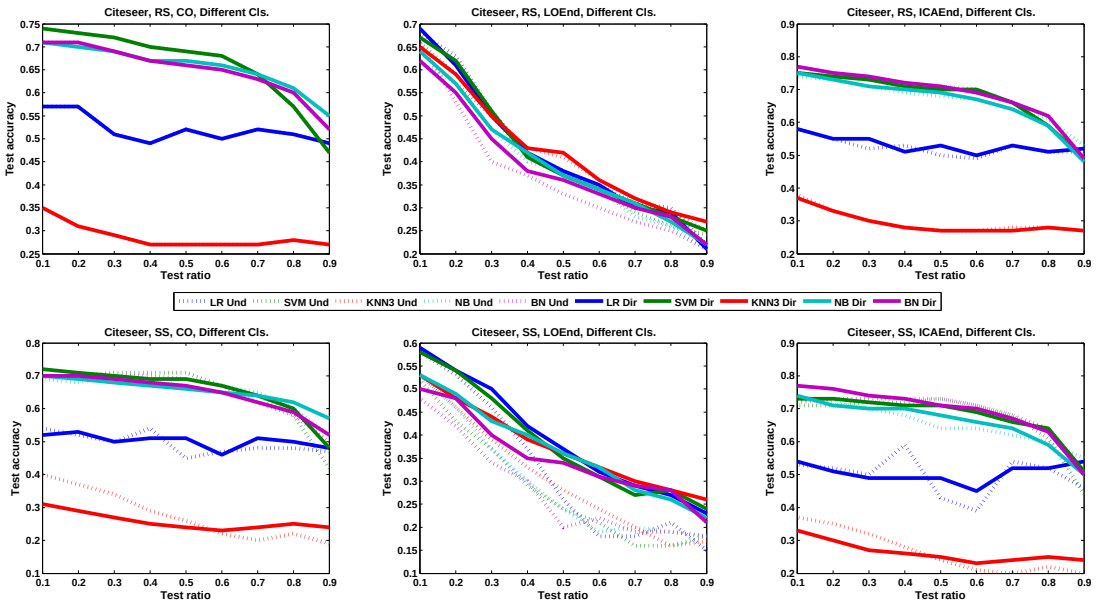


Figure 4.2 : Citeseer SS, DSS, RS, DRS comparison (classifier set 1).

classification is considered. However especially they benefit more when snowball sampling is used. kNN3 benefits best from directed links regardless of the sampling method used. kNN1, RF and SVM follow kNN3 if snowball sampling is used. Usage of directed links are also useful for ICA. NB, LR, SVM and especially J48, benefit more when directed links are used with ICA.

According to the Figures 4.6, 4.7 and 4.12, on WebKb dataset, LR, NB and BN get worse accuracies when directed links are used. libSVM, J48, RF and SVM benefit more for both sampling methods. Usage of directed links are also useful for ICA. Especially BN benefits best when directed links are used with random sampling. SVM, kNN3 and NB benefit more with ICA when snowball sampling is used.

According to the Figures 4.8, 4.9 and 4.13, on HepTh dataset, almost all classifiers get benefit from directed links when snowball sampling method is used with link only classification. RF, J48 and especially BN benefit more. When random sampling is considered, accuracies obtained in case of usage of directed links do not differ significantly. Usage of directed links are also useful for ICA. NB, J48, and especially BN benefit more when directed links are used with ICA.

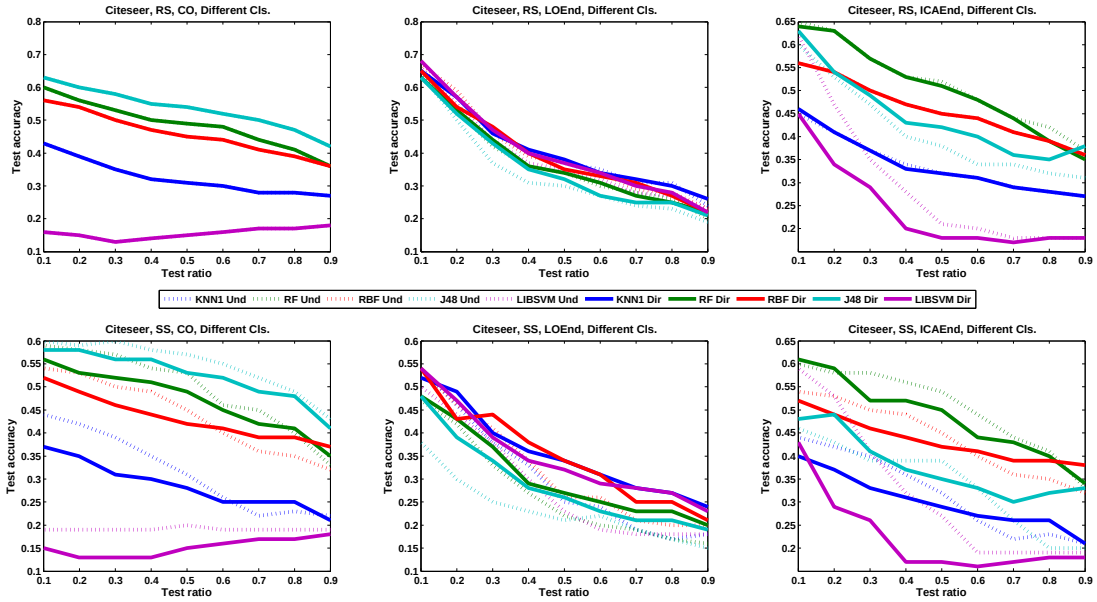


Figure 4.3 : Citeseer SS, DSS, RS, DRS comparison (classifier set 2).

4.4 Discussion

It is observed that using directed links provides significant test accuracy performance increase on high homophily datasets when link only classifier is used. Meaning that directed features are more useful than undirected ones and when we do not use direction information, whenever is available, we may lose some useful information.

Especially link only classifier benefits more from direction information. This useful information also reflects to collective classification (ICA) and we get better results as well. However when homophily of the graph is low, using directed information does not help as expected since links are naturally are not much helpful, like as in the WebKb dataset.

One important thing to mention that whenever directed links are used, the performance of the content only classifier decreases slightly. The reason for that is, snowball sampling is related to links and in case of usage of directed links, growing snowball has less directions to grow compared to undirected one. Content only for random sampling is not affected by directed links since the sampling is independent of the links.

It has been shown that different classifiers may benefit more from directed information on different datasets.

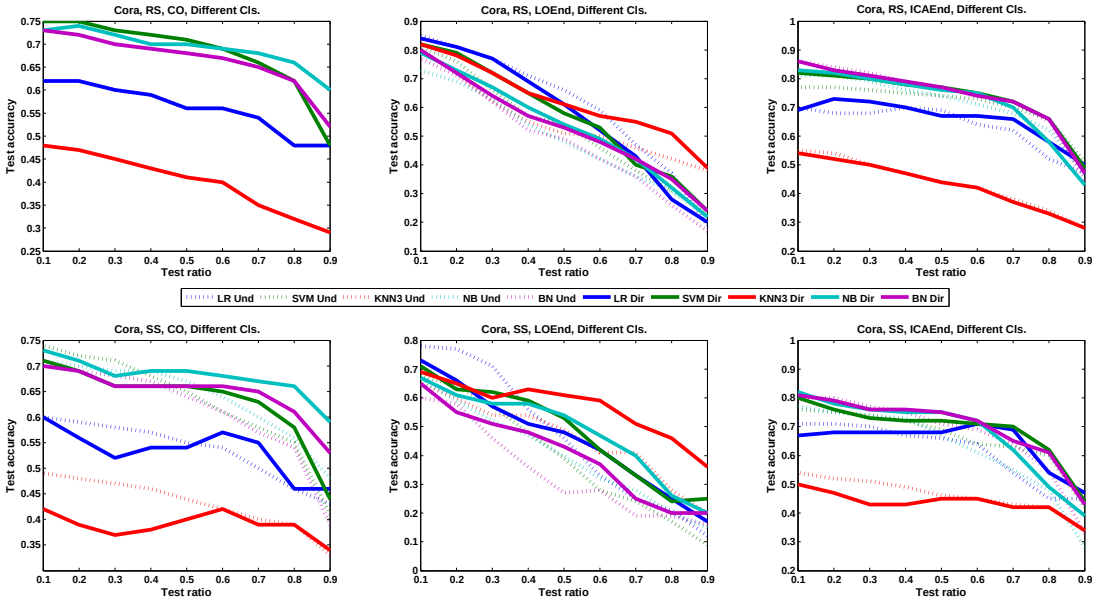


Figure 4.4 : Cora SS, DSS, RS, DRS comparison (classifier set 1).

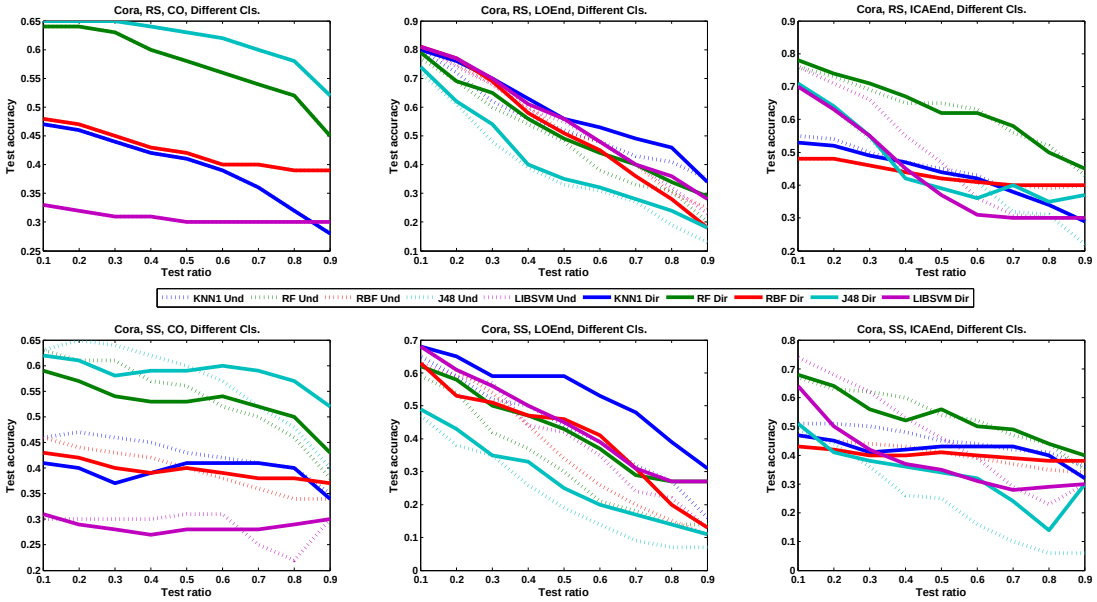


Figure 4.5 : Cora SS, DSS, RS, DRS comparison (classifier set 2).

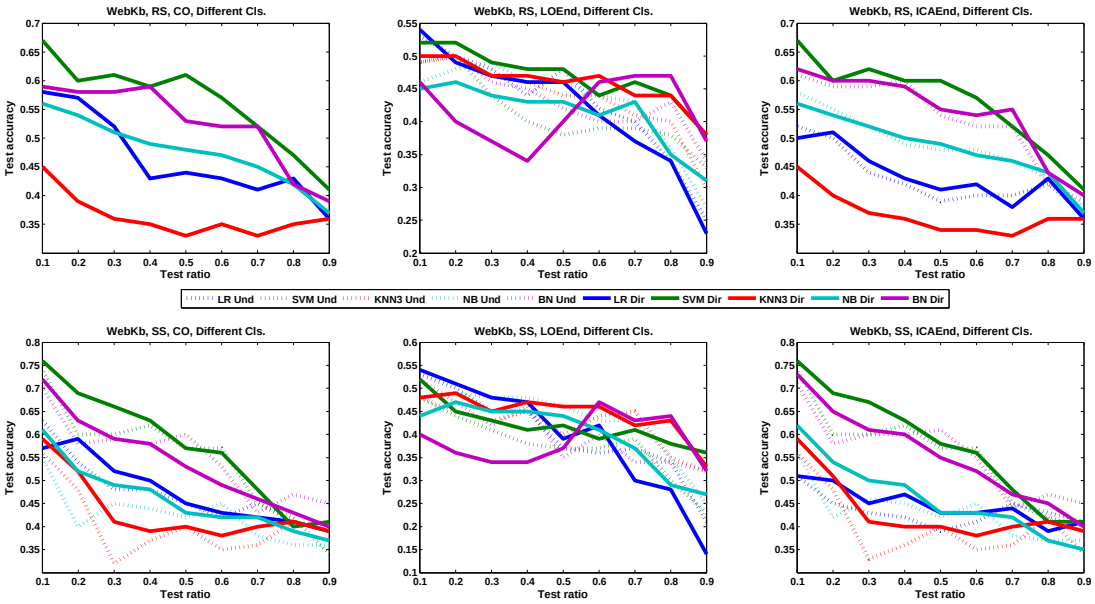


Figure 4.6 : WebKb SS, DSS, RS, DRS comparison (classifier set 1).

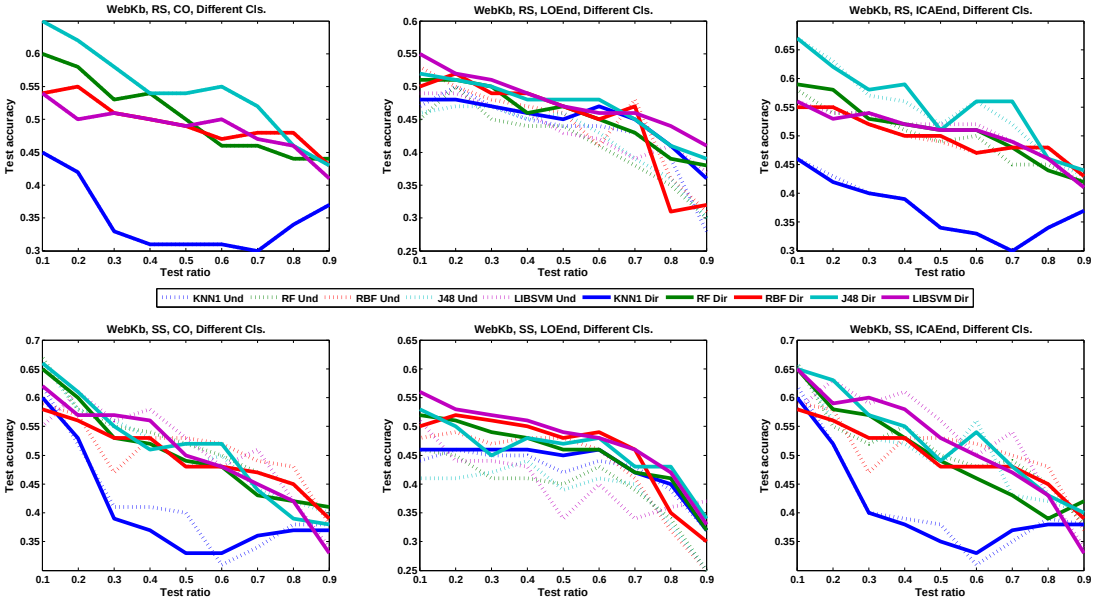


Figure 4.7 : WebKb SS, DSS, RS, DRS comparison (classifier set 2).

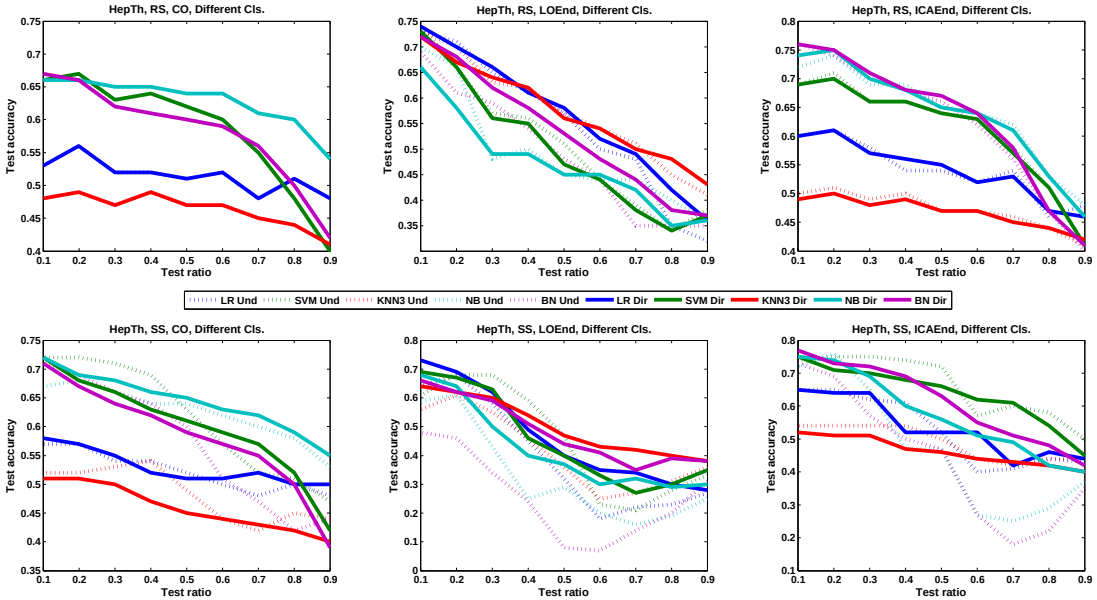


Figure 4.8 : HepTh SS, DSS, RS, DRS comparison (classifier set 1).

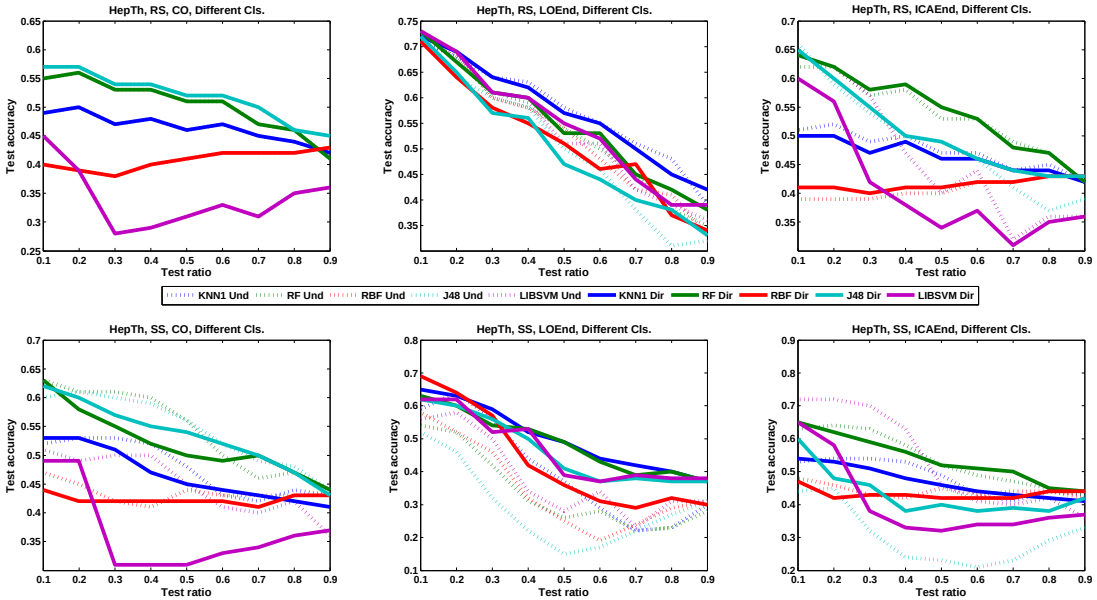
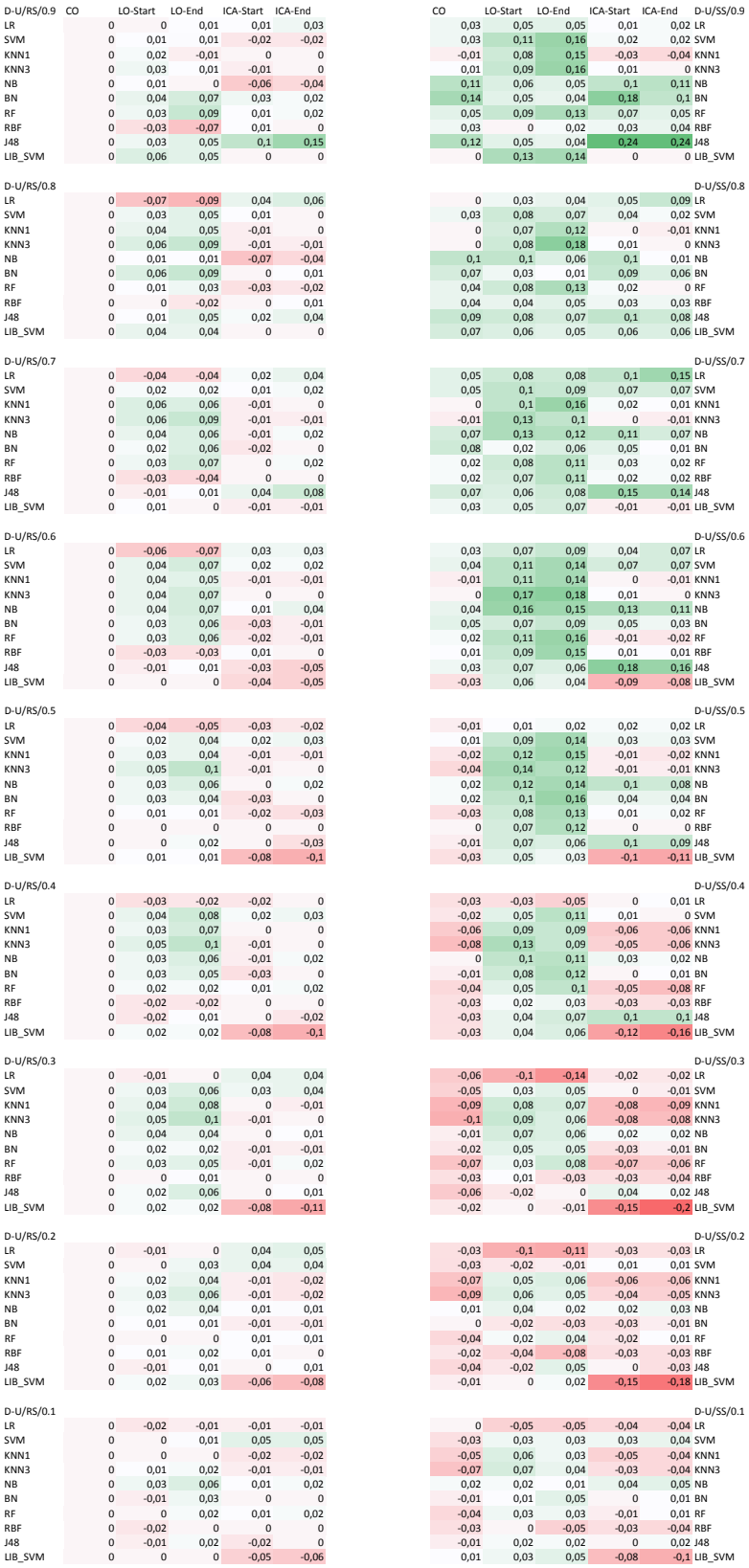


Figure 4.9 : HepTh SS, DSS, RS, DRS comparison (classifier set 2).

D-U/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End	CO	LO-Start	LO-End	ICA-Start	ICA-End	D-U/SS/0.9
LR	0	-0,01	-0,01	0,01	0,01	0,01	0,05	0,08	0,08	0,08	LR
SVM	0	0	0,01	0,01	0	0,06	0,04	0,06	0,07	0,07	SVM
KNN1	0	-0,01	0,02	0	0	-0,01	0,05	0,06	0	0	KNN1
KNN3	0	0	0,02	0	0	0,05	0,05	0,09	0,04	0,04	KNN3
NB	0	0	0,01	-0,08	-0,05	0,05	0,02	0,04	-0,06	-0,05	NB
BN	0	-0,01	0,01	0	0	0,02	0,03	0,03	0,08	-0,01	BN
RF	0	0,01	0,02	-0,02	-0,02	0,02	0,03	0,04	0,03	0,01	RF
RBF	0	-0,01	-0,01	0	0	0,05	0,01	0,01	0,05	0,06	RBF
J48	0	0,01	0,02	0,05	0,07	-0,02	0,05	0,04	0,09	0,13	J48
LIB_SVM	0	-0,01	-0,01	0	0	-0,01	0,04	0,05	-0,01	-0,01	LIB_SVM
D-U/RS/0.8											D-U/SS/0.8
LR	0	-0,01	-0,01	0	0	0,02	0,04	0,06	0,01	0	LR
SVM	0	0	0,02	0	0	0,02	0,08	0,12	0,02	0,03	SVM
KNN1	0	0	-0,01	0	0	0,02	0,07	0,1	0,03	0,03	KNN1
KNN3	0	-0,01	0	0	0	0,03	0,07	0,12	0,03	0,03	KNN3
NB	0	0	0,01	-0,04	-0,02	0,02	0,06	0,07	-0,02	-0,01	NB
BN	0	0,01	0,03	0	0	0,01	0,05	0,09	0,02	-0,01	BN
RF	0	0,01	-0,01	-0,01	-0,03	0,01	0,06	0,06	0,01	-0,01	RF
RBF	0	-0,01	-0,01	0	0	0,04	0,04	0,05	0,04	0,04	RBF
J48	0	0,01	0,02	0	0,03	-0,01	0,06	0,04	0,1	0,12	J48
LIB_SVM	0	0,01	0,01	0	0	-0,02	0,08	0,09	-0,01	-0,01	LIB_SVM
D-U/RS/0.7											D-U/SS/0.7
LR	0	0	-0,01	0	0	0,03	0,07	0,11	0	-0,02	LR
SVM	0	0,01	0,02	0	0	0	0,08	0,11	-0,01	-0,02	SVM
KNN1	0	0	0,01	0	0	0,03	0,05	0,09	0,04	0,04	KNN1
KNN3	0	-0,01	0	0	-0,01	0,04	0,05	0,1	0,04	0,04	KNN3
NB	0	0,02	0,03	-0,01	0	-0,01	0,06	0,08	0,02	0,02	NB
BN	0	0,02	0,03	-0,01	0	0	0,03	0,1	0	-0,01	BN
RF	0	0,01	-0,01	0	0	-0,03	0,05	0,04	-0,01	-0,01	RF
RBF	0	0	0,02	0	0	0,03	0,03	0,04	0,03	0,03	RBF
J48	0	0,01	0,01	0,01	0,02	-0,03	0	0,02	0,03	0,04	J48
LIB_SVM	0	0	0,02	-0,01	-0,01	-0,02	0,07	0,1	-0,02	-0,02	LIB_SVM
D-U/RS/0.6											D-U/SS/0.6
LR	0	-0,01	0	-0,01	0,01	-0,01	0,11	0,14	0,06	0,06	LR
SVM	0	-0,01	0	0	0	0	0,08	0,1	-0,01	-0,02	SVM
KNN1	0	0	-0,01	0	0	-0,01	0,07	0,07	0,01	0,01	KNN1
KNN3	0	-0,01	0	0	0	0,01	0,07	0,09	0,02	0,02	KNN3
NB	0	0,01	0	0	0	-0,02	0,09	0,14	0,04	0,02	NB
BN	0	0,01	0,03	-0,01	0	-0,02	0,05	0,09	0	-0,01	BN
RF	0	0,01	0,01	0	0	-0,01	0,05	0,05	-0,02	-0,05	RF
RBF	0	0	-0,01	0	0	0,01	0,04	0,05	0,01	0,01	RBF
J48	0	0	0	0,04	0,06	-0,03	-0,01	0,01	0,01	0	J48
LIB_SVM	0	0,02	0,02	-0,02	-0,02	-0,03	0,09	0,1	-0,02	-0,03	LIB_SVM
D-U/RS/0.5											D-U/SS/0.5
LR	0	-0,01	0	0,02	0,03	0,06	0,08	0,11	0,05	0,06	LR
SVM	0	-0,01	-0,01	0	-0,01	-0,02	0,07	0,11	-0,02	-0,02	SVM
KNN1	0	0	0,01	0	0	-0,03	0,07	0,09	-0,03	-0,03	KNN1
KNN3	0	0	0,01	0	0	-0,02	0,07	0,08	-0,01	0,01	KNN3
NB	0	0	-0,01	0,01	0,01	-0,03	0,1	0,12	0,05	0,04	NB
BN	0	0	0,03	-0,01	0	-0,02	0,08	0,14	0	-0,02	BN
RF	0	0	0	0	-0,01	-0,04	0,03	0,05	-0,03	-0,04	RF
RBF	0	-0,01	-0,02	0	0	-0,03	0,07	0,08	-0,03	-0,03	RBF
J48	0	0	0,02	0,02	0,04	-0,04	0,03	0,05	0,01	-0,04	J48
LIB_SVM	0	0,01	0,02	-0,03	-0,03	-0,05	0,07	0,09	-0,1	-0,1	LIB_SVM
D-U/RS/0.4											D-U/SS/0.4
LR	0	0	0	-0,01	-0,02	-0,03	0,05	0,05	-0,09	-0,1	LR
SVM	0	-0,02	0	0	0	-0,02	0,05	0,12	0	-0,01	SVM
KNN1	0	0	0,01	0	-0,01	-0,05	0,05	0,03	-0,05	-0,05	KNN1
KNN3	0	-0,01	0	0	0	-0,04	0,07	0,06	-0,02	-0,02	KNN3
NB	0	0,01	0,02	0	0,01	-0,02	0,09	0,1	0,03	0,02	NB
BN	0	-0,01	0,01	-0,01	0	-0,02	0,05	0,05	0	0	BN
RF	0	0,02	0	-0,01	0	-0,03	0,02	0,02	-0,02	-0,04	RF
RBF	0	0,01	0,01	0	0	-0,05	0,03	0,04	-0,05	-0,05	RBF
J48	0	0	0,04	0,01	0,03	-0,02	0,01	0,05	0,01	-0,02	J48
LIB_SVM	0	0,01	0,03	-0,07	-0,08	-0,06	0,04	0,04	-0,14	-0,15	LIB_SVM
D-U/RS/0.3											D-U/SS/0.3
LR	0	-0,01	-0,01	0,02	0,03	0	0,03	0,04	-0,01	-0,01	LR
SVM	0	-0,02	0,01	0	0	-0,01	0,08	0,11	0	0	SVM
KNN1	0	0	-0,01	-0,01	0	-0,08	0,06	0,02	-0,06	-0,07	KNN1
KNN3	0	0	0	-0,01	0	-0,07	0,07	0,05	-0,04	-0,05	KNN3
NB	0	0,01	0	0	0	-0,02	0,06	0,06	0,01	0	NB
BN	0	0,01	0,05	0	0,01	-0,01	0,06	0,06	0,01	0,01	BN
RF	0	0	0,02	-0,01	0	-0,05	0,05	0,04	-0,04	-0,06	RF
RBF	0	-0,01	0,02	0	0	-0,04	0,01	0,03	-0,04	-0,04	RBF
J48	0	0,01	0,06	0	0,02	-0,04	0,03	0,09	0,02	0,02	J48
LIB_SVM	0	0,02	0,03	-0,06	-0,06	-0,06	0,01	0,01	-0,13	-0,15	LIB_SVM
D-U/RS/0.2											D-U/SS/0.2
LR	0	-0,01	-0,02	0	0	0,01	0,03	0,01	-0,01	-0,01	LR
SVM	0	-0,01	0,03	0	0	0,01	0,09	0,11	0,02	0,02	SVM
KNN1	0	0	0	0	0	-0,07	0,07	0,03	-0,04	-0,05	KNN1
KNN3	0	0,01	0,02	0	0	-0,08	0,06	0,02	-0,06	-0,05	KNN3
NB	0	0	0	0	0	0,01	0,04	0,03	0	-0,01	NB
BN	0	-0,01	0,02	0	0	0	0,05	0,06	0,03	0,03	BN
RF	0	0	0	0,01	0	-0,05	0,04	0,01	0,01	0,01	RF
RBF	0	-0,03	-0,05	0	0	-0,04	0,01	-0,03	-0,04	-0,04	RBF
J48	0	-0,01	0,02	0,01	0,01	-0,01	0,05	0,09	0,03	0,06	J48
LIB_SVM	0	0,02	0,02	-0,12	-0,13	-0,06	0,04	0,03	-0,2	-0,24	LIB_SVM
D-U/RS/0.1											D-U/SS/0.1
LR	0	0,02	0,01	0	0	-0,02	0,02	0,01	0	0,01	LR
SVM	0	0	0,01	0	0	0,02	0,06	0,06	0,03	0,02	SVM
KNN1	0	0	0	0	0,01	-0,07	0,06	-0,02	-0,03	-0,04	KNN1
KNN3	0	0,01	0,01	0	-0,01	-0,09	0,05	0	-0,06	-0,04	KNN3
NB	0	0,02	0,02	0,01	0,01	0,01	0,03	0	0,03	0,01	NB
BN	0	-0,02	-0,04	0	0	0	0,03	0,02	0,06	0,05	BN
RF	0	0	-0,01	-0,02	-0,01	-0,03	0,06	0,01	0	0,01	RF
RBF	0	-0,03	-0,02	0	0	-0,02	0,02	0	-0,02	-0,02	RBF
J48	0	-0,01	-0,01	0,02	0,03	-0,02	0,09	0,1	0,03	0,02	J48
LIB_SVM	0	0,01	0,02	-0,14	-0,17	-0,04	0,04	0,04	-0,12	-0,16	LIB_SVM

Figure 4.10 : Citeseer directed-undirected comparison.



D-U/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End	CO	LO-Start	LO-End	ICA-Start	ICA-End	D-U/SS/0.9
LR	0	0	0,01	0,01	0,03	0,03	0,05	0,05	0,01	0,02	LR
SVM	0	0,01	0,01	-0,02	-0,02	0,03	0,11	0,16	0,02	0,02	SVM
KNN1	0	0,02	-0,01	0	0	-0,01	0,08	0,15	-0,03	-0,04	KNN1
KNN3	0	0,03	0,01	-0,01	0	0,01	0,09	0,16	0,01	0	KNN3
NB	0	0,01	0	-0,06	-0,04	0,11	0,06	0,05	0,1	0,11	NB
BN	0	0,04	0,07	0,03	0,02	0,14	0,05	0,04	0,18	0,1	BN
RF	0	0,03	0,09	0,01	0,02	0,05	0,09	0,13	0,07	0,05	RF
RBF	0	-0,03	-0,07	0,01	0	0,03	0	0,02	0,03	0,04	RBF
J48	0	0,03	0,05	0,1	0,15	0,12	0,05	0,04	0,24	0,24	J48
LIB_SVM	0	0,06	0,05	0	0	0	0,13	0,14	0	0	LIB_SVM
D-U/RS/0.8											D-U/SS/0.8
LR	0	-0,07	-0,09	0,04	0,06	0	0,03	0,04	0,05	0,09	LR
SVM	0	0,03	0,05	0,01	0	0,03	0,08	0,07	0,04	0,02	SVM
KNN1	0	0,04	0,05	-0,01	0	0	0,07	0,12	0	-0,01	KNN1
KNN3	0	0,06	0,09	-0,01	-0,01	0	0,08	0,18	0,01	0	KNN3
NB	0	0,01	0,01	-0,07	-0,04	0,1	0,1	0,06	0,1	0,01	NB
BN	0	0,06	0,09	0	0,01	0,07	0,03	0,01	0,09	0,06	BN
RF	0	0,01	0,03	-0,03	-0,02	0,04	0,08	0,13	0,02	0	RF
RBF	0	0	-0,02	0	0,01	0,04	0,04	0,05	0,03	0,03	RBF
J48	0	0,01	0,05	0,02	0,04	0,09	0,08	0,07	0,1	0,08	J48
LIB_SVM	0	0,04	0,04	0	0	0,07	0,06	0,05	0,06	0,06	LIB_SVM
D-U/RS/0.7											D-U/SS/0.7
LR	0	-0,04	-0,04	0,02	0,04	0,05	0,08	0,08	0,1	0,15	LR
SVM	0	0,02	0,02	0,01	0,02	0,05	0,1	0,09	0,07	0,07	SVM
KNN1	0	0,06	0,06	-0,01	0	0	0,1	0,16	0,02	0,01	KNN1
KNN3	0	0,06	0,09	-0,01	-0,01	-0,01	0,13	0,1	0	-0,01	KNN3
NB	0	0,04	0,06	-0,01	0,02	0,07	0,13	0,12	0,11	0,07	NB
BN	0	0,02	0,06	-0,02	0	0,08	0,02	0,06	0,05	0,01	BN
RF	0	0,03	0,07	0	0,02	0,02	0,08	0,11	0,03	0,02	RF
RBF	0	-0,03	-0,04	0	0	0,02	0,07	0,11	0,02	0,02	RBF
J48	0	-0,01	0,01	0,04	0,08	0,07	0,06	0,08	0,15	0,14	J48
LIB_SVM	0	0,01	0	-0,01	-0,01	0,03	0,05	0,07	-0,01	-0,01	LIB_SVM
D-U/RS/0.6											D-U/SS/0.6
LR	0	-0,06	-0,07	0,03	0,03	0,03	0,07	0,09	0,04	0,07	LR
SVM	0	0,04	0,07	0,02	0,02	0,04	0,11	0,14	0,07	0,07	SVM
KNN1	0	0,04	0,05	-0,01	-0,01	-0,01	0,11	0,14	0	-0,01	KNN1
KNN3	0	0,04	0,07	0	0	0	0,17	0,18	0,01	0	KNN3
NB	0	0,04	0,07	0,01	0,04	0,04	0,16	0,15	0,13	0,11	NB
BN	0	0,03	0,06	-0,03	-0,01	0,05	0,07	0,09	0,05	0,03	BN
RF	0	0,03	0,06	-0,02	-0,01	0,02	0,11	0,16	-0,01	-0,02	RF
RBF	0	-0,03	-0,03	0,01	0	0,01	0,09	0,15	0,01	0,01	RBF
J48	0	-0,01	0,01	-0,03	-0,05	0,03	0,07	0,06	0,18	0,16	J48
LIB_SVM	0	0	0	-0,04	-0,05	-0,03	0,06	0,04	-0,09	-0,08	LIB_SVM
D-U/RS/0.5											D-U/SS/0.5
LR	0	-0,04	-0,05	-0,03	-0,02	-0,01	0,01	0,02	0,02	0,02	LR
SVM	0	0,02	0,04	0,02	0,03	0,01	0,09	0,14	0,03	0,03	SVM
KNN1	0	0,03	0,04	-0,01	-0,01	-0,02	0,12	0,15	-0,01	-0,02	KNN1
KNN3	0	0,05	0,1	-0,01	0	-0,04	0,14	0,12	-0,01	-0,01	KNN3
NB	0	0,03	0,06	0	0,02	0,02	0,12	0,14	0,1	0,08	NB
BN	0	0,03	0,04	-0,03	0	0,02	0,1	0,16	0,04	0,04	BN
RF	0	0,01	0,01	-0,02	-0,03	-0,03	0,08	0,13	0,01	0,02	RF
RBF	0	0	0	0	0	0	0,07	0,12	0	0	RBF
J48	0	0	0,02	0	-0,03	-0,01	0,07	0,06	0,1	0,09	J48
LIB_SVM	0	0,01	0,01	-0,08	-0,1	-0,03	0,05	0,03	-0,1	-0,11	LIB_SVM
D-U/RS/0.4											D-U/SS/0.4
LR	0	-0,03	-0,02	-0,02	0	-0,03	-0,03	-0,05	0	0,01	LR
SVM	0	0,04	0,08	0,02	0,03	-0,02	0,05	0,11	0,01	0	SVM
KNN1	0	0,03	0,07	0	0	-0,06	0,09	0,09	-0,06	-0,06	KNN1
KNN3	0	0,05	0,1	-0,01	0	-0,08	0,13	0,09	-0,05	-0,06	KNN3
NB	0	0,03	0,06	-0,01	0,02	0	0,1	0,11	0,03	0,02	NB
BN	0	0,03	0,05	-0,03	0	-0,01	0,08	0,12	0	0,01	BN
RF	0	0,02	0,02	0,01	0,02	-0,04	0,05	0,1	-0,05	-0,08	RF
RBF	0	-0,02	-0,02	0	0	-0,03	0,02	0,03	-0,03	-0,03	RBF
J48	0	-0,02	0,01	0	-0,02	-0,03	0,04	0,07	0,1	0,1	J48
LIB_SVM	0	0,02	0,02	-0,08	-0,1	-0,03	0,04	0,06	-0,12	-0,16	LIB_SVM
D-U/RS/0.3											D-U/SS/0.3
LR	0	-0,01	0	0,04	0,04	-0,06	-0,1	-0,14	-0,02	-0,02	LR
SVM	0	0,03	0,06	0,03	0,04	-0,05	0,03	0,05	0	-0,01	SVM
KNN1	0	0,04	0,08	0	-0,01	-0,09	0,08	0,07	-0,08	-0,09	KNN1
KNN3	0	0,05	0,1	-0,01	0	-0,1	0,09	0,06	-0,08	-0,08	KNN3
NB	0	0,04	0,04	0	0,01	-0,01	0,07	0,06	0,02	0,02	NB
BN	0	0,02	0,02	-0,01	-0,01	-0,02	0,05	0,05	-0,03	-0,01	BN
RF	0	0,03	0,05	-0,01	0,02	-0,07	0,03	0,08	-0,07	-0,06	RF
RBF	0	0	0,01	0	0	-0,03	0,01	-0,03	-0,03	-0,04	RBF
J48	0	0,02	0,06	0	0,01	-0,06	-0,02	0	0,04	0,02	J48
LIB_SVM	0	0,02	0,02	-0,08	-0,11	-0,02	0	-0,01	-0,15	-0,2	LIB_SVM
D-U/RS/0.2											D-U/SS/0.2
LR	0	-0,01	0	0,04	0,05	-0,03	-0,1	-0,11	-0,03	-0,03	LR
SVM	0	0	0,03	0,04	0,04	-0,03	-0,02	-0,01	0,01	0,01	SVM
KNN1	0	0,02	0,04	-0,01	-0,02	-0,07	0,05	0,06	-0,06	-0,06	KNN1
KNN3	0	0,03	0,06	-0,01	-0,02	-0,09	0,06	0,05	-0,04	-0,05	KNN3
NB	0	0,02	0,04	0,01	0,01	0,01	0,04	0,02	0,02	0,03	NB
BN	0	0,01	0,01	-0,01	-0,01	0	-0,02	-0,03	-0,03	-0,01	BN
RF	0	0	0	0,01	0,01	-0,04	0,02	0,04	-0,02	0,01	RF
RBF	0	0,01	0,02	0,01	0	-0,02	-0,04	-0,08	-0,03	-0,03	RBF
J48	0	-0,01	0,01	0	0,01	-0,04	-0,02	0,05	0	-0,03	J48
LIB_SVM	0	0,02	0,03	-0,06	-0,08	-0,01	0	0,02	-0,15	-0,18	LIB_SVM
D-U/RS/0.1											D-U/SS/0.1
LR	0	-0,02	-0,01	-0,01	-0,01	0	-0,05	-0,05	-0,04	-0,04	LR
SVM	0	0	0,01	0,05	0,05	-0,03	0,03	0,03	0,03	0,04	SVM
KNN1	0	0	0	-0,02	-0,02	-0,05	0,06	0,03	-0,05	-0,04	KNN1
KNN3	0	0,01	0,02	-0,01	-0,01	-0,07	0,07	0,04	-0,03	-0,04	KNN3
NB	0	0,03	0,06	0,01	0,02	0,02	0,02	0,01	0,04	0,05	NB
BN	0	-0,01	0,03	0	0	-0,01	0,01	0,05	0	0,01	BN
RF	0	0	0,02	0,01	0,02	-0,04	0,03	0,03	-0,01	0,01	RF
RBF	0	-0,02	0	0	0	-0,03	0	-0,05	-0,03	-0,04	RBF
J48	0	-0,01	0,02	-0,02	0	-0,01	0,02	0,02	0	0,02	J48
LIB_SVM	0	0	0	-0,05	-0,06	0,01	0,03	0,05	-0,08	-0,1	LIB_SVM

Figure 4.11 : Cora directed-undirected comparison.

D-U/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End
LR	0	0	-0,02	-0,02	-0,01
SVM	0	0,03	0,05	-0,01	0
KNN1	0	0,07	0,08	0	0
KNN3	0	0,06	0,08	0	0
NB	0	0,04	0,04	0	-0,01
BN	0	0,02	0,03	0	0,01
RF	0	0,07	0,08	0	0
RBF	0	0,01	0,01	0	0
J48	0	0,08	0,09	0,01	0,01
LIB_SVM	0	0,04	0,04	0	0

CO	LO-Start	LO-End	ICA-Start	ICA-End	D-U/SS/0.9
0	-0,06	-0,07	0	0	LR
0,07	0,07	0,13	0,06	0,07	SVM
-0,01	0,03	0	0	0	KNN1
-0,01	0,01	0,01	-0,01	-0,01	KNN3
0,01	0,02	0,02	-0,03	-0,02	NB
-0,05	-0,02	0	-0,05	-0,05	BN
0,02	0,05	0,07	0,03	0,04	RF
0,01	0,04	0,05	0,02	0,02	RBF
-0,02	0,1	0,09	0	0	J48
-0,01	-0,03	-0,04	-0,01	-0,01	LIB_SVM

D-U/RS/0.8					
LR	0	0	0	0,01	0,01
SVM	0	0,05	0,06	-0,01	0
KNN1	0	0,03	0,02	0	0
KNN3	0	0,06	0,04	0	0
NB	0	0,03	-0,01	0,01	0,01
BN	0	0,04	0,04	0,02	0,02
RF	0	0,01	0,04	-0,01	-0,01
RBF	0	-0,03	-0,05	0	0
J48	0	0,03	0,05	0,01	0
LIB_SVM	0	0,02	0,03	0	0

					D-U/SS/0.8
0	-0,06	-0,06	-0,04	-0,04	LR
-0,02	0,07	0,08	-0,02	-0,01	SVM
-0,01	0,02	0,01	-0,01	-0,01	KNN1
0	0,07	0,08	0	-0,01	KNN3
0,03	-0,08	-0,06	-0,01	0	NB
-0,04	0,09	0,1	-0,04	-0,02	BN
-0,03	0,08	0,07	-0,06	-0,06	RF
-0,03	0,03	0,03	-0,04	-0,03	RBF
-0,03	0,07	0,1	0	0,01	J48
0	0,06	0,06	-0,01	0	LIB_SVM

D-U/RS/0.7					
LR	0	0	-0,03	-0,02	-0,02
SVM	0	0,05	0,07	-0,01	0
KNN1	0	0	0,02	0	0
KNN3	0	-0,01	0,03	0	0
NB	0	0	0	-0,01	0
BN	0	0,07	0,07	0,02	0,03
RF	0	0	0,05	0,03	0,03
RBF	0	0	-0,01	0	0
J48	0	0,04	0,06	0,03	0,04
LIB_SVM	0	0,06	0,07	-0,01	0

					D-U/SS/0.7
-0,03	-0,02	-0,07	0	-0,01	LR
0,03	0	0,02	0,03	0,03	SVM
0,02	-0,01	-0,01	0,02	0,02	KNN1
0,04	-0,01	-0,03	0,04	0,04	KNN3
0,04	-0,02	-0,06	0,02	0,04	NB
0,03	0,1	0,09	0,02	0,03	BN
-0,04	-0,01	0,03	-0,06	-0,06	RF
-0,02	0,04	0,05	-0,03	-0,02	RBF
0	0,02	0,03	0,04	0,05	J48
-0,06	0,11	0,12	-0,06	-0,07	LIB_SVM

D-U/RS/0.6					
LR	0	0	-0,01	0,01	0,02
SVM	0	0,05	0,05	0	0
KNN1	0	0,01	0,03	0	0
KNN3	0	0	0,03	0	0
NB	0	-0,02	-0,03	-0,02	-0,01
BN	0	0,06	0,06	0,03	0,02
RF	0	0	0,04	0	0,01
RBF	0	0,04	0,04	0	0
J48	0	0,02	0,05	0,01	0
LIB_SVM	0	0,04	0,04	0	-0,01

					D-U/SS/0.6
0,01	0,06	0,06	0,02	0,02	LR
-0,01	0,06	0,02	-0,01	-0,01	SVM
0,02	0,02	0,02	0,03	0,02	KNN1
0,03	0,03	0,02	0,02	0,03	KNN3
-0,03	-0,01	-0,02	-0,05	-0,02	NB
-0,04	0,03	0,07	-0,02	-0,03	BN
-0,02	0	0,03	-0,02	-0,02	RF
-0,04	0,04	0,03	-0,04	-0,04	RBF
0,02	0,04	0,07	-0,01	-0,02	J48
0	0,09	0,08	0	0	LIB_SVM

D-U/RS/0.5					
LR	0	0,01	-0,02	0,02	0,02
SVM	0	0,06	0,1	0	0
KNN1	0	0,01	0,01	0	0
KNN3	0	0,02	0,02	0,01	0
NB	0	-0,01	-0,03	-0,01	0,01
BN	0	-0,01	-0,02	0,01	0,01
RF	0	0,03	0,03	0,03	0,02
RBF	0	0,02	0,01	0	0,01
J48	0	0,01	0,02	-0,01	-0,01
LIB_SVM	0	0,04	0,04	-0,01	-0,01

					D-U/SS/0.5
0	0,01	0,02	0,06	0,04	LR
0	0,04	0,05	0,01	0,01	SVM
-0,07	0,02	0,03	-0,02	-0,03	KNN1
0	0,04	0,06	0	0	KNN3
0,01	0,01	-0,02	-0,02	0,01	NB
-0,07	0,02	0,02	-0,05	-0,06	BN
-0,03	0,05	0,06	0	-0,01	RF
-0,05	0,01	0	-0,05	-0,05	RBF
0,04	0,02	0,08	0,02	0,01	J48
-0,03	0,12	0,15	-0,03	-0,03	LIB_SVM

D-U/RS/0.4					
LR	0	0,02	0,02	0,02	0,01
SVM	0	0,05	0,08	0	0
KNN1	0	0,02	0,01	0,03	0
KNN3	0	0,03	0,01	0,01	0
NB	0	-0,01	-0,03	-0,01	0,01
BN	0	-0,07	-0,11	-0,01	-0,01
RF	0	0,02	0,02	0,01	0,01
RBF	0	0,04	0,02	0	0
J48	0	0,02	0,03	0,03	0,03
LIB_SVM	0	0,03	0,03	0	0

					D-U/SS/0.4
0,02	0	0,02	0,04	0,05	LR
0,01	0,06	0,03	0,01	0,01	SVM
-0,04	0,02	0,01	0,01	-0,01	KNN1
0,02	0	0,02	0,04	0,04	KNN3
0,04	-0,03	-0,03	0,04	0,04	NB
0	-0,1	-0,12	0,02	0	BN
-0,02	0,03	0,07	-0,02	-0,02	RF
0	0,04	0,02	0	0	RBF
-0,02	0,02	0,04	0,02	0,03	J48
-0,02	0,07	0,08	-0,02	-0,03	LIB_SVM

D-U/RS/0.3					
LR	0	0,02	-0,01	0,02	0,02
SVM	0	0,02	0,05	0	0
KNN1	0	0,01	0	0,03	0
KNN3	0	0,01	0	0,01	0
NB	0	-0,03	-0,05	0	0
BN	0	-0,08	-0,09	0,01	0,01
RF	0	0,04	0,05	-0,02	-0,01
RBF	0	0,02	0,01	0	0
J48	0	0,03	0,03	0,02	0,01
LIB_SVM	0	0,03	0,04	0	0

					D-U/SS/0.3
0,04	0,04	0,03	0,06	0,02	LR
0,06	0,09	0,02	0,07	0,07	SVM
-0,02	0,03	0,01	0	0	KNN1
0,09	0,01	0,02	0,08	0,08	KNN3
0,04	-0,01	-0,03	0,04	0,04	NB
0	-0,07	-0,08	0,01	0,01	BN
-0,02	0,04	0,08	0,04	0,05	RF
0,06	0,04	0,04	0,06	0,06	RBF
-0,01	0,05	0,03	0,04	0,04	J48
0,01	0,05	0,08	0,01	0,01	LIB_SVM

D-U/RS/0.2					
LR	0	-0,01	-0,01	0,01	0,01
SVM	0	0,02	0,02	0	0
KNN1	0	-0,01	-0,02	0	-0,01
KNN3	0	0	0	0	0
NB	0	-0,02	-0,02	0	-0,01
BN	0	-0,06	-0,1	0,01	0,01
RF	0	0,02	0,01	0,04	0,04
RBF	0	0,06	0,02	0	0
J48	0	0,03	0,04	0,01	-0,01
LIB_SVM	0	0,03	0,03	0	-0,01

					D-U/SS/0.2
0,05	0,03	0,01	0,06	0,05	LR
0,09	0,02	0,01	0,08	0,09	SVM
0,01	0	0	0	0	KNN1
0,04	0,01	0,04	0,03	0,03	KNN3
0,12	0,01	-0,02	0,11	0,12	NB
0,05	-0,06	-0,11	0,07	0,07	BN
0,02	0,02	0,06	0,03	0,03	RF
-0,01	0,05	0,03	-0,02	-0,02	RBF
0,03	0,05	0,09	0,08	0,07	J48
-0,04	0,07	0,09	-0,03	-0,04	LIB_SVM

D-U/RS/0.1					
LR	0	0	0,01	-0,02	-0,02
SVM	0	0,03	0,03	0	0
KNN1	0	0	0,01	0	0
KNN3	0	-0,02	0,01	0	0
NB	0	-0,01	-0,01	-0,02	-0,02
BN	0	-0,05	-0,03	0,01	0,01
RF	0	0,02	0,06	0	0,01
RBF	0	-0,02	-0,03	0	0
J48	0	0,02	0,06	0,02	0
LIB_SVM	0	0,05	0,06	-0,01	0

					D-U/SS/0.1
-0,06	0,04	0,01	0,03	0	LR
0,02	0,06	0,04	0,02	0,02	SVM
-0,02	0,01	0,02	-0,02	-0,02	KNN1
0,03	0,02	0	0,01	0,03	KNN3
0,06	0,04	-0,04	0,06	0,06	NB
0,02	-0,02	-0,12	0,02	0,02	BN
-0,02	0,02	0,06	-0,02	-0,01	RF
-0,01	0,04	0,02	-0,01	-0,01	RBF
0	0,03	0,12	0,02	-0,01	J48
0,07	0,05	0,05	0,08	0,07	LIB_SVM

Figure 4.12 : WebKb directed-undirected comparison.

D-U/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End	CO	LO-Start	LO-End	ICA-Start	ICA-End	D-U/SS/0.9
LR	0	0,02	0,04	-0,02	-0,02	0,02	0,01	0,01	-0,01	0,01	LR
SVM	0	0	-0,01	0	0	-0,05	0,01	0,02	-0,06	-0,05	SVM
KNN1	0	0,01	0,03	-0,01	0	-0,02	0,04	0,07	-0,02	-0,02	KNN1
KNN3	0	0	0,02	0	0,01	-0,04	0,02	0,02	-0,03	-0,04	KNN3
NB	0	-0,01	0	-0,03	-0,02	0,02	0,05	0,05	0,02	0,03	NB
BN	0	0,02	0,02	-0,01	0,01	-0,05	0,05	0,07	0,05	0,07	BN
RF	0	0,02	0,03	0,01	0	0	0,05	0,09	0,01	0,01	RF
RBF	0	0	0,01	0	0	0	0,02	-0,01	0,01	0,02	RBF
J48	0	0,01	0,01	0,02	0,04	-0,01	0,05	0,06	0,06	0,09	J48
LIB_SVM	0	0,02	0,03	0	0	0,01	0,05	0,07	0,01	0,01	LIB_SVM
D-U/RS/0.8											D-U/SS/0.8
LR	0	0,04	0,07	0,01	0,01	0	0,04	0,07	0,02	0,02	LR
SVM	0	0,01	0	0	0	0,01	0,02	0,02	-0,03	-0,04	SVM
KNN1	0	-0,03	-0,03	0	-0,01	-0,02	0,09	0,17	-0,02	-0,02	KNN1
KNN3	0	0	0,03	0	0	-0,03	0,06	0,09	-0,02	-0,02	KNN3
NB	0	-0,06	-0,05	-0,02	0	0,01	0,11	0,1	0,12	0,13	NB
BN	0	0,01	0,03	-0,01	0	0,08	0,15	0,19	0,2	0,26	BN
RF	0	0	0,04	0	0	0	0,13	0,17	0,01	0	RF
RBF	0	-0,01	-0,04	0	0	0	0,05	0,03	0	0	RBF
J48	0	0,04	0,07	0,04	0,06	-0,01	0,07	0,1	0,06	0,09	J48
LIB_SVM	0	0	0	0	-0,01	-0,06	0,09	0,07	-0,06	-0,06	LIB_SVM
D-U/RS/0.7											D-U/SS/0.7
LR	0	0	0,01	-0,01	-0,01	0,04	0,1	0,12	0,02	0,01	LR
SVM	0	0	-0,01	0	-0,01	0,05	0,04	0,06	0,03	0,01	SVM
KNN1	0	-0,01	-0,01	0	0	0,01	0,12	0,2	0,01	0,02	KNN1
KNN3	0	-0,01	-0,01	-0,01	-0,01	0,01	0,1	0,15	0,01	0,01	KNN3
NB	0	-0,04	-0,02	-0,02	-0,01	0,02	0,13	0,16	0,18	0,24	NB
BN	0	0,03	0,09	0	0,02	0,08	0,19	0,21	0,25	0,33	BN
RF	0	0	0	-0,01	-0,01	0,04	0,12	0,17	0,04	0,03	RF
RBF	0	0,03	0,05	0	0	-0,02	0,03	0,05	-0,01	-0,02	RBF
J48	0	0,01	0,02	0,03	0,03	0,01	0,12	0,16	0,13	0,16	J48
LIB_SVM	0	0,02	0,02	-0,02	-0,01	-0,06	0,16	0,16	-0,06	-0,06	LIB_SVM
D-U/RS/0.6											D-U/SS/0.6
LR	0	0,01	0,02	0	0	0,01	0,12	0,17	0,06	0,12	LR
SVM	0	0,01	0	0,01	0	0,02	0,08	0,1	0,01	0,05	SVM
KNN1	0	-0,01	0	-0,01	-0,01	0,01	0,13	0,15	0,01	0,01	KNN1
KNN3	0	-0,01	0	0	0	0	0,13	0,18	0	0	KNN3
NB	0	0	0,01	-0,01	0	0,01	0,12	0,1	0,19	0,24	NB
BN	0	0,02	0,04	0,01	0,02	0,06	0,23	0,34	0,2	0,28	BN
RF	0	0,02	0,03	0,01	0	-0,01	0,14	0,15	0,04	0,02	RF
RBF	0	0	-0,02	0	0	-0,01	0,09	0,12	-0,01	-0,01	RBF
J48	0	-0,01	-0,02	-0,01	0	0	0,14	0,2	0,12	0,17	J48
LIB_SVM	0	0,01	0,01	-0,06	-0,07	-0,08	0,06	0,03	-0,07	-0,07	LIB_SVM
D-U/RS/0.5											D-U/SS/0.5
LR	0	0	0,01	0	0,01	-0,01	0,08	0,08	0,01	0	LR
SVM	0	-0,02	-0,04	0	0	-0,02	0	-0,07	-0,05	-0,06	SVM
KNN1	0	-0,01	-0,01	-0,01	-0,01	-0,03	0,08	0,12	-0,02	-0,03	KNN1
KNN3	0	-0,02	-0,01	0	0	-0,04	0,07	0,11	-0,04	-0,04	KNN3
NB	0	0	0	0	0	0,01	0,08	0,08	0,06	0,09	NB
BN	0	0,02	0,05	-0,01	0,01	-0,01	0,16	0,36	0,11	0,16	BN
RF	0	0	-0,01	0,02	0,02	-0,06	0,17	0,23	0,02	0	RF
RBF	0	0	-0,03	0	0,01	-0,02	0,1	0,11	-0,03	-0,03	RBF
J48	0	-0,01	-0,03	0	0	-0,02	0,22	0,26	0,13	0,17	J48
LIB_SVM	0	0,04	0,04	-0,06	-0,06	-0,14	0,11	0,11	-0,14	-0,14	LIB_SVM
D-U/RS/0.4											D-U/SS/0.4
LR	0	0	0	0,01	0,02	-0,02	0,07	0,04	-0,04	-0,09	LR
SVM	0	-0,01	-0,01	0,01	0	-0,06	-0,03	-0,13	-0,07	-0,06	SVM
KNN1	0	0	-0,01	-0,01	-0,01	-0,05	0,07	0,08	-0,04	-0,05	KNN1
KNN3	0	-0,01	0	-0,01	-0,01	-0,07	0,05	0,08	-0,07	-0,07	KNN3
NB	0	-0,02	-0,01	0	-0,01	0,02	0,09	0,15	0,08	0,13	NB
BN	0	0,02	0,04	-0,01	0	-0,02	0,2	0,27	0,14	0,19	BN
RF	0	0	0,02	0	0,01	-0,08	0,14	0,22	-0,02	-0,02	RF
RBF	0	0	-0,03	0	0,01	0,01	0,11	0,1	0	0,01	RBF
J48	0	-0,04	-0,02	0	0	-0,04	0,18	0,28	0,14	0,14	J48
LIB_SVM	0	0,01	0,01	-0,08	-0,09	-0,19	0,14	0,19	-0,27	-0,3	LIB_SVM
D-U/RS/0.3											D-U/SS/0.3
LR	0	0,01	0,01	-0,01	-0,01	0,01	0,05	0,04	0,03	0,02	LR
SVM	0	-0,01	-0,01	0	0	-0,05	-0,01	-0,05	-0,05	-0,05	SVM
KNN1	0	-0,01	0	-0,01	-0,02	-0,02	0,02	0,03	-0,03	-0,03	KNN1
KNN3	0	-0,02	0,01	-0,01	-0,01	-0,03	0,01	0,05	-0,04	-0,03	KNN3
NB	0	0	0,01	-0,01	0,01	0,02	0,01	0,07	0,02	0,04	NB
BN	0	0,01	0,03	0	0,01	-0,02	0,15	0,25	0,1	0,15	BN
RF	0	0,01	0,01	0	0,01	-0,06	0,08	0,12	-0,03	-0,04	RF
RBF	0	0	-0,02	0	0,01	0	0,12	0,11	0	0	RBF
J48	0	-0,03	-0,03	0	0,01	-0,03	0,11	0,24	0,08	0,14	J48
LIB_SVM	0	0,01	0	-0,14	-0,15	-0,19	0,04	0,02	-0,28	-0,32	LIB_SVM
D-U/RS/0.2											D-U/SS/0.2
LR	0	0,01	-0,01	0	0	0	0,05	0,01	0,01	-0,01	LR
SVM	0	-0,01	0	-0,01	-0,01	-0,04	0,02	-0,01	-0,04	-0,04	SVM
KNN1	0	-0,01	0,01	-0,01	-0,02	0	0,02	-0,01	0	-0,01	KNN1
KNN3	0	-0,02	-0,03	-0,01	-0,01	-0,01	0	0,01	-0,01	-0,03	KNN3
NB	0	-0,09	-0,08	0,01	0,01	0,01	0,02	0,03	-0,01	-0,02	NB
BN	0	0,05	0,07	0,01	0,01	-0,02	0,09	0,16	0,04	0,04	BN
RF	0	-0,01	-0,01	0	0	-0,03	0,05	0,08	-0,01	-0,02	RF
RBF	0	-0,02	-0,05	0,02	0,02	-0,03	0,11	0,12	-0,03	-0,04	RBF
J48	0	-0,01	0,01	0,01	0,01	-0,01	0,1	0,14	0,03	0,03	J48
LIB_SVM	0	0,01	0,01	-0,05	-0,06	0	0,06	0,04	-0,11	-0,14	LIB_SVM
D-U/RS/0.1											D-U/SS/0.1
LR	0	0,01	0,01	-0,01	0	0,01	0,1	0,09	0,02	0	LR
SVM	0	0	-0,01	0	0	0	0,09	0,08	0,01	0,01	SVM
KNN1	0	0	0,01	-0,01	-0,01	0,01	0,1	0,06	0	0,01	KNN1
KNN3	0	0	-0,01	-0,01	-0,01	-0,01	0,09	0,08	-0,01	-0,02	KNN3
NB	0	-0,04	-0,04	0,02	0,02	0,05	0,07	0,09	0,04	0,03	NB
BN	0	0,04	0,03	-0,01	0	0	0,09	0,18	0,05	0,04	BN
RF	0	0,01	0,02	0,03	0,02	0	0,08	0,09	0,03	0,02	RF
RBF	0	0	-0,01	0,02	0,02	-0,03	0,12	0,11	-0,01	-0,01	RBF
J48	0	0,01	0,01	-0,01	-0,01	0,02	0,09	0,1	0,09	0,16	J48
LIB_SVM	0	0	0	-0,04	-0,04	-0,02	0,1	0,06	-0,04	-0,07	LIB_SVM

Figure 4.13 : HepTh directed-undirected comparison.

5. FEATURE ENRICHMENT AND SELECTION FOR COLLECTIVE CLASSIFICATION

5.1 Background and Purpose

In this section, we introduce a new method of transductive network classification which can use the test node features when training the classifier (Cataltepe et al. 2014). We train our classifier using enriched node features. The enriched node features include, in addition to the node's own features, the aggregated neighbors' features and aggregation of node and neighbor features passed through simple logical operators OR and AND. Enriched features may contain irrelevant or redundant features, which could decrease classifier performance. Therefore, we employ feature selection to determine whether a feature among the set of enriched features should be used for classifier training or not. Our feature selection method, called FCBF#, is a mutual information based, filter type, fast, feature selection method (Senliol, Gulgezen, Yu, & Cataltepe 2008).

The methods introduced in the context of this section of the thesis, is an extension of previous work on feature enrichment, which was first introduced in (Senliol 2010). In the previous work, enriched features are constructed as combinations of plain (node's own features), neighbors features and ORed features as defined in Equation 5.6. Here, neighbors features are calculated using equation 5.1, while ORed features are calculated with equation 5.3. In addition to those, we also introduce ANDed features as defined in equation 5.2 and take into account all major combinations of those enrichment methods as separate cases as defined in equations 5.4, 5.5, 5.6 and 5.7. It only uses EnrSel method, in which, after enrichment of all node features, feature selection is applied. However we also introduce SelEnr method, in which, feature selection among the node features is done and then those features are enriched. In addition to the Logistic Regression, we also experiment with the Bayes Net as the base classifier.

Experimental results on three different network datasets show that classification accuracies obtained using network enriched and selected features are comparable or better than content only or collective classification.

5.2 Methodology

5.2.1 Enriched features for classification of networked data

Networked data offer not only the features of a node but also features of neighbors of a node. Our enriched features aim to make the best use of the neighbor features for classification. Enrichment process uses not only the nodes in the training data, but also features of nodes in the test data, enabling it to make use of any available test node features. We redefine instances in a dataset using the new set of enriched features so that the same type of classifiers used for node content only features can also be used for enriched features.

Just like the aggregated neighbor labels of a node (Equation 1.3), we aggregate feature vectors of the neighbors of a node u to form the *neighbor features vector* $\mathbf{x}_N(u) \in \mathcal{N}^d$:

$$\mathbf{x}_N(u) = \sum_{v \in N(u)} \mathbf{x}(v). \tag{5.1}$$

For some classification problems, features which are common between a node and its neighbors may indicate a stronger class membership. These features are aggregated to form the *neighbor features AND vector*:

$$\mathbf{x}_{\&}(u) = \sum_{v \in N(u)} \mathbf{x}(u) \& \mathbf{x}(v), \tag{5.2}$$

where $\mathbf{x}_{\&}(u) \in \mathcal{N}^d$ and & denotes the bitwise logical AND of the feature vectors $\mathbf{x}(u)$ and $\mathbf{x}(v)$.

For classification problems where only a small portion of a large number of features are observed (for example, words for a web page classification problem), whether a feature appears on the node or one of its neighbors, it may help with classification and

should be taken into consideration. These features are aggregated to form the *neighbor features OR vector*:

$$\mathbf{x}_{|}(u) = \sum_{v \in N(u)} \mathbf{x}(u) | \mathbf{x}(v), \tag{5.3}$$

where $\mathbf{x}_{|}(u) \in \mathcal{N}^d$ and $|$ denotes the bitwise logical OR of the feature vectors $\mathbf{x}(u)$ and $\mathbf{x}(v)$.

We also use enriched feature vectors which include the node features and different combinations of enrichment.

$$\mathbf{x}_{xN}(u) = [\mathbf{x}(u)\ \mathbf{x}_N(u)]. \tag{5.4}$$

$$\mathbf{x}_{xN\&}(u) = [\mathbf{x}(u)\ \mathbf{x}_N(u)\ \mathbf{x}_{\&}(u)]. \tag{5.5}$$

$$\mathbf{x}_{xN|}(u) = [\mathbf{x}(u)\ \mathbf{x}_N(u)\ \mathbf{x}_{|}(u)]. \tag{5.6}$$

$$\mathbf{x}_{xN\&|}(u) = [\mathbf{x}(u)\ \mathbf{x}_N(u)\ \mathbf{x}_{\&}(u)\ \mathbf{x}_{|}(u)]. \tag{5.7}$$

5.2.2 Feature selection with FCBF#

Feature enrichment increases the number of features and therefore the training and classification time. It may also introduce redundant or irrelevant features. We use feature selection to get rid of those redundant or irrelevant features and hence, hopefully, have faster and more accurate classifiers. We consider two different schemes: (i) Feature selection among the node features and then enrichment of those features (SelEnr) and (ii) Enrichment of all node features and then feature selection among them (EnrSel). If EnrSel method is used, we produce $k * d$ dimensional feature vectors where k is the number of combined enrichment methods. Then we apply feature selection to obtain d dimensional feature vectors. On the other hand, when SelEnr method is used, we first apply feature selection on the original node features

to obtain d/k features. Feature enrichment is applied only on the selected features to reconstruct the new d dimensional feature vectors. As shown by the experimental results below, for high dimensional datasets (e.g. Cora, CiteSeer) SelEnr method is better, however for lower dimensional datasets (synthetic data) EnrSel may perform as well as or better than EnrSel.

As the feature selection method, we use FCBF# (Senliol et al. 2008). Previous work (Senliol et al. 2008) on many datasets showed that FCBF# produces size k feature subsets that are more accurate than FCBF. It was also shown in (Senliol et al. 2008) that while FCBF# is orders of magnitude faster than a recent information theoretical feature selection algorithm mRMR (minimum Redundancy Maximum Relevance) (Peng et al. 2005), the classifier accuracies achieved on FCBF# selected feature subsets are comparable to that of mRMR. FCBF# is inspired by the filter type feature subset selection algorithm FCBF (Fast Correlation Based Feature Selection) (Yu & Liu 2003). In terms of its feature evaluation criterion (Guyon & Elisseeff 2003), FCBF# is a supervised, information theoretical, filter type feature selection method. In terms of its search procedure (Guyon & Elisseeff 2003) FCBF# uses backward selection.

Both FCBF and FCBF# use Symmetric Uncertainty (SU) for feature evaluation:

$$SU(X,Y) = 2\frac{MI(X;Y)}{H(X)+H(Y)} \tag{5.8}$$

In this equation, X and Y are two discrete random variables, $H(.)$ is the entropy and $MI(X;Y)$ is the mutual information between X and Y (Cover, Thomas, Proakis, Salehi, & Morelos-Zaragoza 1991).

FCBF# uses SU between two features F_i and F_j, $SU(F_i,F_j)$, to measure their redundancy and SU between a feature F_i and the label R, $SU(F_i,R)$ to evaluate feature relevance. SU allows FCBF# to evaluate each feature much faster than wrapper type feature selection methods that require classifier training for each feature evaluation.

In terms of its search method, FCBF# differs from FCBF. In FCBF, a predominant (more relevant than others) feature eliminates all features whose correlation to the predominant feature are higher than their relevance. Although this is an efficient subset selection method, the final selected subset may end up being much smaller than the

best subset. FCBF# gives every feature a temporary predominance in the elimination process and lets them eliminate a single feature at each iteration, beginning with the least relevant features. After all features have their chances, elimination process starts from beginning with the remaining features.

Algorithm 3 provides the pseudo code for the FCBF# feature selection algorithm.

In Section 5.3 we compare FCBF# and mRMR in terms of the accuracies obtained on networked data.

5.2.3 Content only and collective classification of networked data using enriched and selected features

5.2.3.1 Content only classification (CO)

Once the newly constructed feature vectors via EnrSel or SelEnr methods (See Section 5.2.1), are available, any supervised content only (Macskassy & Provost 2007, Sen et al. 2008) classification method could be used to train a classifier. Given a networked dataset with labeled training nodes V_{train}, test nodes V_{test}, we perform content only classification with reconstructed enriched features as follows: First, we produce the reconstructed enriched set of features for training and test nodes. Then we project both training and test features according to the reconstructed enriched features. We train a content only classifier, $g_{CO}(.)$, on the enriched training data. Using the trained classifier, we predict the labels for the test nodes.

5.2.3.2 Collective classification (ICA)

Collective Classification algorithms, such as ICA (Iterative Classification Algorithm) (Macskassy & Provost 2007, Sen et al. 2008) were shown to perform better than content only classification for networked data. Collective classification can be employed using a classifier, $g_{COLO}(.)$, trained on the plain or enriched features, appended with the aggregated neighbor labels. In order to train the classifier, we employ all the steps followed for the content only classification. When test nodes need to be classified, we need to use collective classification, because when two linked test nodes are not yet classified, they require each others' labels to decide on their own

Algorithm 3 FCBF#.

Input: $\mathbf{f}_1, \ldots, \mathbf{f}_d$: Feature vectors,

Input: $\mathbf{r}$: Class labels vector,

Input: k: No of features to select.

Output: S: Set of selected features.

//Compute feature relevances (Equation 5.8)

for all $i \in 1, \ldots, d$ **do**

    $SU_{rel}[i] = calculateSU(\mathbf{f}_i, \mathbf{r})$

end for

//Order feature indices according to their relevance values.

$S_{dec} = sort(SU_{rel}, decreasing)$

$S_{inc} = sort(SU_{rel}, increasing)$

$S = S_{prev} = \{1, \ldots, d\}$

repeat

    $p = head(S_{dec})$ //index of the temporarily predominant feature

    while $p \neq NULL$ **do**

        $q = head(S_{inc})$

        $numEliminated = 0$

        while $q \neq NULL \ \& \ numEliminated < 1$ **do**

            //feature p could eliminate only a less relevant feature.

            if $p == q$ **then**

                break

            end if

            if $calculateSU(\mathbf{f}_p, \mathbf{f}_q) >= SU_{rel}[q]$ **then**

                $S_{prev} = S$

                $S = remove(S, q)$

                $S_{dec} = remove(SU_{dec}, q)$

                $S_{inc} = remove(SU_{inc}, q)$

                $numEliminated = 1$

            else

                $q = next(S_{inc})$

            end if

        end while

        $p = next(S_{dec})$

    end while

until $S == S_{prev} \ || \ |S| = k$

label. Collective classification methods allow classification of test nodes in a network simultaneously (Macskassy & Provost 2007).

In Algorithm 4, we describe the use of enriched and selected features for collective classification. In this algorithm, *NetworkEnrichedFeatures* computes the enriched features (Equation 5.5) for a given dataset. *ProjectFeatures* gets the selected subset of features. *AggregateNeighborLabels* adds the aggregated labels of neighbors which are in the training set. In the last line,
$ICA(G, V_{train}, V_{test}, X_{enr,train}, R_{train}, X_{enr,test}, g_{COLO})$, calls the ICA algorithm (please see (Sen et al. 2008)) to label the nodes in V_{test} using the trained classifier g_{COLO}.

Algorithm 4 Collective classification with enriched features.

Input: $G = (V, E), V_{train}, V_{test}$
Input: X_{train}, R_{train} (training inputs and outputs)
Input: X_{test} (test inputs)
Input: k (number of features to select)
Output: g_{COLO} (Classifier, see Equation 1.4)
Output: $\hat{R}_{test}$ (Classifier outputs for test nodes in V_{test})
$X = X_{train} \cup X_{test}$
if (EnrSel) then
    $X_{enr,train} = NetworkEnrichedFeatures(V_{train}, G, X)$
    $X_{enr,test} = NetworkEnrichedFeatures(V_{test}, G, X)$
    $S = FCBF\#(X_{enr,train}, R_{train}, k)$ //Selected feature indices
    $X_{enr,train} = ProjectFeatures(X_{enr,train}, S)$
    $X_{enr,test} = ProjectFeatures(X_{enr,test}, S)$
elseif (SelEnr) then
    $S = FCBF\#(X_{train}, R_{train}, k/numEnr)$ //Selected feature indices
    $X_{train} = ProjectFeatures(X_{train}, S)$
    $X_{enr,train} = NetworkEnrichedFeatures(V_{train}, G, X)$
    $X_{enr,test} = NetworkEnrichedFeatures(V_{test}, G, X)$
endif
$R_{N,train} = AggregateNeighborLabels(V_{train}, G, R_{train})$
Train classifier g_{COLO} using $[X_{enr,train} R_{N,train}], R_{train}$
$\hat{R}_{test} = ICA(G, V_{train}, V_{test}, X_{enr,train}, R_{train}, X_{enr,test}, g_{COLO})$

5.3 Experiments

In order to determine how useful reconstructed enriched features are for content only or collective classification, we performed a number of experiments on Cora, Citeseer and WebKb datasets (See Section 1.4).

5.3.1 Experimental setup

We partition all available datasets into $k = 10$ training (90%) and test (10%) sets using snowball sampling (See Section 1.2.1).

Since the feature vectors produced by enrichment are fixed dimensional vectors, any vector based classification algorithm can be used. In this study, we use the Bayes Net classifier provided by Weka (Hall et al. 2009).

5.3.2 Experimental results

In this section, we first compare the test classification accuracies obtained using FCBF# or mRMR feature selection on Cora dataset. Then we compare the two feature selection and enrichment methods, SelEnr and EnrSel, on all three datasets, using different single enrichment choices (neighbor, AND, OR) and using content only (CO) or collective (ICA) classification. We also show that the size of the training and test sets may make a difference in the test classification accuracies of different methods. We compare the test classification accuracies when different combinations of enrichment methods are used. Experiments with aggregation method of average (instead of sum), experiments to find out whether using test inputs change classification accuracies are also included.

5.3.2.1 Comparison of *FCBF#* vs mRMR for feature selection

In earlier work (Senliol et al. 2008) it has been shown that *FCBF#* features' classification accuracy is comparable to that of mRMR (minimum Redundancy Maximum Relevance) (Peng et al. 2005) feature selection, while mRMR is much slower.

In Figure 5.1, we compare the average (over 10 folds) test accuracies obtained when FCBF# or mRMR are used for feature selection when the plain features ($\mathbf{x}(u)$) or enriched features ($\mathbf{x}_{xN}(u) = [\mathbf{x}(u)\ \mathbf{x}_N(u)]$) are used to train a content only (CO) Bayes Net classifier. While both algorithms achieve the same accuracy when a large number of features are selected, FCBF# is able to reach higher accuracies and using a lot less

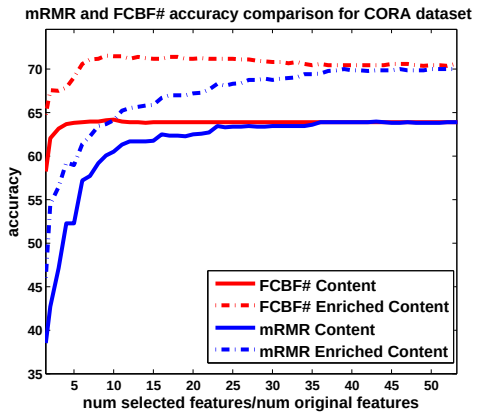


Figure 5.1 : Test accuracy comparison of mRMR and FCBF# on Cora dataset.

number of features than mRMR. In addition, FCBF# is about 40 times faster than mRMR. Therefore, for classification of networked data, both in terms of speed and accuracy, FCBF# is preferable to mRMR. For the rest of the section, we use FCBF# for feature selection.

5.3.2.2 Feature selection and enrichment

Figure 5.2 compares the average (over 10 folds) test classification accuracies obtained when enriched features are used for content only (CO) classification (Section 5.2.3.1). The horizontal axis shows the percentage of features used. We perform feature selection and then feature enrichment (SelEnr, straight lines in the figure) or feature enrichment and then feature selection (EnrSel, dotted lines in the figure). As it can be seen from the figure, OR feature enrichment (shown in red) improves the classification accuracy best, followed by the N (neighbor) feature enrichment (shown in blue) for Citeseer and Cora datasets. AND feature enrichment (shown in green) results in a performance decrease for these datasets. Feature selection and then enrichment results in better accuracy than feature enrichment and then selection, especially for small number of selected features. This could be due to the fact that the original features contain a lot of information about the class and determining the important features should be performed among the original features. However, when WebKb dataset is considered AND SelEnr enrichment method result in better accuracies among other enrichment methods, but still more than using just the plain features. This could be a consequence of the low homophily values for the WebKb dataset, compared to the

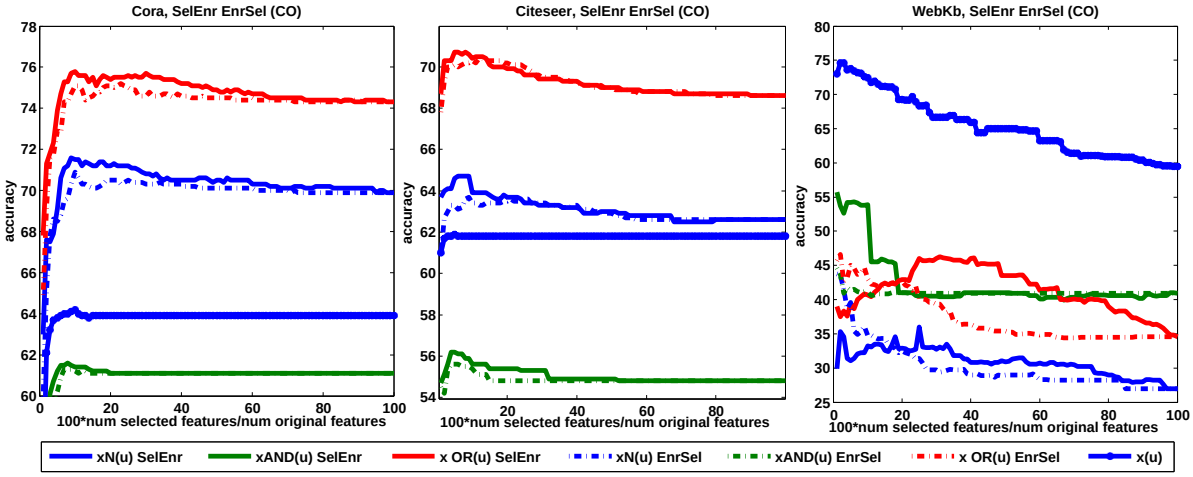


Figure 5.2 : Test accuracies of SelEnr and EnrSel feature selection and enrichment methods (content only classification).

other two datasets. When homophily (label correlation among neighbors) is low, using neighbors' label or feature information does not help much for classification.

In Figure 5.3, we compare the average (over 10 folds) test accuracies when instead of content only classification (CO), collective classification (ICA) is used. The figure shows that, as reported by (Sen et al. 2008), for higher homophily Cora and Citeseer datasets, using ICA with the original features improves the classification accuracy compared to using CO with the original features (both shown in blue). On the other hand, for the low homophily WebKB dataset, ICA and CO using the original features perform similarly. For Cora and Citeseer, OR enrichment performs best, followed by neighbor (N) enrichment and ICA improves classification performance compared to CO. ICA with AND enriched features perform only as well as CO classification using the original features. On the other hand, for the WebKB dataset, using OR and AND enrichment is better than using the neighbor (N) enrichment, although enrichment does reduce the classification accuracy compared to using the original features only.

We conclude that, for high homophily datasets like Cora and Citeseer, both collective classification and feature selection and enrichment improve classification accuracy by themselves and using them both gives even better results.

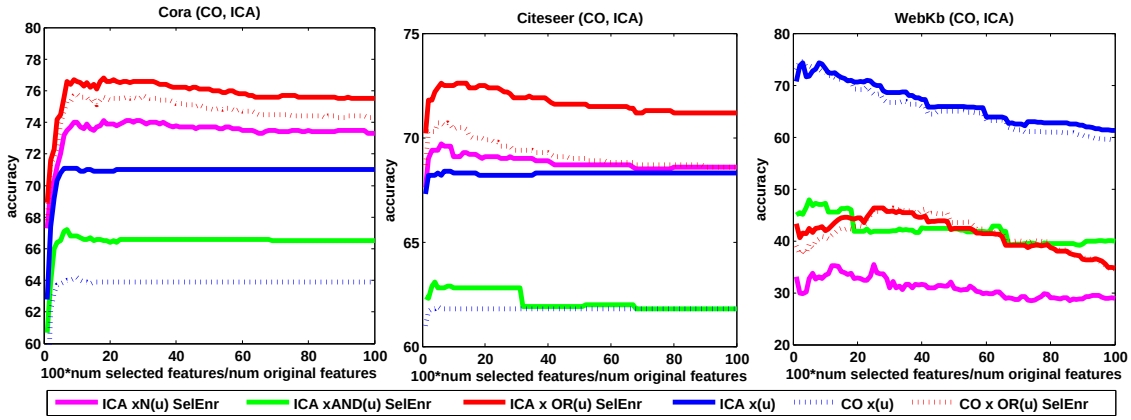


Figure 5.3 : Test accuracies of SelEnr and EnrSel feature selection and enrichment methods (collective classification).

5.3.2.3 Effect of test set size

In machine learning, it is well known that when a small training set size is used a classifier may overfit, and hence may perform poorly on unseen test data. Changing test set size may affect the variance of the test accuracy, but not necessarily its expected value. On the other hand, when networked data is used for classification, larger test sets could hinder the classification accuracy of ICA, because many predicted neighbor labels, instead of actuals ones, are used as inputs for classification. For enriched and content only classification, the actual test set features are available, therefore, enrichment could be less sensitive to changing test set size.

Figure 5.4 shows the average test classification accuracies obtained when original, neighbor and OR enriched features are used with different percentage of test set sizes. The horizontal axis shows the percentage of test size selected during snowball sampling. Once a test set is selected, the remaining data is used as the training set, therefore increased test set size means decreased training test set size and therefore decreased test accuracy. In the figure, feature selection is used to select the first 25% of the features considered and average test accuracies over 10 folds are shown. The figure shows that, when original features are used, compared to CO classification, ICA results in a performance increase for Cora and Citeseer and does not make a change for the WebKB dataset. On the other hand, for the high homophily Cora dataset, OR feature enrichment (with ICA or CO) is better than ICA on the original features as

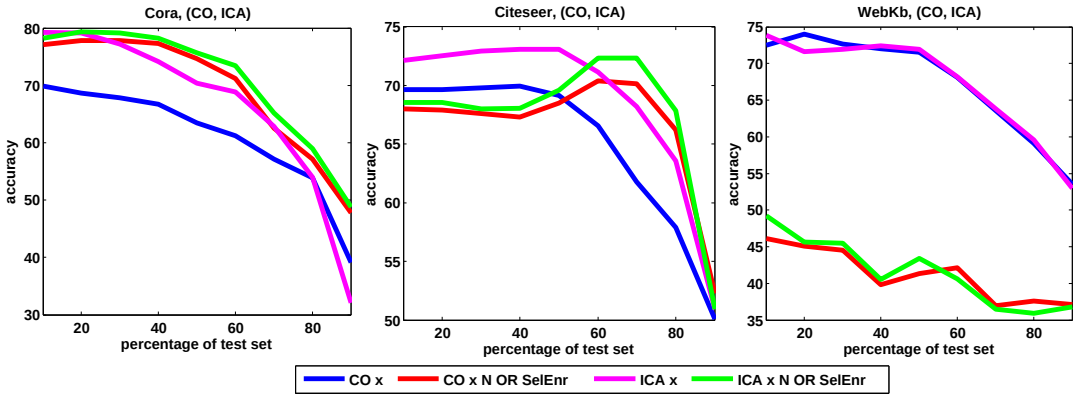


Figure 5.4 : Effect of test set size on test classification accuracy.

long as the test set size is larger than 30%. For the Citeseer dataset, which has less homophily, feature enrichment results in worst performance for small test set sizes and best performance as long as the test set size is larger than 60%. For Citeseer dataset, the test accuracies for small test set size together with the SelEnr method are low (shown in red). We inspected the training accuracies also and found out that, not only the test accuracies, but also training accuracies are low, which means that the model has *underfitted* the data. As long as there is homophily, for larger test set sizes, feature enrichment is better than ICA using the original features. When there is no homophily, as in the case of WebKB dataset, using only the enriched features and not considering the original features reduces performance regardless of the test set size.

5.3.2.4 Combinations of enrichment

Different types of feature enrichments can be combined as given in Equations 5.4-5.7. In Figure 5.5, we show the average (over 10 folds) test classification accuracies obtained using feature selection and then enrichment together with content only (CO) classification and collective classification (ICA). In these experiments, different from the ones given above, we include the original features among the features used as follows: we first select k original features, we enrich those features and select k of them, among the k original and k enriched features, we select a final set of k features. The test set size is 10% of the all available data for Cora and WebKB datasets, 30% of the all available data for the Citeseer dataset. Figure 5.5 shows that using the OR enriched features is the best when a very small number of features are selected for the Cora dataset. On the other hand, better classification accuracies are obtained when

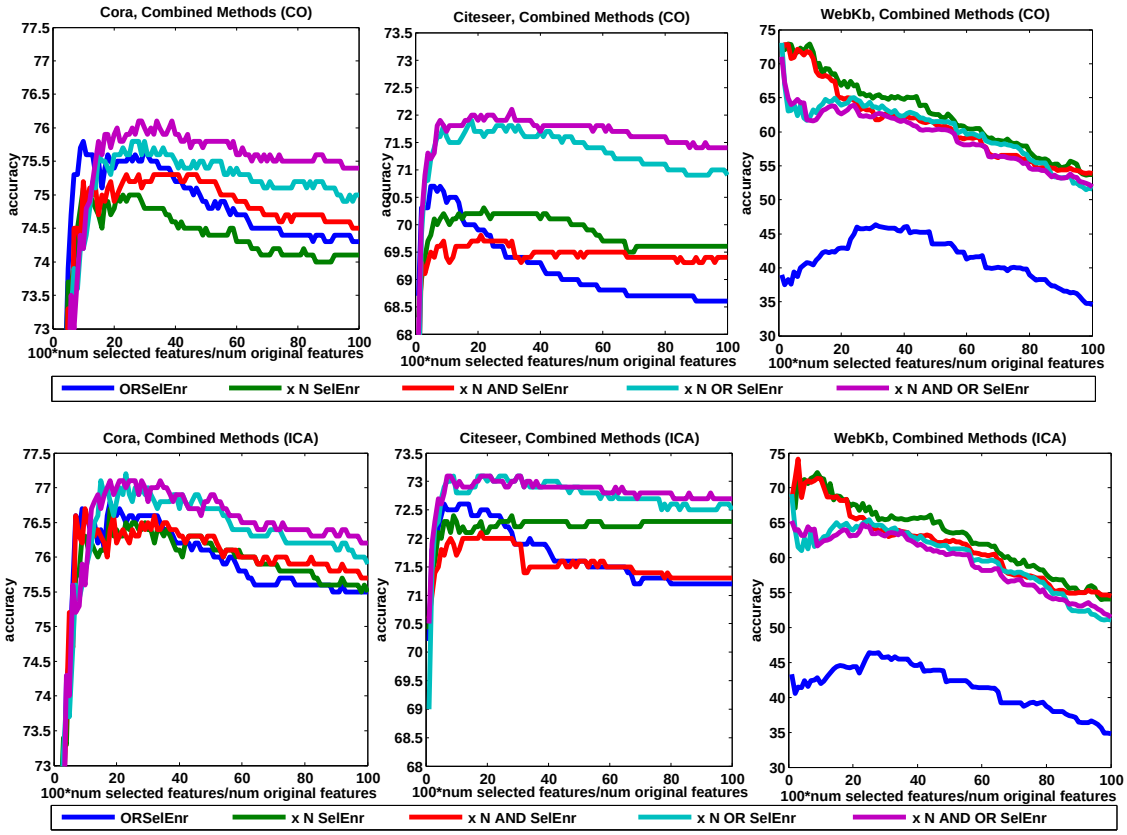


Figure 5.5 : Test accuracies for different combinations of enrichment and content only classification (top:CO, bottom:ICA).

the original features as well as all of the enriched features are used to select the final set of features. For the Citeseer dataset, again, using this form of enrichment gives better results than using only only OR enrichment. Test classification accuracies on WebKB dataset are very close to that of using the original features. Therefore, when homophily is low, considering the original features in addition to the enriched ones ensures that the accuracies are not affected when enrichment is used. For both Cora and Citeseer datasets, using ICA together with enriched features results in even better accuracies than using CO classification. When ICA is used, using AND enriched features in addition to all the other original and enriched features does not result in a performance change. For WebKB dataset, when ICA is used with both the the original and enriched features, the test classification accuracies are a little better than using ICA with the original features only. These experiments show that for both high and low homophily datasets, selecting from among the original and enriched features, it is possible to improve classification accuracies, regardless of whether content only or collective classification is used.

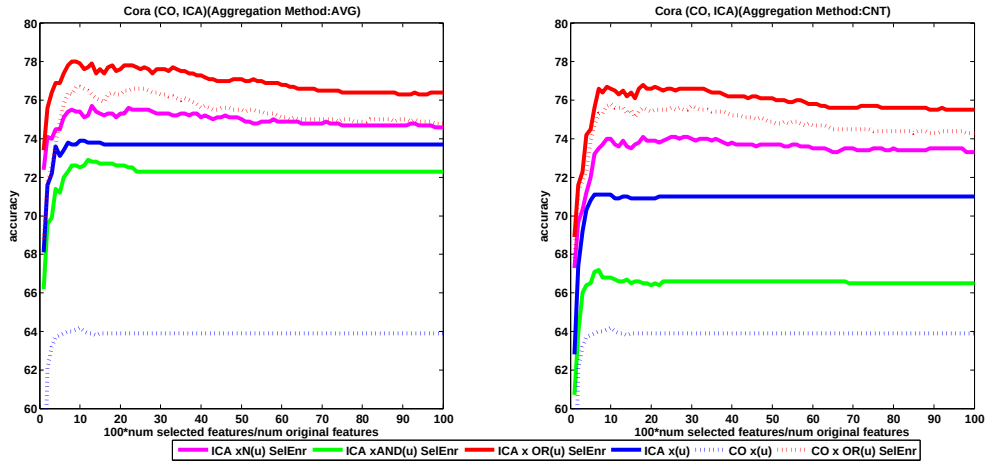


Figure 5.6 : Test accuracies for average aggregation (left) and count aggregation (right) methods on Cora dataset (CO, ICA).

5.3.2.5 Aggregation methods

In order to use features or labels of neighbors as features to a classifier, we employ aggregation methods. For label aggregation, previously, it was shown that count performed as good as other methods (Sen et al. 2008). So we used the count method in the previous experiments for both label and feature aggregation. In this section, we report experiments using the average aggregation, which is the count aggregation divided by the number of neighbor nodes. For label aggregation, average aggregated neighbor labels vector is defined as: $\mathbf{r}_N(u) = \frac{1}{|N(u)|}\sum_{v \in N(u)} \mathbf{r}(v)$ and for feature aggregation a similar definition is used. Figure 5.6 compares the count and average aggregation methods. Although the relative performances of enrichment and CO/ICA methods with respect to each other remain the same, there is a significant accuracy increase for all the methods when average aggregation is used. Content only classification using just plain features has the same performance as in the previous figures, because this method does not use aggregation. When we use ICA with plain (original) features, using average aggregated neighbor labels gives around 3% more accuracy. The best performance is again obtained with ICA using OR enriched features and by the average aggregation method. It should also be noted that AND enriched features perform significantly better with the average aggregation.

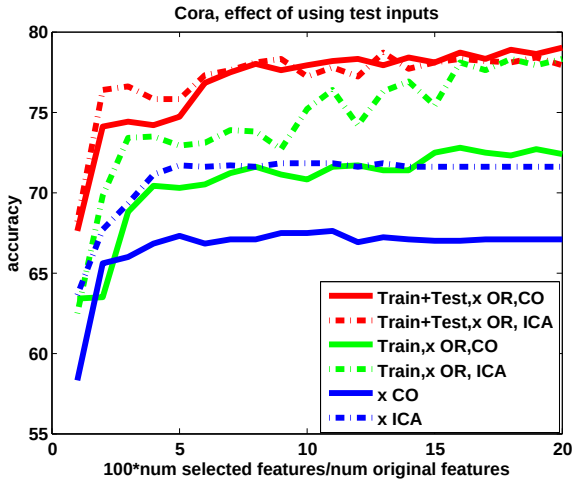


Figure 5.7 : Test accuracies on Cora dataset using train+test inputs (red lines) and using train inputs only (green lines).

5.3.2.6 Enrichment with and without test nodes

In all the experiments reported so far, we reported experiments in a transductive setting and while producing the enriched features we used both train and test nodes' features and links. In Figure 5.7, we show experiments using only the train nodes (Train) versus both train and test nodes (Train + Test) on Cora dataset. When test nodes are not used, all the classifiers' accuracies decrease. The accuracies obtained using feature enrichment with plain and ORed features are more accurate than content only and ICA when plain features are used. This result supports our previous thesis that feature enrichment is useful on high homophily datasets. An interesting point in the figure is that ICA using enriched features trained without test content features starts from an accuracy far lower when the number of selected features are low and converges to methods using test inputs when the number of selected features increase. The reason for that is as follows: ICA uses test nodes' labels during its iterations and indirectly uses test inputs' features during assignment of labels to test nodes.

5.3.2.7 Synthetic dataset experiments

In this section, we report experiments using enriched features on a synthetic networked dataset. In order to create synthetic networked data, we used a method that allows varying content and link relevances with the class label and varying dependence

(redundancy) between content and link. As in the "content based" networks of (Balcan & Erzan 2004), we generated content and link bits, and based on their link similarity we connected the nodes in the network. Content features $\mathbf{x}(u)$ are produced for each node u. In order to produce links between nodes, a similarity measure between them is needed. We used an integer power of inverse normalized hamming distance as the similarity measure. Class labels are determined according to the mode of the complete feature vector. We used the same dataset produced with $ms = 32$ as in (Cataltepe et al. 2011). Please see (Cataltepe et al. 2011) for more details.

In Figure 5.8, we compare the test accuracies of different SelEnr and EnrSel feature enrichment methods for content only and collective classification on synthetic dataset. According to the figure, almost all methods perform similar to each other except two of them. These are x N SelEnr and x OR Enr Sel. While x OR Enr Sel method starts from a point far back from others, as the selected number of features increases, it converges to the others. On the other hand, x N SelEnr method diverges from the others and end with a lower accuracy. The behavior is the same when collective classification is used. Using collective classification, accuracies are higher. The behavior of feature enrichment methods on synthetic datasets differ from real datasets considered. The main reason for that can be explained by the big difference in the number of content features. While real datasets have content features in thousands, our synthetic dataset has only 32 content features. Since there are only a few features and labels are computed directly based on the content features, we get the best accuracy values when we use all the features. In order to validate this hypothesis, we added additional 96 random features to nodes' existing 32 features, resulting in a set of 128 content features. Then we performed the same set of experiments, whose results are shown in Figure 5.8, bottom row. In this figure, the horizontal axis, again, shows the ratio of the selected features. The original 32 features are obtained when 25% of the 128 features are selected. Selection of the additional random features does not increase and may even decrease the test accuracy.

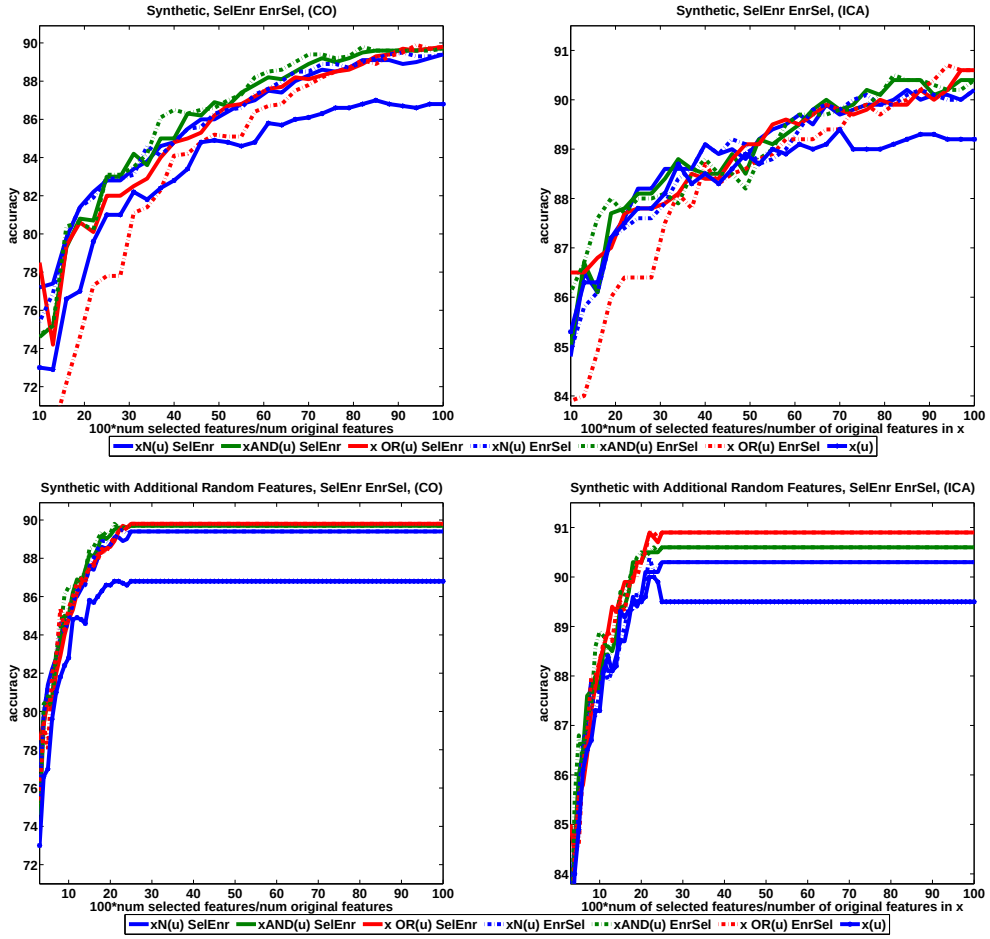


Figure 5.8 : Test accuracies of SelEnr and EnrSel feature selection and enrichment methods on Synthetic datasets (top:no noise, bottom:noisy).

5.4 Discussion

We have introduced a number of feature enrichment methods that can be used for transductive classification of networked data. The enriched features enable each node in the network to use its features as well as its neighbors' features. In addition to simple aggregation of neighbor features, we have used aggregated functions, such as AND, OR, of node and neighbor features. We have shown that, on the enriched features, the FCBF# feature selection method can be used to rapidly select a set of relevant and non-redundant set of features for classification. We have observed that when a networked dataset has high homophily, as in the case of the Cora and Citeseer datasets, feature selection and enrichment together with content only classification, results in better classification accuracies than using original features together with collective

classification. Collective classification with selected and enriched features may further increase the classification accuracies.

Selected and enriched features do not need the labels but only features of neighbor nodes, therefore, especially for large test sets, using selected and enriched features is more beneficial than using collective classification, which needs estimated test labels.

We have also shown that even for low homophily datasets, if in addition to the enriched features, original features are also considered for feature selection, feature enrichment is at least as good as using the original features only. If a dataset has high homophily, as in the case of Cora or CiteSeer, just like collective classification, enriched features also benefit from this. On the other hand, if homophily is small, as in the case of the WebKB dataset, enriched and selected features may be as good as the plain node features. In general, one can check if enriched features could be useful for classification, by only computing their relevances, which is a lot faster than training a classifier for all possible sets of selected and enriched features.

There are a number of future work directions. Other feature enrichment methods based on the characteristics of a specific networked dataset could be explored. Exploration of aggregation methods other than count and average and exploration of neighborhood functions that consider more than just the immediate neighbors are other research directions.

6. ONE AGAINST ALL CLASSIFICATION OF NETWORKED DATA

6.1 Background and Purpose

In networked data there may be dependencies between labels of neighboring nodes. These dependencies can be used to improve classification accuracy, by means of training classifiers that use node features and aggregated neighbor labels. While the labels are available for neighbors in the training set, they need to be estimated by the classifier for neighbors in the test set, using methods such as collective classification. In previous studies, it has been observed that in order to have an accuracy increase when neighbor information is used, homophily, which measures the label-label correlation between neighboring nodes, is required. On the other hand, for multi-class datasets, homophily for each class could be different. In this section, for classification of multi-class networked data, instead of using a single classifier to learn all classes, we use a one-against-all scheme and learn a separate classifier for each class. We extend this one-against-all setting to collective classification also. Although one-against-all classification increases the training and testing time due to the increase in the number of classifiers, experimental results show that OAA content only and collective classification is better than single classifier content only and collective classification. The benefit of OAA learning becomes more emphasized with increase in homophily or decrease in available training data size.

6.2 Methodology

The algorithm created for one against all classification of networked data is given in Algorithm 5. The algorithm gets the original graph and number of classes as inputs. Also needs maximum stability and maximum number of iterations parameters in case collective classification requested is an instance of link only or ICA classifier. The algorithm first checks if the network is a multi-class network or not. If it is not then

apply the traditional collective classification algorithm. If it is a multi-class network, which is suitable for one against all scheme, then creates class based binary graphs for each class. In these binary graphs, the nodes in the graph belonging to that class are assigned a a label of "1", the class labels for all the remaining nodes are assigned "0". After the class based graphs are created, a classifier is trained for each graph.

Any test node is classified as follows (Please see Algorithm 6) : The test node is given to each one against all trained classifier. The node is assigned to the class, for which class membership probability output of the corresponding classifier is maximum.

For link only and ICA collective classification, iterations continue until network reaches to a certain level of stability or number of iterations reach to maximum allowed. In the iterations stage, the classification of the node is done as in Algorithm 6. However since labels of the test nodes may change during iterations, it is important to keep the class based binary graphs synchronized. To accomplish this, if the node's label is changed, both old label's graph and new label's graph are updated.

6.3 Experiments

6.3.1 Experimental setup

6.3.1.1 Datasets

Citeseer, Cora, HepTh scientific publication datasets and WebKb web pages dataset are used in the experiments. For details about these datasets please refer to Section 1.4.

6.3.1.2 Sampling

Test-train splits are produced 10 times and average accuracies over these folds are reported for all experiments. In the experiments,different train/test percentages are used, where test percentage is changed from 10 percent to 90 percent by an increment of 10 percent of the whole dataset. For all of these combinations, both random sampling and snowball sampling are used whose details are given in Section 1.2.1.

Algorithm 5 One-Against-All algorithm flow for collective classification.

Input: K = Number of classes
Input: MS = Maximum stability
Input: M = Maximum number of iterations
Input: $G = (V, E), V_{train}, V_{test}$
Output: Classified instances with one against all scheme
if $K \leq 2$ **then**
    doPlainClassificationX($G_{original}$) where $X \in CO, LO, ICA$
else
    // Construct class based binary graphs G_{C_y} where $y \in 1..K$
    for $i = 1$ to K **do**
        for all $u \in V_{train}$ **do**
            if $i = r(u)$ **then**
                $r(u) = 1$
            else
                $r(u) = 0$
            end if
        end for
    end for
    // Train classifiers with G_{C_y} where $y \in 1..K$
    // $CBTCA$ = Class based trained classifiers array
    for $i = 1$ to K **do**
        $CBTCA(i) = TrainClassifier(G_{C_i})$
    end for
    // Evaluate the nodes in test set.
    // Then set the estimated label as maximum of classifier outputs.
    for all $u \in V_{test}$ **do**
        $\hat{r(u)}$ = arg max $(EvaluateGivenNodeAsClassBased(G, K, u, CBTCA))$
    end for
    if Classifier is an instance of Link Only or ICA Classifier **then**
        while $iterations < M$ & $stability < MS$ **do**
            Generate ordering O over nodes in V_{test}
            for all $u \in O$ **do**
                r_{old} = $\hat{r(u)}$
                $\hat{r(u)}$ = arg max $(EvaluateGivenNodeAsClassBased(G, K, u, CBTCA))$
                if $r_{old} \neq r(u)$ **then**
                    Update class based binary graphs G_{C_y} where $y \in r_{old}, r(u)$
                end if
            end for
        end while
    end if
end if

Algorithm 6 Class based evaluation of a given node.

1: **function** EVALUATEGIVENNODEASCLASSBASED(G, K, u, $CBTCA$)
2: Input: K = Number of classes
3: Input: u = Node to be evaluated
4: Input: $G = (V,E), V_{train}, V_{test}$
5: Input: $CBTCA$ = Class Based Trained Classifiers Array
6: Output: $testProbabilityArrayForTheNode$
7: **for** $clsIndex = 0$ to K **do**
8: testProbabilityArrayForTheNode(clsIndex) = Evaluate the node u with $CBTCA(clsIndex)$ and get $P_{g_{clsIndex}}(\tilde{\mathbf{r}} = 1|u)$
9: **end for**
10: **return** testProbabilityArrayForTheNode
11: **end function**

6.3.1.3 Classification methods

A base classifier which is trained on node features and local connectivity information is needed for collective classification.

In the experiments, logistic regression (LR), Support Vector Machine (SVM), Naive Bayes (NB), Bayes Net (BN), k-Nearest Neighbor (kNN, k=1 and k=3), Random Forest (RF), Radial Basis Function (RBF) and J48 are used as base classifiers for the g_{CO}, g_{LO} and $g_{CO,LO}$ classification. For all of the methods Weka implementations with default parameters (unless otherwise noted) have been used.

6.3.2 Experimental results

6.3.2.1 Experimental evaluation

First of all, we conducted the experiments with different classifiers given in Section 6.3.1.3 by using content only, link only and ICA and their one against all versions for all different train/test ratios mentioned in section 6.3.1.2 for both random and snowball sampling for all datasets. Since a total of 10 different classifiers were used with two different sampling methods with 9 different train/test ratios for each fold, 180 content only experiment results, 360 link only results, and 360 ICA results were obtained. For link only and ICA, we did not only generate the results obtained at the end of the iterations, but also the result obtained during start of the iterations. That's why the

results obtained for link only and ICA were twice of the content only results. Since we used 10-fold cross validation, during the experiments, the number of results were multiplied by 10, resulting with a total 18000 experiment results composed of 9000 plain and 9000 OAA experiment results.

6.3.2.2 Performance of different classifiers

The results of the experiments mentioned in Section 6.3.2.1 are given in Figures 6.1 and 6.2 for Citeseer dataset, in Figures 6.3 and 6.4 for Cora dataset, in Figures 6.5 and 6.6 for WebKb dataset and in Figures 6.7 and 6.8 for HepTh dataset. To be able to visualize the results better, the results obtained with OAA scheme versus plain scheme, are presented in two different figures for each dataset. The figures are separated according to different set of classifiers, which of each includes 5 different classifiers. The first classifier set includes LR, SVM, BN, NB and RF, while the second classifier set includes kNN1, kNN3, RBF, J48 and libSVM. The difference between test accuracy performance of the classifiers with OAA scheme versus plain scheme are also given in Figures 6.9, 6.10, 6.11 and 6.12.

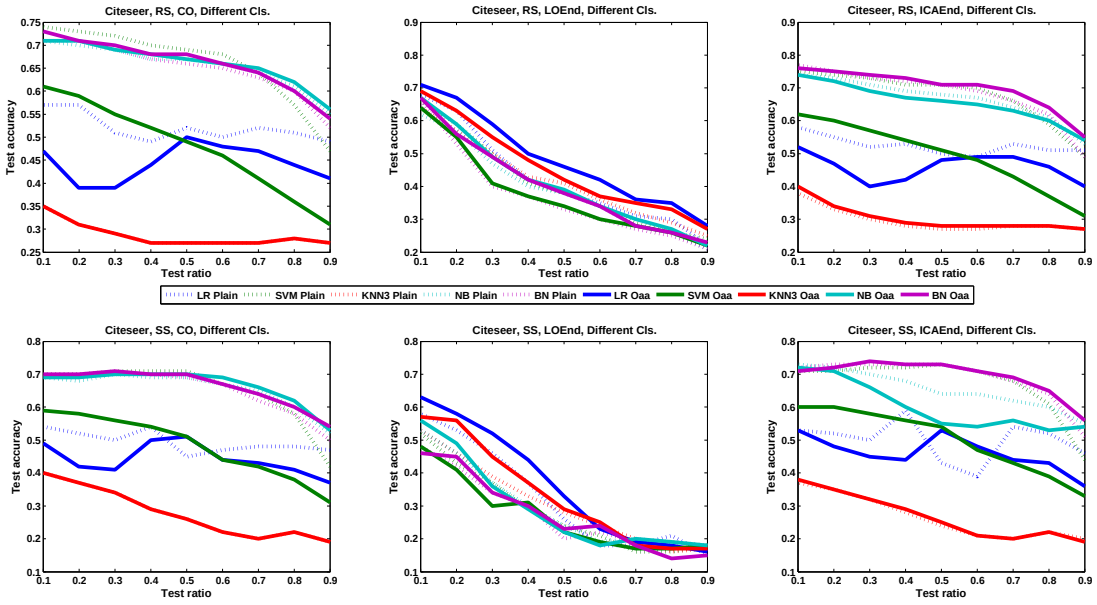


Figure 6.1 : Citeseer OAA versus Plain methods comparison (classifier set 1)

According to the Figures 6.1, 6.2 and 6.9, on Citeseer dataset, for content only classification, RF benefits most from OAA scheme, while SVM, RBF and LR fails to achieve better performance regardless of the sampling method used. When we consider link only, almost all classifiers' iterations start accuracy is better with OAA

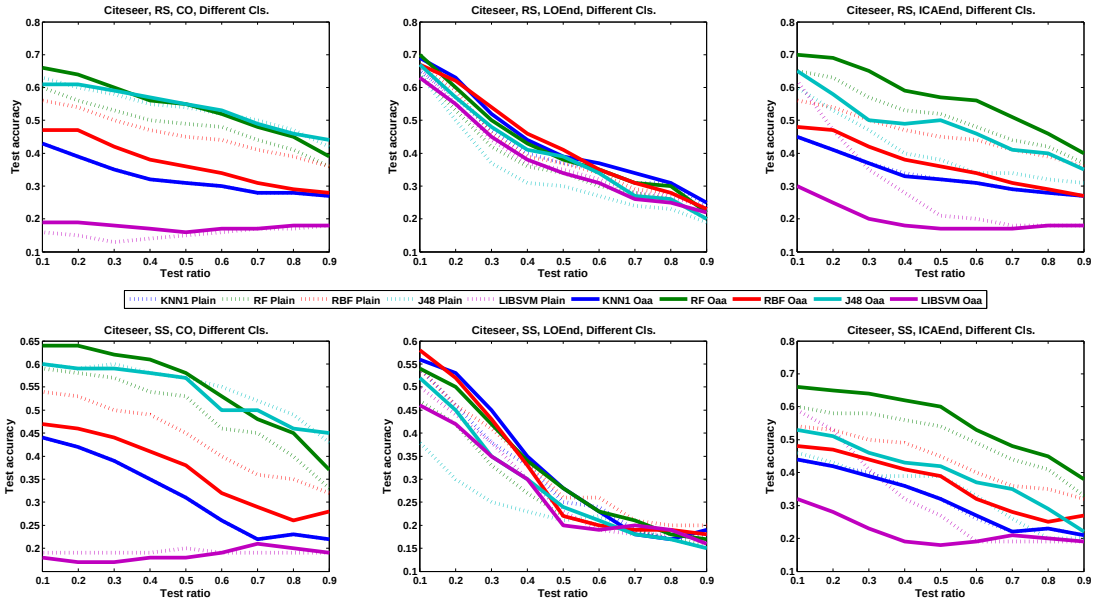


Figure 6.2 : Citeseer OAA versus Plain methods comparison (classifier set 2).

scheme. RF, J48, BN, and LR achieve better test accuracy performance with OAA scheme. KNN classifiers also get better performance. The situation does not change for link only iterations end accuracies. SVM's performance decreases significantly again with OAA scheme. When we consider collective classification with ICA, RF gets best performance increase while SVM gets significant performance decrease with the scheme. Another observation is that J48 benefits more from RF at the ICA iterations end stage when snowball sampling is considered.

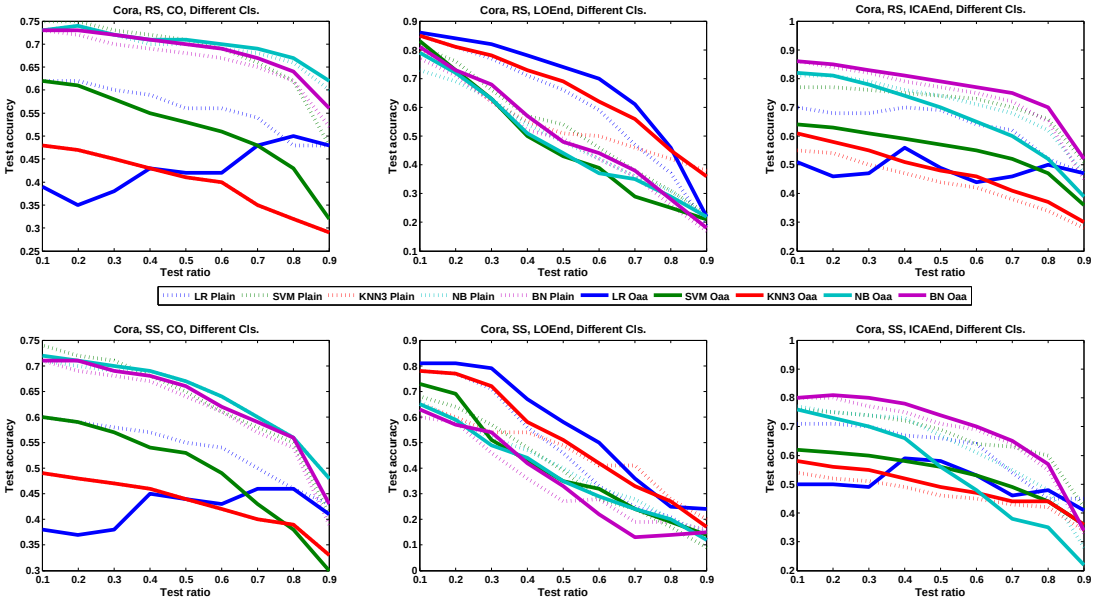


Figure 6.3 : Cora OAA versus Plain methods comparison (classifier set 1).

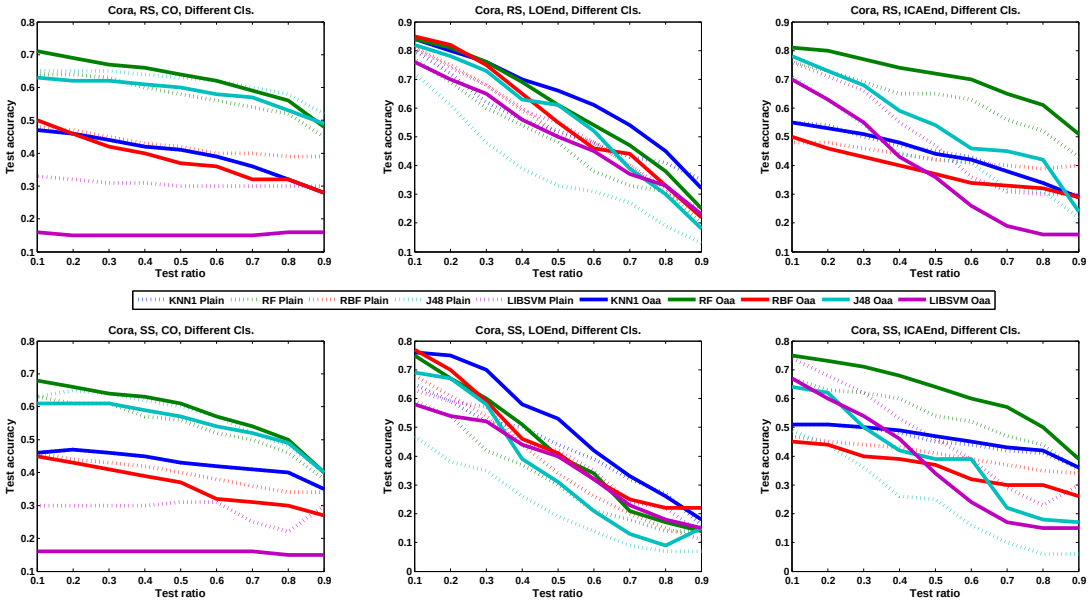


Figure 6.4 : Cora OAA versus Plain methods comparison (classifier set 2).

According to the Figures 6.3, 6.4 and 6.10, on Cora dataset, for content only classification, while RF benefits most from OAA scheme, SVM, LR, and libSVM fails with the scheme. When we consider link only, again like on Citeseer dataset, almost all classifiers' iterations start accuracy is better with OAA scheme. The performance of SVM, NB and libSVM decreases significantly at the end of iterations. When ICA is considered, RF, J48 and BN get the most benefit from OAA scheme. The performance of LR, SVM, NB and libSVM decreases at the end of ICA iterations with the scheme.

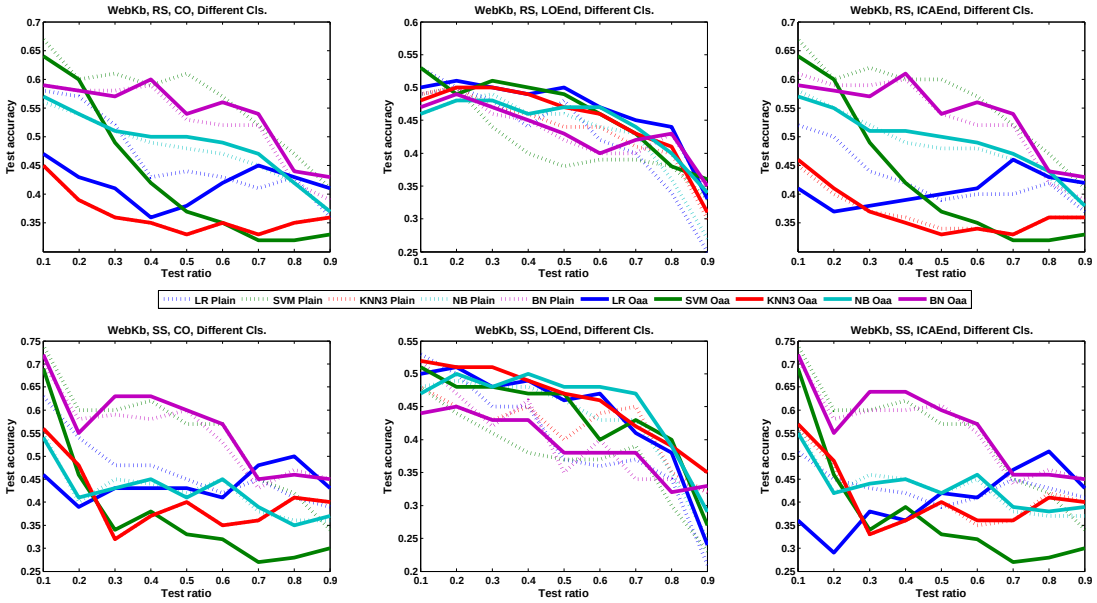


Figure 6.5 : WebKb OAA versus Plain methods comparison (classifier set 1).

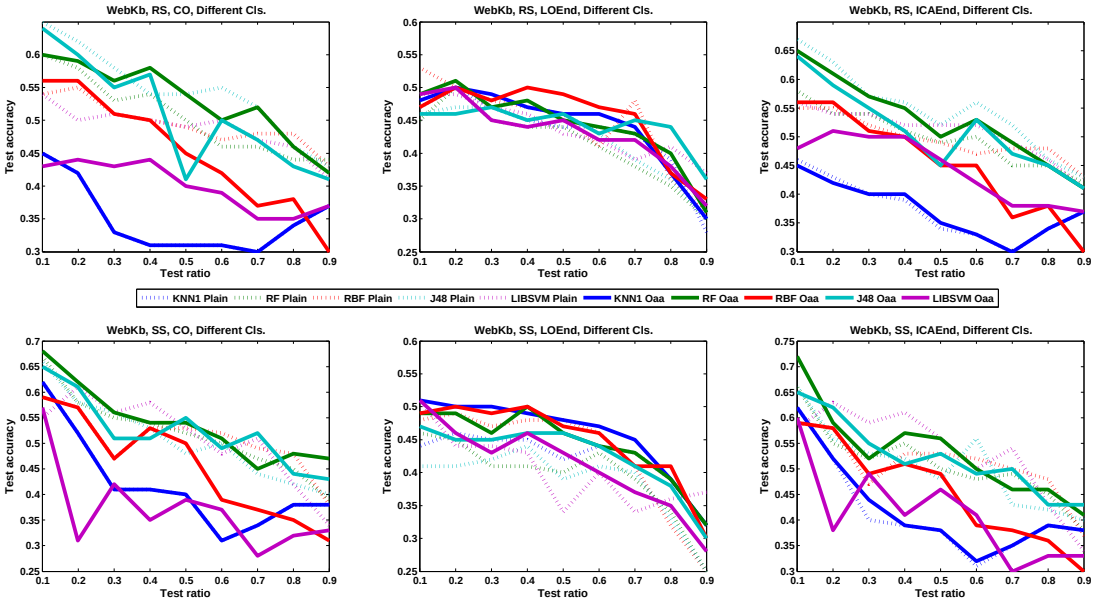


Figure 6.6 : WebKb OAA versus Plain methods comparison (classifier set 2).

According to the Figures 6.5, 6.6 and 6.11, on WebKb dataset, BN and RF get benefit from OAA scheme when content only classification is considered. The performance of SVM, LR and libSVM decrease significantly. However when link only is considered, apart from the situation observed on Citeseer and Cora datasets, not all classifiers get better performance at the iterations start stage. With OAA scheme, SVM followed by LR, perform best. RF and J48 also achieve better performance with OAA scheme at the iterations end stage. When ICA is considered, a significant decrease in performance of SVM is observed. RF again gets the most benefit from OAA scheme.

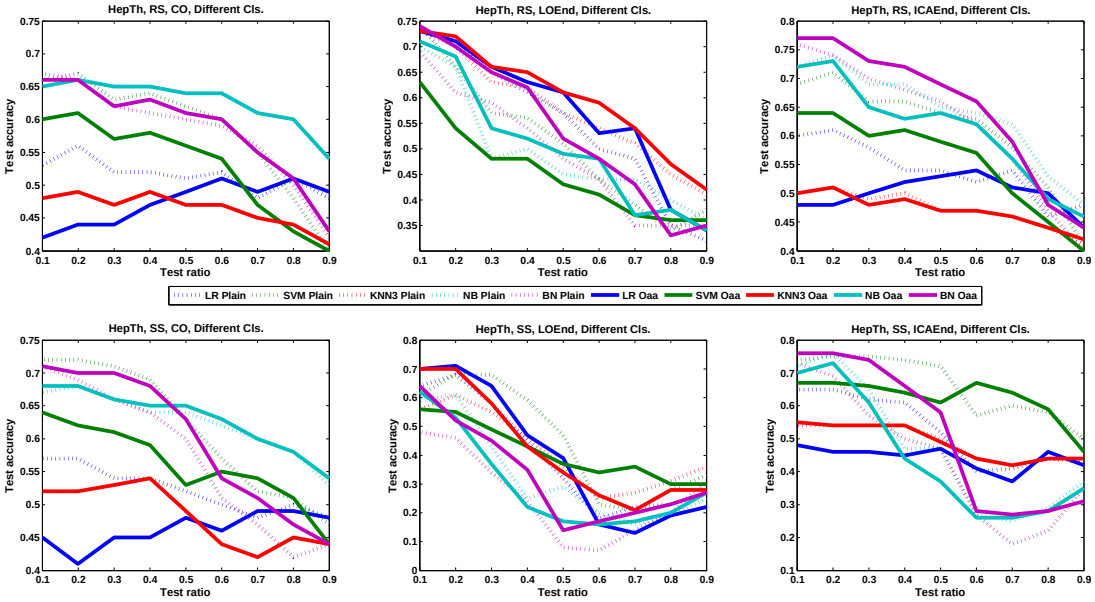


Figure 6.7 : HepTh OAA versus Plain methods comparison (classifier set 1).

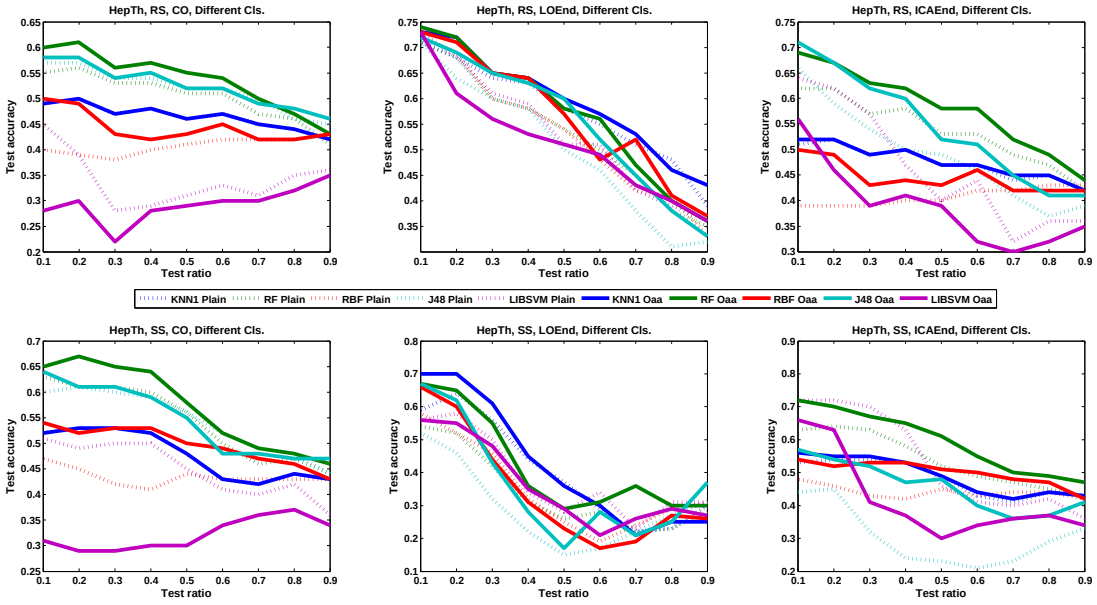


Figure 6.8 : HepTh OAA versus Plain methods comparison (classifier set 2).

According to the Figures 6.7, 6.8 and 6.12, on HepTh dataset, RF and surprisingly RBF get better results from OAA scheme when random sampling is considered with content only classification. The performance increase of RBF is more emphasized when snowball sampling is used. LR, SVM and libSVM again get worse performance with the scheme. When link only classification is considered, all classifiers especially BN and J48 for random sampling, BN, J48 and RF for snowball sampling get better performance with OAA scheme with two exceptions, namely SVM and libSVM for random sampling. NB also joins to these exceptions when snowball sampling is considered. J48, RBF, RF and BN are the classifiers that benefit more when collective classification with ICA is consider. SVM and LR again show decrease in performance with OAA scheme.

6.4 Discussion

In this section, an algorithm for application of one-against-all scheme to networked data is constructed. The most important part of this algorithm is the information flow of labels during iterations over multiple graphs. Test accuracy performances of the new methods, namely OAACO, OAALO, OAAICA, are compared with traditional collective classification methods CO (Content Only), LO (Link Only) and ICA test

accuracy performances for different train/test ratios on four multi-class network datasets.

It has been shown that the benefit of using OAA scheme is strongly dependent to the classifier used. Generally like RF and J48, decision tree based methods get most benefit when OAA scheme is used. BN follows them. When link only classifiers are considered, almost all classifiers get better performance with OAA scheme. OAA scheme also provides a much better starting point for link only and ICA iterations and generally end up with an almost equal or better test accuracy.

The performance of SVM followed by LR generally decrease with OAA scheme. BN is suggested to be used with OAA scheme, since the plain version of BN achieves best performance and using OAA scheme provides additional performance to it.

OAA-Plain/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End
LR	-0,08	0,05	0,06	-0,09	-0,11
SVM	-0,16	0	-0,02	-0,17	-0,18
KNN1	0	0	0,01	0	0
KNN3	0	0,02	0,02	0	0
NB	0,01	0,01	0,01	0,05	0,01
BN	0,02	0,03	0,02	0,03	0,06
RF	0,03	0,02	0,02	0,03	0,03
RBF	-0,08	0,01	0	-0,09	-0,09
J48	0,02	0,02	0,01	0,02	0,04
LIB_SVM	0	0	-0,01	0	0

CO	LO-Start	LO-End	ICA-Start	ICA-End	OAA-Plain/SS/0.9
-0,1	0,01	0,01	-0,09	-0,1	LR
-0,11	0	0	-0,1	-0,11	SVM
0	0	0,01	0	0	KNN1
0	0	0	-0,01	-0,01	KNN3
0,01	0	0	0,06	-0,01	NB
0,04	-0,02	-0,03	0,06	0,05	BN
0,04	0	0,01	0,05	0,05	RF
-0,04	-0,01	-0,02	-0,05	-0,05	RBF
0,02	0	0	-0,03	0,02	J48
0	-0,01	-0,02	0	0	LIB_SVM

OAA-Plain/RS/0.8					
LR	-0,07	0,04	0,05	-0,05	-0,05
SVM	-0,21	0	0	-0,22	-0,22
KNN1	0	0,03	0	0	0
KNN3	0	0,02	0,04	0	0
NB	0,01	0,01	0,01	0,02	-0,01
BN	0	0,04	0,01	0,05	0,02
RF	0,04	0,06	0,04	0,04	0,04
RBF	-0,1	0,02	0	-0,1	-0,1
J48	-0,01	0,04	0,03	0,01	0,08
LIB_SVM	0,01	0,02	-0,02	0	0

					OAA-Plain/SS/0.8
-0,07	-0,02	-0,03	-0,08	-0,09	LR
-0,2	0,01	0,01	-0,21	-0,22	SVM
0	0,01	0	0	0	KNN1
0	0,01	0,01	0	0	KNN3
0,02	0	0	-0,02	-0,07	NB
0,02	-0,04	-0,05	0,03	0,01	BN
0,05	0,01	0,01	0,04	0,04	RF
-0,09	-0,01	-0,01	-0,1	-0,1	RBF
-0,03	0,01	0	0,01	0,09	J48
0,01	0,02	0,01	0,01	0,01	LIB_SVM

OAA-Plain/RS/0.7					
LR	-0,05	0,05	0,05	-0,04	-0,04
SVM	-0,23	0	-0,01	-0,23	-0,23
KNN1	0	0,04	0,03	0	0
KNN3	0	0,03	0,03	0	0
NB	0,01	0,02	0,02	0,02	-0,01
BN	0,01	0,05	0,01	0,05	0,03
RF	0,04	0,05	0,03	0,06	0,07
RBF	-0,1	0,03	0,02	-0,1	-0,1
J48	-0,01	0,04	0,03	0,02	0,07
LIB_SVM	0	0,02	-0,02	0	-0,01

					OAA-Plain/SS/0.7
-0,05	-0,01	0,01	-0,1	-0,1	LR
-0,22	0,02	0,01	-0,22	-0,25	SVM
0	-0,01	-0,01	0	0	KNN1
0	-0,01	-0,02	0	0	KNN3
0,01	-0,01	0	-0,01	-0,06	NB
0,02	-0,04	-0,01	0,04	0,01	BN
0,03	0,01	0,02	0,02	0,04	RF
-0,07	-0,02	-0,02	-0,08	-0,08	RBF
-0,02	-0,03	-0,01	0	0,09	J48
0,02	0,03	0,02	0,02	0,02	LIB_SVM

OAA-Plain/RS/0.6					
LR	-0,02	0,05	0,07	-0,02	0
SVM	-0,22	-0,01	-0,04	-0,22	-0,22
KNN1	0	0,05	0,02	0	0
KNN3	0	0,03	0,01	0,01	0,01
NB	0	0,02	0	0,02	-0,02
BN	0,01	0,07	0,04	0,04	0,02
RF	0,04	0,06	0,05	0,08	0,08
RBF	-0,1	0,04	0,01	-0,1	-0,1
J48	0,01	0,07	0,07	0,04	0,12
LIB_SVM	0,01	0,02	-0,01	-0,03	-0,03

					OAA-Plain/SS/0.6
-0,03	0,02	0,05	0,08	0,09	LR
-0,23	-0,02	-0,02	-0,23	-0,24	SVM
0	0,01	-0,01	0	0,01	KNN1
0	0	0,01	0	0	KNN3
0,02	-0,01	-0,01	-0,02	-0,1	NB
0	-0,02	0,02	0,03	0	BN
0,07	0	0,03	0,01	0,04	RF
-0,08	-0,03	-0,06	-0,08	-0,08	RBF
-0,05	-0,04	-0,01	0	0,04	J48
0	0,02	0	0	0	LIB_SVM

OAA-Plain/RS/0.5					
LR	-0,02	0,05	0,08	-0,02	-0,02
SVM	-0,2	-0,02	-0,04	-0,2	-0,2
KNN1	0	0,05	0,02	0	0
KNN3	0	0,05	0,01	0,01	0,01
NB	0	0,02	0,01	0,02	-0,02
BN	0,02	0,07	0,05	0,03	0
RF	0,06	0,07	0,04	0,07	0,05
RBF	-0,09	0,06	0,04	-0,09	-0,09
J48	0,01	0,07	0,09	0,06	0,12
LIB_SVM	0,01	0,03	-0,01	-0,04	-0,04

					OAA-Plain/SS/0.5
0,06	0,05	0,07	0,08	0,1	LR
-0,2	-0,02	-0,02	-0,19	-0,19	SVM
0	0,04	0,03	-0,01	0	KNN1
0	0,05	0,01	-0,01	0,01	KNN3
0,01	0,01	-0,02	-0,01	-0,09	NB
0,01	0,04	0,03	0,03	0	BN
0,05	0,04	0,06	0,06	0,06	RF
-0,07	0,03	-0,04	-0,06	-0,06	RBF
0	0,05	0,03	0,03	0,03	J48
-0,02	0,01	-0,03	-0,09	-0,09	LIB_SVM

OAA-Plain/RS/0.4					
LR	-0,05	0,05	0,08	-0,1	-0,11
SVM	-0,18	-0,01	-0,04	-0,18	-0,17
KNN1	0	0,05	0,04	0	-0,01
KNN3	0	0,04	0,05	0,01	0,01
NB	0,01	0,03	0,02	0,01	-0,02
BN	0,01	0,08	0,05	0,02	0,01
RF	0,06	0,09	0,07	0,08	0,06
RBF	-0,09	0,06	0,07	-0,09	-0,09
J48	0,02	0,07	0,1	0,05	0,09
LIB_SVM	0,03	0,04	0,01	-0,1	-0,1

					OAA-Plain/SS/0.4
-0,04	0,06	0,07	-0,15	-0,15	LR
-0,17	-0,01	0,02	-0,16	-0,16	SVM
0	0,06	0,02	-0,01	0	KNN1
0	0,07	0,04	0	0,01	KNN3
0,01	0,02	-0,01	0	-0,08	NB
0	0,09	0	0,03	0	BN
0,07	0,07	0,07	0,06	0,06	RF
-0,08	0,04	-0,01	-0,08	-0,08	RBF
0	0,08	0,07	0,02	0,04	J48
-0,01	0,03	0	-0,12	-0,13	LIB_SVM

OAA-Plain/RS/0.3					
LR	-0,12	0,06	0,08	-0,12	-0,12
SVM	-0,17	-0,04	-0,09	-0,16	-0,16
KNN1	0	0,07	0,05	0	0
KNN3	0	0,06	0,05	0	0,01
NB	0	0,03	0,02	0,01	-0,02
BN	0,01	0,1	0,09	0,02	0,01
RF	0,07	0,09	0,08	0,08	0,08
RBF	-0,08	0,07	0,08	-0,08	-0,08
J48	0,01	0,08	0,11	0,01	0,03
LIB_SVM	0,05	0,04	0,01	-0,15	-0,15

					OAA-Plain/SS/0.3
-0,09	0,05	0,06	-0,06	-0,05	LR
-0,15	-0,01	-0,07	-0,14	-0,14	SVM
0	0,11	0,07	0	-0,01	KNN1
0	0,12	0,06	0	0	KNN3
0	0,03	-0,01	0,01	-0,04	NB
0,01	0,11	0	0,03	0,01	BN
0,05	0,13	0,09	0,06	0,06	RF
-0,06	0,06	0,02	-0,06	-0,06	RBF
-0,01	0,11	0,1	0,02	0,07	J48
-0,02	0	-0,03	-0,16	-0,18	LIB_SVM

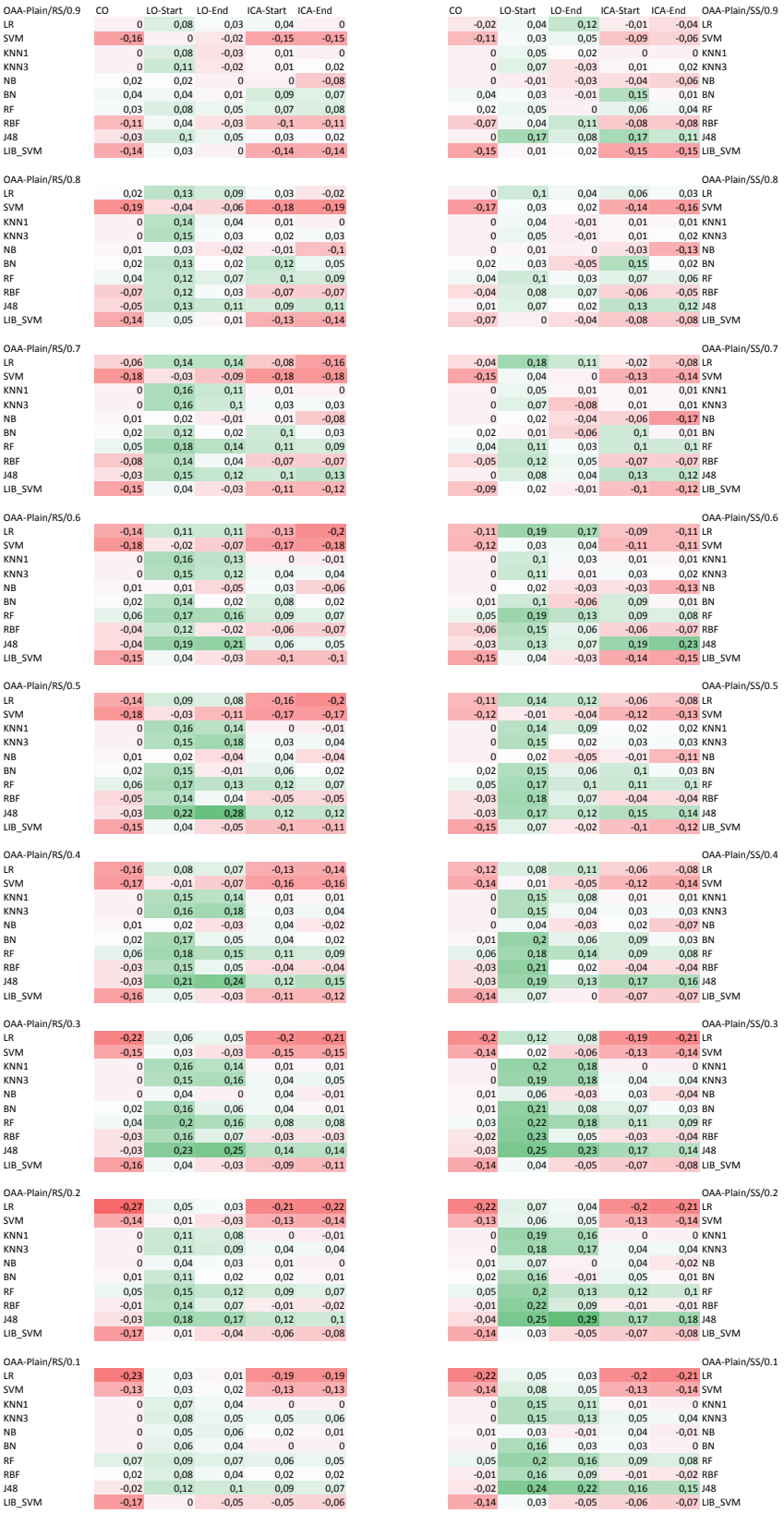
OAA-Plain/RS/0.2					
LR	-0,18	0,05	0,04	-0,08	-0,08
SVM	-0,14	-0,03	-0,04	-0,14	-0,14
KNN1	0	0,08	0,06	0	0
KNN3	0	0,07	0,06	0	0,01
NB	0,01	0,03	0,02	0,01	-0,01
BN	0	0,07	0,03	0,01	0
RF	0,08	0,1	0,07	0,08	0,06
RBF	-0,07	0,07	0,03	-0,07	-0,07
J48	0,01	0,07	0,07	0,04	0,05
LIB_SVM	0,04	0,03	0	-0,2	-0,22

					OAA-Plain/SS/0.2
-0,1	0,05	0,05	-0,03	-0,04	LR
-0,12	0,01	-0,02	-0,11	-0,11	SVM
0	0,11	0,07	0	0	KNN1
0	0,11	0,1	0	0	KNN3
0,01	0,04	0,03	0,01	-0,01	NB
0	0,11	0,03	0,02	-0,01	BN
0,06	0,12	0,08	0,06	0,07	RF
-0,07	0,08	0,06	-0,06	-0,06	RBF
0	0,12	0,15	0,04	0,08	J48
-0,02	0,04	-0,02	-0,21	-0,25	LIB_SVM

OAA-Plain/RS/0.1					
LR	-0,1	0,03	0,03	-0,06	-0,06
SVM	-0,13	-0,01	-0,02	-0,13	-0,13
KNN1	0	0,05	0,04	0	0
KNN3	0	0,06	0,05	0,02	0,02
NB	0	0,05	0,05	0,01	0
BN	0,02	0,05	0,01	-0,01	-0,01
RF	0,06	0,07	0,06	0,06	0,05
RBF	-0,09	0,02	0	-0,08	-0,08
J48	-0,02	0,05	0,03	0,03	0,05
LIB_SVM	0,03	-0,02	-0,03	-0,29	-0,32

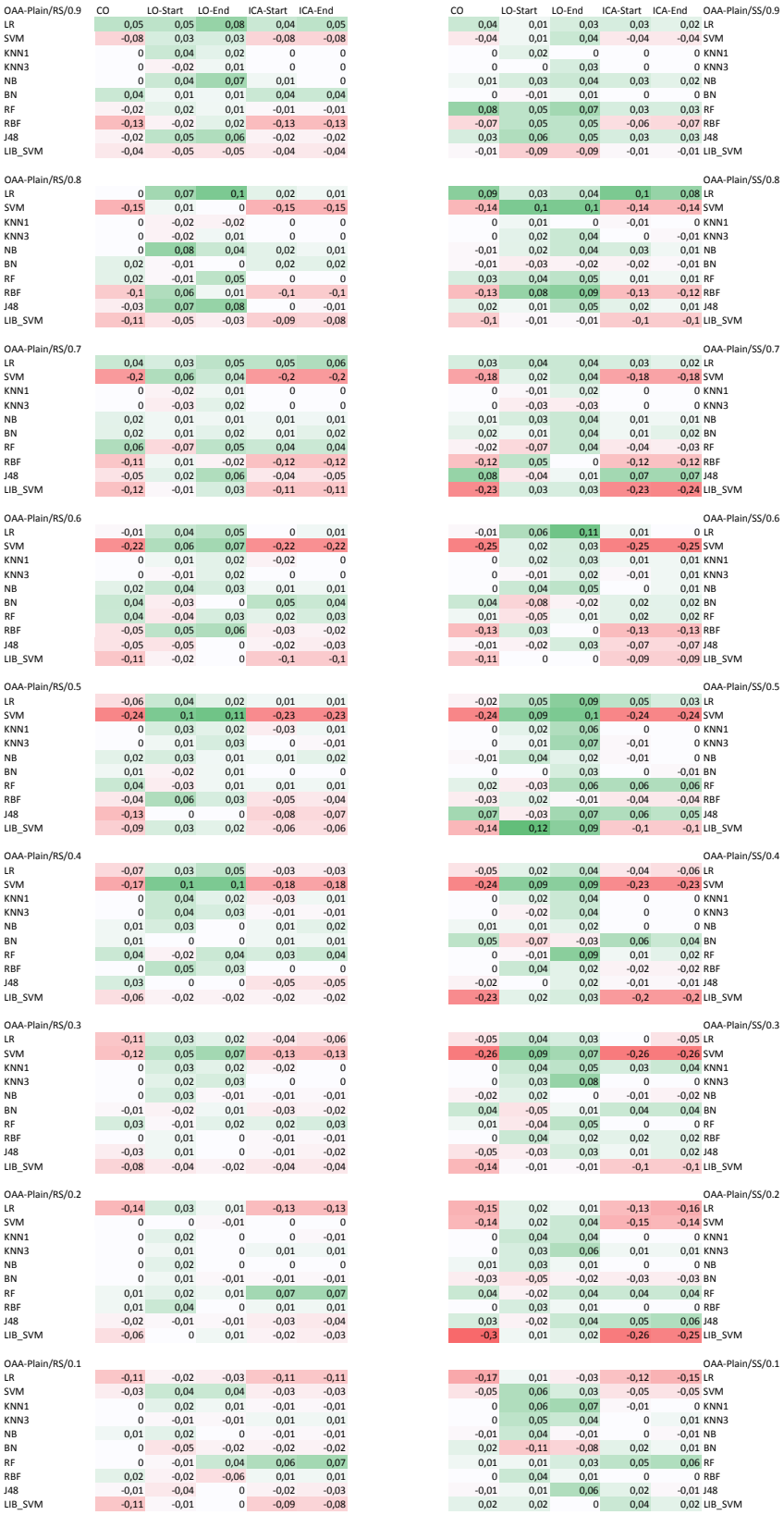
					OAA-Plain/SS/0.1
-0,05	0,05	0,05	-0,01	0	LR
-0,11	0,01	-0,04	-0,1	-0,11	SVM
0	0,11	0,02	0,01	0	KNN1
0	0,11	0,04	0,01	0,01	KNN3
0	0,03	0,03	0,02	-0,01	NB
0	0,08	-0,02	0,02	-0,01	BN
0,05	0,14	0,07	0,09	0,06	RF
-0,07	0,07	0,04	-0,06	-0,06	RBF
0	0,15	0,14	0,06	0,07	J48
-0,01	0,03	-0,04	-0,23	-0,27	LIB_SVM

Figure 6.9 : Citeseer OAA-Plain comparison.



OAA-Plain/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End	CO	LO-Start	LO-End	ICA-Start	ICA-End	OAA-Plain/SS/0.9
LR	0	0,08	0,03	0,04	0	-0,02	0,04	0,12	-0,01	-0,04	LR
SVM	-0,16	0	-0,02	-0,15	-0,15	-0,11	0,03	0,05	-0,09	-0,06	SVM
KNN1	0	0,08	-0,03	0,01	0	0	0,05	0,02	0	0	KNN1
KNN3	0	0,11	-0,02	0,01	0,02	0	0,07	-0,03	0,01	0,02	KNN3
NB	0,02	0,02	0	0	-0,08	0	-0,01	-0,03	-0,04	-0,06	NB
BN	0,04	0,04	0,01	0,09	0,07	0,04	0,03	-0,01	0,15	0,01	BN
RF	0,03	0,08	0,05	0,07	0,08	0,02	0,05	0	0,06	0,04	RF
RBF	-0,11	0,04	-0,03	-0,1	-0,11	-0,07	0,04	0,11	-0,08	-0,08	RBF
J48	-0,03	0,1	0,05	0,03	0,02	0	0,17	0,08	0,17	0,11	J48
LIB_SVM	-0,14	0,03	0	-0,14	-0,14	-0,15	0,01	0,02	-0,15	-0,15	LIB_SVM
OAA-Plain/RS/0.8											OAA-Plain/SS/0.8
LR	0,02	0,13	0,09	0,03	-0,02	0	0,1	0,04	0,06	0,03	LR
SVM	-0,19	-0,04	-0,06	-0,18	-0,19	-0,17	0,03	0,02	-0,14	-0,16	SVM
KNN1	0	0,14	0,04	0,01	0	0	0,04	-0,01	0,01	0,01	KNN1
KNN3	0	0,15	0,03	0,02	0,03	0	0,05	-0,01	0,01	0,02	KNN3
NB	0,01	0,03	-0,02	-0,01	-0,1	0	0,01	0	-0,03	-0,13	NB
BN	0,02	0,13	0,02	0,12	0,05	0,02	0,03	-0,05	0,15	0,02	BN
RF	0,04	0,12	0,07	0,1	0,09	0,04	0,1	0,03	0,07	0,06	RF
RBF	-0,07	0,12	0,03	-0,07	-0,07	-0,04	0,08	0,07	-0,06	-0,05	RBF
J48	-0,05	0,13	0,11	0,09	0,11	0,01	0,07	0,02	0,13	0,12	J48
LIB_SVM	-0,14	0,05	0,01	-0,13	-0,14	-0,07	0	-0,04	-0,08	-0,08	LIB_SVM
OAA-Plain/RS/0.7											OAA-Plain/SS/0.7
LR	-0,06	0,14	0,14	-0,08	-0,16	-0,04	0,18	0,11	-0,02	-0,08	LR
SVM	-0,18	-0,03	-0,09	-0,18	-0,18	-0,15	0,04	0	-0,13	-0,14	SVM
KNN1	0	0,16	0,11	0,01	0	0	0,05	0,01	0,01	0,01	KNN1
KNN3	0	0,16	0,1	0,03	0,03	0	0,07	-0,08	0,01	0,01	KNN3
NB	0,01	0,02	-0,01	0,01	-0,08	0	0,02	-0,04	-0,06	-0,17	NB
BN	0,02	0,12	0,02	0,1	0,03	0,02	0,01	-0,06	0,1	0,01	BN
RF	0,05	0,18	0,14	0,11	0,09	0,04	0,11	0,03	0,1	0,1	RF
RBF	-0,08	0,14	0,04	-0,07	-0,07	-0,05	0,12	0,05	-0,07	-0,07	RBF
J48	-0,03	0,15	0,12	0,1	0,13	0	0,08	0,04	0,13	0,12	J48
LIB_SVM	-0,15	0,04	-0,03	-0,11	-0,12	-0,09	0,02	-0,01	-0,1	-0,12	LIB_SVM
OAA-Plain/RS/0.6											OAA-Plain/SS/0.6
LR	-0,14	0,11	0,11	-0,13	-0,2	-0,11	0,19	0,17	-0,09	-0,11	LR
SVM	-0,18	-0,02	-0,07	-0,17	-0,18	-0,12	0,03	0,04	-0,11	-0,11	SVM
KNN1	0	0,16	0,13	0	-0,01	0	0,1	0,03	0,01	0,01	KNN1
KNN3	0	0,15	0,12	0,04	0,04	0	0,11	0,01	0,03	0,02	KNN3
NB	0,01	0,01	-0,05	0,03	-0,06	0	0,02	-0,03	-0,03	-0,13	NB
BN	0,02	0,14	0,02	0,08	0,02	0,01	0,1	-0,06	0,09	0,01	BN
RF	0,06	0,17	0,16	0,09	0,07	0,05	0,19	0,13	0,09	0,08	RF
RBF	-0,04	0,12	-0,02	-0,06	-0,07	-0,06	0,15	0,06	-0,06	-0,07	RBF
J48	-0,04	0,19	0,21	0,06	0,05	-0,03	0,13	0,07	0,19	0,23	J48
LIB_SVM	-0,15	0,04	-0,03	-0,1	-0,1	-0,15	0,04	-0,03	-0,14	-0,15	LIB_SVM
OAA-Plain/RS/0.5											OAA-Plain/SS/0.5
LR	-0,14	0,09	0,08	-0,16	-0,2	-0,11	0,14	0,12	-0,06	-0,08	LR
SVM	-0,18	-0,03	-0,11	-0,17	-0,17	-0,12	-0,01	-0,04	-0,12	-0,13	SVM
KNN1	0	0,16	0,14	0	-0,01	0	0,14	0,09	0,02	0,02	KNN1
KNN3	0	0,15	0,18	0,03	0,04	0	0,15	0,02	0,03	0,03	KNN3
NB	0,01	0,02	-0,04	0,04	-0,04	0	0,02	-0,05	-0,01	-0,11	NB
BN	0,02	0,15	-0,01	0,06	0,02	0,02	0,15	0,06	0,1	0,03	BN
RF	0,06	0,17	0,13	0,12	0,07	0,05	0,17	0,1	0,11	0,1	RF
RBF	-0,05	0,14	0,04	-0,05	-0,05	-0,03	0,18	0,07	-0,04	-0,04	RBF
J48	-0,03	0,22	0,28	0,12	0,12	-0,03	0,17	0,12	0,15	0,14	J48
LIB_SVM	-0,15	0,04	-0,05	-0,1	-0,11	-0,15	0,07	-0,02	-0,1	-0,12	LIB_SVM
OAA-Plain/RS/0.4											OAA-Plain/SS/0.4
LR	-0,16	0,08	0,07	-0,13	-0,14	-0,12	0,08	0,11	-0,06	-0,08	LR
SVM	-0,17	-0,01	-0,07	-0,16	-0,16	-0,14	0,01	-0,05	-0,12	-0,14	SVM
KNN1	0	0,15	0,14	0,01	0,01	0	0,15	0,08	0,01	0,01	KNN1
KNN3	0	0,16	0,18	0,03	0,04	0	0,15	0,04	0,03	0,03	KNN3
NB	0,01	0,02	-0,03	0,04	-0,02	0	0,04	-0,03	0,02	-0,07	NB
BN	0,02	0,17	0,05	0,04	0,02	0,01	0,2	0,06	0,09	0,03	BN
RF	0,06	0,18	0,15	0,11	0,09	0,06	0,18	0,14	0,09	0,08	RF
RBF	-0,03	0,15	0,05	-0,04	-0,04	-0,03	0,21	0,02	-0,04	-0,04	RBF
J48	-0,03	0,21	0,24	0,12	0,15	-0,03	0,19	0,13	0,17	0,16	J48
LIB_SVM	-0,16	0,05	-0,03	-0,11	-0,12	-0,14	0,07	0	-0,07	-0,07	LIB_SVM
OAA-Plain/RS/0.3											OAA-Plain/SS/0.3
LR	-0,22	0,06	0,05	-0,2	-0,21	-0,2	0,12	0,08	-0,19	-0,21	LR
SVM	-0,15	0,03	-0,03	-0,15	-0,15	-0,14	0,02	-0,06	-0,13	-0,14	SVM
KNN1	0	0,16	0,14	0,01	0,01	0	0,2	0,18	0	0	KNN1
KNN3	0	0,15	0,16	0,04	0,05	0	0,19	0,18	0,04	0,04	KNN3
NB	0	0,04	0	0,04	-0,01	0,01	0,06	-0,03	0,03	-0,04	NB
BN	0,02	0,16	0,06	0,04	0,01	0,01	0,21	0,08	0,07	0,03	BN
RF	0,04	0,2	0,16	0,08	0,08	0,03	0,22	0,18	0,11	0,09	RF
RBF	-0,03	0,16	0,07	-0,03	-0,03	-0,02	0,23	0,05	-0,03	-0,04	RBF
J48	-0,03	0,23	0,25	0,14	0,14	-0,03	0,25	0,23	0,17	0,14	J48
LIB_SVM	-0,16	0,04	-0,03	-0,09	-0,11	-0,14	0,04	-0,05	-0,07	-0,08	LIB_SVM
OAA-Plain/RS/0.2											OAA-Plain/SS/0.2
LR	-0,27	0,05	0,03	-0,21	-0,22	-0,22	0,07	0,04	-0,2	-0,21	LR
SVM	-0,14	0,01	-0,03	-0,13	-0,14	-0,13	0,06	0,05	-0,13	-0,14	SVM
KNN1	0	0,11	0,08	0	-0,01	0	0,19	0,16	0	0	KNN1
KNN3	0	0,11	0,09	0,04	0,04	0	0,18	0,17	0,04	0,04	KNN3
NB	0	0,04	0,03	0,01	0	0,01	0,07	0	0,04	-0,02	NB
BN	0,01	0,11	0,02	0,02	0,01	0,02	0,16	-0,01	0,05	0,01	BN
RF	0,05	0,15	0,12	0,09	0,07	0,05	0,2	0,13	0,12	0,1	RF
RBF	-0,01	0,14	0,07	-0,01	-0,02	-0,01	0,22	0,09	-0,01	-0,01	RBF
J48	-0,03	0,18	0,17	0,12	0,1	-0,04	0,25	0,29	0,17	0,18	J48
LIB_SVM	-0,17	0,01	-0,04	-0,06	-0,08	-0,14	0,03	-0,05	-0,07	-0,08	LIB_SVM
OAA-Plain/RS/0.1											OAA-Plain/SS/0.1
LR	-0,23	0,03	0,01	-0,19	-0,19	-0,22	0,05	0,03	-0,2	-0,21	LR
SVM	-0,13	0,03	0,02	-0,13	-0,13	-0,14	0,08	0,05	-0,13	-0,14	SVM
KNN1	0	0,07	0,04	0	0	0	0,15	0,11	0,01	0	KNN1
KNN3	0	0,08	0,05	0,05	0,06	0	0,15	0,13	0,05	0,04	KNN3
NB	0	0,05	0,06	0,02	0,01	0,01	0,03	-0,01	0,04	-0,01	NB
BN	0	0,06	0,04	0	0	0	0,16	0,03	0,03	0	BN
RF	0,07	0,09	0,07	0,06	0,05	0,05	0,2	0,16	0,09	0,08	RF
RBF	0,02	0,08	0,04	0,02	0,02	-0,01	0,16	0,09	-0,01	-0,02	RBF
J48	-0,02	0,12	0,1	0,09	0,07	-0,02	0,24	0,22	0,16	0,15	J48
LIB_SVM	-0,17	0	-0,05	-0,05	-0,06	-0,14	0,03	-0,05	-0,06	-0,07	LIB_SVM

Figure 6.10 : Cora OAA-Plain comparison.



OAA-Plain/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End
LR	0,05	0,05	0,08	0,04	0,05
SVM	-0,08	0,03	0,03	-0,08	-0,08
KNN1	0	0,04	0,02	0	0
KNN3	0	-0,02	0,01	0	0
NB	0	0,04	0,07	0,01	0
BN	0,04	0,01	0,01	0,04	0,04
RF	-0,02	0,02	0,01	-0,01	-0,01
RBF	-0,13	-0,02	0,02	-0,13	-0,13
J48	-0,02	0,05	0,06	-0,02	-0,02
LIB_SVM	-0,04	-0,05	-0,05	-0,04	-0,04

CO	LO-Start	LO-End	ICA-Start	ICA-End	OAA-Plain/SS/0.9
0,04	0,01	0,03	0,03	0,02	LR
-0,04	0,01	0,04	-0,04	-0,04	SVM
0	0,02	0	0	0	KNN1
0	0	0,03	0	0	KNN3
0,01	0,03	0,04	0,03	0,02	NB
0	-0,01	0,01	0	0	BN
0,08	0,05	0,07	0,03	0,03	RF
-0,07	0,05	0,05	-0,06	-0,07	RBF
0,03	0,06	0,05	0,03	0,03	J48
-0,01	-0,09	-0,09	-0,01	-0,01	LIB_SVM

OAA-Plain/RS/0.8					
LR	0	0,07	0,1	0,02	0,01
SVM	-0,15	0,01	0	-0,15	-0,15
KNN1	0	-0,02	-0,02	0	0
KNN3	0	-0,02	0,01	0	0
NB	0	0,08	0,04	0,02	0,01
BN	0,02	-0,01	0	0,02	0,02
RF	0,02	-0,01	0,05	0	0
RBF	-0,1	0,06	0,01	-0,1	-0,1
J48	-0,03	0,07	0,08	0	-0,01
LIB_SVM	-0,11	-0,05	-0,03	-0,09	-0,08

					OAA-Plain/SS/0.8
0,09	0,03	0,04	0,1	0,08	LR
-0,14	0,1	0,1	-0,14	-0,14	SVM
0	0,01	0	-0,01	0	KNN1
0	0,02	0,04	0	-0,01	KNN3
-0,01	0,02	0,04	0,03	0,01	NB
-0,01	-0,03	-0,02	-0,02	-0,01	BN
0,03	0,04	0,05	0,01	0,01	RF
-0,13	0,08	0,09	-0,13	-0,12	RBF
0,02	0,01	0,05	0,02	0,01	J48
-0,1	-0,01	-0,01	-0,1	-0,1	LIB_SVM

OAA-Plain/RS/0.7					
LR	0,04	0,03	0,05	0,05	0,06
SVM	-0,2	0,06	0,04	-0,2	-0,2
KNN1	0	-0,02	0,01	0	0
KNN3	0	-0,03	0,02	0	0
NB	0,02	0,01	0,01	0,01	0,01
BN	0,02	0,01	0,02	0,01	0,02
RF	0,06	-0,07	0,05	0,04	0,04
RBF	-0,11	0,01	-0,02	-0,12	-0,12
J48	-0,05	0,02	0,06	-0,04	-0,05
LIB_SVM	-0,12	-0,01	0,03	-0,11	-0,11

					OAA-Plain/SS/0.7
0,03	0,04	0,04	0,03	0,02	LR
-0,18	0,02	0,04	-0,18	-0,18	SVM
0	-0,01	0,02	0	0	KNN1
0	-0,03	-0,03	0	0	KNN3
0,01	0,03	0,04	0,01	0,01	NB
0,02	0,01	0,04	0,01	0,02	BN
-0,02	-0,07	0,04	-0,04	-0,03	RF
-0,12	0,05	0	-0,12	-0,12	RBF
0,08	-0,04	0,01	0,07	0,07	J48
-0,23	0,03	0,03	-0,23	-0,24	LIB_SVM

OAA-Plain/RS/0.6					
LR	-0,01	0,04	0,05	0	0,01
SVM	-0,22	0,06	0,07	-0,22	-0,22
KNN1	0	0,01	0,02	-0,02	0
KNN3	0	-0,01	0,02	0	0
NB	0,02	0,04	0,03	0,01	0,01
BN	0,04	-0,03	0	0,05	0,04
RF	0,04	-0,04	0,03	0,02	0,03
RBF	-0,05	0,05	0,06	-0,03	-0,02
J48	-0,05	-0,05	0	-0,02	-0,03
LIB_SVM	-0,11	-0,02	0	-0,1	-0,1

					OAA-Plain/SS/0.6
-0,01	0,06	0,11	0,01	0	LR
-0,25	0,02	0,03	-0,25	-0,25	SVM
0	0,02	0,03	0,01	0,01	KNN1
0	-0,01	0,02	-0,01	0,01	KNN3
0	0,04	0,05	0	0,01	NB
0,04	-0,08	-0,02	0,02	0,02	BN
0,01	-0,05	0,01	0,02	0,02	RF
-0,13	0,03	0	-0,13	-0,13	RBF
-0,01	-0,02	0,03	-0,07	-0,07	J48
-0,11	0	0	-0,09	-0,09	LIB_SVM

OAA-Plain/RS/0.5					
LR	-0,06	0,04	0,02	0,01	0,01
SVM	-0,24	0,1	0,11	-0,23	-0,23
KNN1	0	0,03	0,02	-0,03	0,01
KNN3	0	0,01	0,03	0	-0,01
NB	0,02	0,03	0,01	0,01	0,02
BN	0,01	-0,02	0,01	0	0
RF	0,04	-0,03	0,01	0,01	0,01
RBF	-0,04	0,06	0,03	-0,05	-0,04
J48	-0,13	0	0	-0,08	-0,07
LIB_SVM	-0,09	0,03	0,02	-0,06	-0,06

					OAA-Plain/SS/0.5
-0,02	0,05	0,09	0,05	0,03	LR
-0,24	0,09	0,1	-0,24	-0,24	SVM
0	0,02	0,06	0	0	KNN1
0	0,01	0,07	-0,01	0	KNN3
-0,01	0,04	0,02	-0,01	0	NB
0	0	0,03	0	-0,01	BN
0,02	-0,03	0,06	0,06	0,06	RF
-0,03	0,02	-0,01	-0,04	-0,04	RBF
0,07	-0,03	0,07	0,06	0,05	J48
-0,14	0,12	0,09	-0,1	-0,1	LIB_SVM

OAA-Plain/RS/0.4					
LR	-0,07	0,03	0,05	-0,03	-0,03
SVM	-0,17	0,1	0,1	-0,18	-0,18
KNN1	0	0,04	0,02	-0,03	0,01
KNN3	0	0,04	0,03	-0,01	-0,01
NB	0,01	0,03	0	0,01	0,02
BN	0,01	0	0	0,01	0,01
RF	0,04	-0,02	0,04	0,03	0,04
RBF	0	0,05	0,03	0	0
J48	0,03	0	0	-0,05	-0,05
LIB_SVM	-0,06	-0,02	-0,02	-0,02	-0,02

					OAA-Plain/SS/0.4
-0,05	0,02	0,04	-0,04	-0,06	LR
-0,24	0,09	0,09	-0,23	-0,23	SVM
0	0,02	0,04	0	0	KNN1
0	-0,02	0,04	0	0	KNN3
0,01	0,01	0,02	0	0	NB
0,05	-0,07	-0,03	0,06	0,04	BN
0	-0,01	0,09	0,01	0,02	RF
0	0,04	0,02	-0,02	-0,02	RBF
-0,02	0	0,02	-0,01	-0,01	J48
-0,23	0,02	0,03	-0,2	-0,2	LIB_SVM

OAA-Plain/RS/0.3					
LR	-0,11	0,03	0,02	-0,04	-0,06
SVM	-0,12	0,05	0,07	-0,13	-0,13
KNN1	0	0,03	0,02	-0,02	0
KNN3	0	0,02	0,03	0	0
NB	0	0,03	-0,01	-0,01	-0,01
BN	-0,01	-0,02	0,01	-0,03	-0,02
RF	0,03	-0,01	0,02	0,02	0,03
RBF	0	0,01	0	-0,01	-0,01
J48	-0,03	0,01	0	-0,01	-0,02
LIB_SVM	-0,08	-0,04	-0,02	-0,04	-0,04

					OAA-Plain/SS/0.3
-0,05	0,04	0,03	0	-0,05	LR
-0,26	0,09	0,07	-0,26	-0,26	SVM
0	0,04	0,05	0,03	0,04	KNN1
0	0,03	0,08	0	0	KNN3
-0,02	0,02	0	-0,01	-0,02	NB
0,04	-0,05	0,01	0,04	0,04	BN
0,01	-0,04	0,05	0	0	RF
0	0,04	0,02	0,02	0,02	RBF
-0,05	-0,03	0,03	0,01	0,02	J48
-0,14	-0,01	-0,01	-0,1	-0,1	LIB_SVM

OAA-Plain/RS/0.2					
LR	-0,14	0,03	0,01	-0,13	-0,13
SVM	0	0	-0,01	0	0
KNN1	0	0,02	0	0	-0,01
KNN3	0	0,01	0	0,01	0,01
NB	0	0,02	0	0	0
BN	0	0,01	-0,01	-0,01	-0,01
RF	0,01	0,02	0,01	0,07	0,07
RBF	0,01	0,04	0	0,01	0,01
J48	-0,02	-0,01	-0,01	-0,03	-0,04
LIB_SVM	-0,06	0	0,01	-0,02	-0,03

					OAA-Plain/SS/0.2
-0,15	0,02	0,01	-0,13	-0,16	LR
-0,14	0,02	0,04	-0,15	-0,14	SVM
0	0,04	0,04	0	0	KNN1
0	0,03	0,06	0,01	0,01	KNN3
0,01	0,03	0,01	0	0	NB
-0,03	-0,05	-0,02	-0,03	-0,03	BN
0,04	-0,02	0,04	0,04	0,04	RF
0	0,03	0,01	0	0	RBF
0,03	-0,02	0,04	0,05	0,06	J48
-0,3	0,01	0,02	-0,26	-0,25	LIB_SVM

OAA-Plain/RS/0.1					
LR	-0,11	-0,02	-0,03	-0,11	-0,11
SVM	-0,03	0,04	0,04	-0,03	-0,03
KNN1	0	0,02	0,01	-0,01	-0,01
KNN3	0	-0,01	-0,01	0,01	0,01
NB	0,01	0,02	0	-0,01	-0,01
BN	0	-0,05	-0,02	-0,02	-0,02
RF	0	-0,01	0,04	0,06	0,07
RBF	0,02	-0,02	-0,06	0,01	0,01
J48	-0,01	-0,04	0	-0,02	-0,03
LIB_SVM	-0,11	-0,01	0	-0,09	-0,08

					OAA-Plain/SS/0.1
-0,17	0,01	-0,03	-0,12	-0,15	LR
-0,05	0,06	0,03	-0,05	-0,05	SVM
0	0,06	0,07	-0,01	0	KNN1
0	0,05	0,04	0	0,01	KNN3
-0,01	0,04	-0,01	0	-0,01	NB
0,02	-0,11	-0,08	0,02	0,01	BN
0,01	0,01	0,03	0,05	0,06	RF
0	0,04	0,01	0	0	RBF
-0,01	0,01	0,06	0,02	-0,01	J48
0,02	0,02	0	0,04	0,02	LIB_SVM

Figure 6.11 : WebKb OAA-Plain comparison.

OAA-Plain/RS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End
LR	0,01	0,02	0,02	0	-0,04
SVM	0	0,01	-0,02	-0,01	-0,01
KNN1	0	0,02	0,04	0	0
KNN3	0	0,01	0,01	0	0,01
NB	0	0	-0,02	-0,01	-0,02
BN	0,01	0,02	0	0,05	0,04
RF	0,02	0,04	0,01	0,01	0,02
RBF	0	0,04	0,04	-0,01	-0,01
J48	0,01	0,03	0,01	0	0,02
LIB_SVM	-0,01	0,01	0	-0,01	-0,01

OAA-Plain/SS/0.9	CO	LO-Start	LO-End	ICA-Start	ICA-End
LR	0	-0,07	-0,05	-0,02	-0,01
SVM	-0,03	-0,03	-0,03	-0,03	-0,04
KNN1	0	-0,05	-0,05	0	0
KNN3	0	-0,07	-0,08	0,01	0
NB	0,01	0,02	0,02	-0,03	-0,02
BN	0	-0,04	-0,04	-0,04	-0,04
RF	0,02	-0,02	0,02	0,04	0,04
RBF	0	-0,03	-0,05	0	0
J48	0,03	0,06	0,06	0,08	0,08
LIB_SVM	-0,02	-0,06	-0,04	-0,02	-0,02

OAA-Plain/RS/0.8					
LR	0	0,01	0,03	0,05	0,04
SVM	-0,05	0,02	0,02	-0,05	-0,06
KNN1	0	-0,04	-0,02	0	0
KNN3	0	-0,02	0,02	0	0
NB	0	-0,03	-0,02	-0,06	-0,04
BN	0,01	0,03	-0,02	0,03	0,01
RF	0,01	0,03	0,02	0,01	0,02
RBF	0	0,02	0	-0,01	-0,01
J48	0,02	0,07	0,07	0,04	0,04
LIB_SVM	-0,03	0,04	0,01	-0,04	-0,04

OAA-Plain/SS/0.8					
LR	-0,01	-0,05	-0,04	0,03	0,02
SVM	0	-0,02	0,02	0,01	0,01
KNN1	0	-0,02	0,02	0	0
KNN3	0	-0,04	-0,03	0	0
NB	0	0	0,01	-0,01	-0,01
BN	0,05	0,01	0,03	0,05	0,06
RF	0,01	0,02	0,07	0,04	0,04
RBF	0,03	-0,02	-0,02	0,04	0,03
J48	-0,01	-0,05	-0,02	0,04	0,08
LIB_SVM	-0,05	0	-0,02	-0,05	-0,05

OAA-Plain/RS/0.7					
LR	0,01	0,06	0,06	0	-0,03
SVM	-0,08	-0,01	-0,02	-0,07	-0,08
KNN1	0	0,02	0,02	0	0,01
KNN3	0	0,02	0,03	0	0
NB	0	-0,08	-0,07	-0,04	-0,06
BN	-0,01	0,09	0,08	0,05	0,03
RF	0,03	0,05	0,02	0,04	0,03
RBF	0	0,1	0,1	0	0
J48	-0,01	0,07	0,07	0,04	0,04
LIB_SVM	-0,01	0,01	0,01	-0,02	-0,02

OAA-Plain/SS/0.7					
LR	0,01	-0,04	-0,09	-0,01	-0,04
SVM	0,02	0,09	0,15	0,03	0,04
KNN1	0	-0,06	-0,01	0	0,01
KNN3	0	-0,08	-0,06	0	0
NB	0	0	0,01	-0,02	0,01
BN	0,04	0,04	0,06	0,03	0,09
RF	0,03	-0,04	0,14	0,04	0,03
RBF	0,04	-0,05	-0,05	0,05	0,04
J48	-0,01	-0,05	-0,01	0,03	0,13
LIB_SVM	-0,04	0,04	0,03	-0,04	-0,04

OAA-Plain/RS/0.6					
LR	-0,01	0,03	0,03	0,02	0,02
SVM	-0,06	-0,02	-0,03	-0,05	-0,06
KNN1	0	0	0,02	0	0
KNN3	0	0,01	0,05	0	0
NB	0	0,02	0,04	-0,03	-0,02
BN	0,01	0,06	0,04	0,05	0,04
RF	0,03	0,07	0,06	0,05	0,05
RBF	0,03	0,02	0	0,04	0,04
J48	0	0,05	0,06	0,05	0,05
LIB_SVM	-0,03	-0,02	-0,02	-0,11	-0,12

OAA-Plain/SS/0.6					
LR	-0,04	0,02	-0,02	0,01	0,01
SVM	-0,02	0,04	0,11	0,03	0,1
KNN1	0	-0,03	0,01	0	0,01
KNN3	0	-0,04	0,01	0	0
NB	0,01	-0,03	-0,04	-0,03	-0,01
BN	0,03	0,08	0,1	0,03	0,01
RF	0,02	0	0,03	0,06	0,06
RBF	0,06	-0,04	-0,02	0,07	0,07
J48	-0,04	-0,05	0,11	0,03	0,19
LIB_SVM	-0,07	-0,04	-0,13	-0,07	-0,07

OAA-Plain/RS/0.5					
LR	-0,02	0,05	0,04	0	-0,01
SVM	-0,06	-0,05	-0,08	-0,05	-0,05
KNN1	0	0,04	0,02	0	0
KNN3	0	0,04	0,04	0	0
NB	0	0,04	0,04	-0,01	-0,01
BN	0,01	0,07	0,04	0,03	0,03
RF	0,04	0,07	0,04	0,06	0,05
RBF	0,02	0,06	0,03	0,03	0,03
J48	0	0,1	0,1	0,02	0,03
LIB_SVM	-0,02	0,01	0	-0,01	-0,01

OAA-Plain/SS/0.5					
LR	-0,04	0,06	0,07	0,01	-0,05
SVM	-0,1	-0,08	-0,1	-0,08	-0,11
KNN1	0	-0,05	-0,01	0	0
KNN3	0	-0,04	-0,02	-0,01	-0,01
NB	0,01	-0,02	-0,12	-0,02	-0,1
BN	0,03	0,03	0,06	0,11	0,11
RF	0,02	0,05	0,03	0,09	0,09
RBF	0,06	-0,04	-0,02	0,05	0,06
J48	-0,01	0,13	0,02	0,07	0,25
LIB_SVM	-0,15	0,01	0,01	-0,16	-0,16

OAA-Plain/RS/0.4					
LR	-0,05	0,04	0,02	-0,03	-0,02
SVM	-0,06	-0,07	-0,08	-0,05	-0,05
KNN1	0	0,04	0,01	0	0
KNN3	0	0,04	0,03	0	-0,01
NB	0	0,03	0,02	-0,04	-0,06
BN	0,02	0,08	0,08	0,04	0,04
RF	0,04	0,08	0,06	0,05	0,04
RBF	0,02	0,09	0,06	0,04	0,04
J48	0,01	0,06	0,05	0,06	0,1
LIB_SVM	-0,01	-0,03	-0,06	-0,05	-0,06

OAA-Plain/SS/0.4					
LR	-0,09	0,05	0,02	-0,08	-0,16
SVM	-0,1	-0,09	-0,16	-0,09	-0,1
KNN1	0	0,01	0,01	0,01	0
KNN3	0	-0,03	-0,03	0	0
NB	0,01	0,03	-0,03	0,03	-0,03
BN	0,04	0,1	0,11	0,12	0,16
RF	0,04	0,1	0,05	0,07	0,07
RBF	0,12	0,04	-0,01	0,11	0,11
J48	0	0,03	0,06	0,13	0,23
LIB_SVM	-0,2	0,04	0,01	-0,24	-0,26

OAA-Plain/RS/0.3					
LR	-0,08	0,03	0,01	-0,08	-0,08
SVM	-0,06	-0,09	-0,09	-0,06	-0,06
KNN1	0	0,03	0,01	0	0
KNN3	0	0,04	0,03	0	-0,01
NB	0	0,04	0,06	-0,04	-0,04
BN	0	0,07	0,06	0,04	0,03
RF	0,03	0,07	0,05	0,06	0,06
RBF	0,05	0,08	0,05	0,04	0,04
J48	0	0,04	0,05	0,05	0,08
LIB_SVM	-0,06	-0,04	-0,05	-0,17	-0,18

OAA-Plain/SS/0.3					
LR	-0,09	0,06	0,06	-0,09	-0,16
SVM	-0,1	-0,12	-0,19	-0,08	-0,09
KNN1	0	0,02	0,05	0,01	0,01
KNN3	0	-0,01	0,03	0	0
NB	0	0	-0,06	-0,02	-0,04
BN	0,04	0,15	0,11	0,13	0,17
RF	0,04	0,13	0,13	0,07	0,04
RBF	0,11	0,08	-0,02	0,1	0,1
J48	0,01	0,01	0,11	0,14	0,2
LIB_SVM	-0,21	0,03	-0,02	-0,24	-0,29

OAA-Plain/RS/0.2					
LR	-0,12	0,03	0	-0,12	-0,13
SVM	-0,06	-0,11	-0,12	-0,06	-0,07
KNN1	0	0,03	0,04	0,01	0
KNN3	0	0,04	0,02	0	0
NB	0	0,01	0,02	0	-0,01
BN	0	0,09	0,09	0,03	0,03
RF	0,05	0,05	0,04	0,06	0,05
RBF	0,1	0,04	0,02	0,1	0,1
J48	0,01	0,04	0,05	0,07	0,08
LIB_SVM	-0,09	-0,06	-0,07	-0,16	-0,16

OAA-Plain/SS/0.2					
LR	-0,16	0,05	0,03	-0,13	-0,19
SVM	-0,1	-0,07	-0,13	-0,08	-0,08
KNN1	0	0,05	0,06	0,01	0,01
KNN3	0	0,03	0,09	0,01	0
NB	0	-0,05	-0,08	-0,01	-0,03
BN	0,01	0,14	0,06	0,06	0,07
RF	0,06	0,11	0,13	0,06	0,06
RBF	0,07	0,09	0,08	0,06	0,06
J48	0	0,11	0,16	0,08	0,09
LIB_SVM	-0,2	0,03	-0,03	-0,08	-0,09

OAA-Plain/RS/0.1					
LR	-0,11	0,01	0	-0,12	-0,12
SVM	-0,06	-0,1	-0,11	-0,05	-0,05
KNN1	0	0,03	0,02	0	0,01
KNN3	0	0,02	0	0	0
NB	-0,01	0,02	0,01	0	0
BN	-0,01	0,06	0,05	0	0,01
RF	0,05	0,03	0,03	0,08	0,07
RBF	0,1	0,04	0,01	0,11	0,11
J48	0,01	0,03	0,01	0,05	0,05
LIB_SVM	-0,17	0	0	-0,09	-0,08

OAA-Plain/SS/0.1					
LR	-0,12	0,05	0,06	-0,13	-0,17
SVM	-0,08	-0,03	-0,05	-0,07	-0,07
KNN1	0	0,12	0,11	0,02	0,03
KNN3	0	0,11	0,14	0,01	0,01
NB	0,01	0,03	0,03	0,01	-0,02
BN	0	0,12	0,16	0,04	0,03
RF	0,02	0,1	0,13	0,12	0,09
RBF	0,07	0,11	0,08	0,06	0,06
J48	0,04	0,08	0,15	0,05	0,13
LIB_SVM	-0,2	0,08	0	-0,02	-0,06

Figure 6.12 : HepTh OAA-Plain comparison.

7. CONCLUSIONS AND FUTURE RESEARCH

In this thesis, first, we compared certain graph properties of the datasets used in the context of this thesis, namely Citeseer, Cora, WebKb, HepTh and Synthetic datasets, to understand similarities and differences between these datasets. In addition to graph properties existing in the literature, we introduced some new graph properties such as local alpha and local beta, which depend on the neighbors' classifier accuracy and neighbors' homophily respectively. We also calculated the correlations between the graph properties.

We found out that, as expected and pointed out by previous authors, homophily and accuracy are highly correlated. Homophily and entropy are negatively correlated as expected. Local alpha and accuracy are highly correlated on some datasets. Local alpha and local beta are positively highly correlated for the link graphs. Local beta and homophily are also correlated as expected.

We can suggest a number of research directions related to graph properties. Class based graph properties can be investigated. That is, graph properties such as degree distribution, clustering coefficient per class basis can be calculated and compared. Exploration of relative graph properties may be another research direction. Since classification accuracy highly depends on the separation of the test/train partitions, whenever a test instance is too far away from the training data it gets very little information from its neighbors. So, for each partitioning of the dataset, the correlations between relative graph properties and accuracy of the classifiers may be explored. One other direction may be calculation of directed graph properties which may help in understanding of the classifier behavior of directed graphs.

In the thesis, in order to benefit from the diversity of the domains (such as content, link) and classifiers, we proposed training classifiers separately on each domain and combining those classifiers. We proposed seven different classifier combination methods to combine classifiers. In order to select the most diverse and useful set of

classifiers for a combination scheme, we used a genetic algorithm based method. We used the validation sets to determine which classifiers should be combined to achieve better performance for the given train/test partition of the dataset. After we determined the classifier set to be combined, we used the same classifier set for the test set. Our experiments on four different datasets, namely Citeseer, Cora, WebKb and HepTh, show that our new method outperforms best of the base classifiers on the datasets used. Our method can also be extended to collective classification scenarios with multiple types of content and link in the future.

Investigation of the effect of using directed link information on classifier performance for collective classification, was another contribution of the thesis. We evaluated performance of collective classification algorithms using different local classifiers on Citeseer, Cora, WebKb and HepTh datasets with random and snowball sampling, using directed and undirected links for different train/test ratios. It has been shown that by using directed graphs, significant performance increase is obtained when link only classifier is used. This useful information also reflects to collective classification (ICA) and we get better results as well. Whenever directed links are used with snowball sampling, the performance of the content only classifier decreased slightly. The reason for that is, snowball sampling is related to links and in case of usage of directed links, growing snowball has less directions to grow, compared to undirected one. Content only for random sampling was not affected by directed links since random sampling is independent of the links. The proposed method to use directed link information, can be extended to any network classification problem where direction information is available.

While it is usually difficult to obtain labels on the whole dataset, features are usually easier to obtain. In this thesis, in order to benefit from this fact that, we introduced a new method of transductive network classification which can use the test node features when training the classifier (Cataltepe et al. 2014). We train our classifier using enriched node features. The enriched node features include, in addition to the node's own features, the aggregated neighbors' features and aggregation of node and neighbor features passed through simple logical operators OR and AND. Enriched features may contain irrelevant or redundant features, which could decrease classifier performance.

Therefore, we employ feature selection to determine whether a feature among the set of enriched features should be used for classifier training or not. The feature selection method used, FCBF#, is a mutual information based, filter type, fast, feature selection method (Senliol et al. 2008).

The methods introduced in the context of this section of the thesis, is an extension of previous work on feature enrichment, which was first introduced in (Senliol 2010). In the previous work, enriched features are constructed as combinations of plain (node's own features), neighbors features and ORed features. In addition to those, we also introduce ANDed features and take into account all major combinations of those enrichment methods as separate cases. It only uses EnrSel method, in which, after enrichment of all node features, feature selection is applied. However we also introduce SelEnr method, in which, feature selection among the node features is done and then those features are enriched. In addition to Logistic Regression which was used in the previous study, we also use Bayes Net as the base classifier.

Experimental results on three different network datasets show that classification accuracies obtained using network enriched and selected features are comparable or better than content only or collective classification.

There are a number of future work directions related to feature enrichment. Other feature enrichment methods based on the characteristics of a specific networked dataset may be explored. Exploration of aggregation methods other than count and average and exploration of neighborhood functions that consider more than just the immediate neighbors may be other research directions.

In this thesis, then, for classification of multi-class networked data, instead of using a single classifier to learn all classes, we used a one-against-all scheme and learned a separate classifier for each class. We extended this one-against-all setting to collective classification also. Although one-against-all classification increases the training and testing time due to the increase in the number of classifiers, experimental results show that OAA content only and collective classification is better than single classifier content only and collective classification. It has been also shown that the benefit of using OAA scheme is strongly dependent to the classifier used. The benefit of

OAA learning becomes more emphasized with increase in homophily or decrease in available training data size.

As a future work, an algorithm for one against one (OAO) classification for networked data may be implemented and OAO and OAA may be compared.

There are also a number of future work directions related to collective classification.

- **Classifier Pooling:** During the execution of collective classification algorithms the same local classifier is used. However it is possible to change the local classifier during the iterative progress.
- **Graph Properties as Features:** It is possible to calculate unlabeled graph properties whenever links are available. For each node, these values can be used as additional the features of the node. A feature selection algorithm can be applied for feature selection of -extended- feature set of the node. The result of the selection should give an idea about the importance of that property for the dataset and these additional features may help to improve classification performance.
- **Stacking of Classifiers:** The method introduced for heterogeneous classifier combination may be extended to stacking of classifiers.

References

A. Bernstein, A., Clearwater, S., Hill, S., Perlich, C., & Provost, F. (2002). Discovering knowledge from relational data extracted from business news. In *Proceedings of the workshop on multi-relational data mining at kdd-2002* (pp. 7–22).

Ahmed, N. K., Neville, J., & Kompella, R. (2012). *Network sampling: From static to streaming graphs.*

Angin, P., & Neville, J. (2008). A shrinkage approach for modeling non-stationary relational autocorrelation. In *Sna/kdd.*

Anthony, G., Gregg, H., & Tshilidzi, M. (2007). Image classification using svms: one-against-one vs one-against-all. *arXiv preprint arXiv:0711.2914*.

Awan, A., Bari, H., Yan, F., Moksong, S., Yang, S., Chowdhury, S., ... Wang, E. (2007). Regulatory network motifs and hotspots of cancer genes in a mammalian cellular signalling network. *IET Syst. Biol.*, *1*(5), 292–297.

Balcan, D., & Erzan, A. (2004). Random model for rna interference yields scale free network. *Eur. Phys. J. B*, *38*, 253–260.

Bilgic, M., & Getoor, L. (2008). Effective label acquisition for collective classification. In Y. Li, B. Liu, & S. Sarawagi (Eds.), *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, las vegas, nevada, USA, august 24-27, 2008* (pp. 43–51). ACM.

Bilgic, M., Mihalkova, L., & Getoor, L. (2010). Active learning for networked data. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 79–86).

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on computational learning theory* (pp. 92–100).

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., ... others (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *Pattern recognition, 1994. vol. 2-conference b: Computer vision & image processing., proceedings of the 12th iapr international. conference on* (Vol. 2, pp. 77–82).

Bulacio, P., Guillaume, S., Tapia, E., & Magdalena, L. (2010). A selection approach for scalable fuzzy integral combination. *Information Fusion*, *11*(2), 208–213.

Cataltepe, Z., Sonmez, A., Baglioglu, K., & Erzan, A. (2011). Collective classification using heterogeneous classifiers. In P. Perner (Ed.), *Machine learning and data mining in pattern recognition* (Vol. 6871, p. 155-169). Springer Berlin Heidelberg.

Cataltepe, Z., Sonmez, A., & Senliol, B. (2014). Feature enrichment and selection for transductive classification on networked data. *Pattern Recognition Letters*, *37*, 41–53.

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In *Proceedings of the acm sigmod international conference on management of data* (p. 307-318).

Chapelle, O., & Zien, A. (2005). Semi–supervised classification by low density separation. In *Proceedings of the international workshop on artificial intelligence and statistics* (pp. 57–64).

Colizza, V., Flammini, A., Serrano, M. A., & Vespignani, A. (2006, February 20). *Detecting rich-club ordering in complex networks* (Vol. 2). Retrieved from `http://arxiv.org/abs/physics/0602134`

Cover, T., Thomas, J., Proakis, J. G., Salehi, M., & Morelos-Zaragoza, R. H. (1991). *Elements of information theory*. Wiley-Interscience, NY.

Cozman, F. G., Cohen, I., & Cirelo, M. (2002). Unlabeled data can degrade classification performance of generative classifiers. In *Flairs conference* (pp. 327–331).

Culp, M., & Michailidis, G. (2008). Graph-based semisupervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *30*(1), 174–179.

Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A. A., & Joshi, A. (2008). Social ties and their relevance to churn in mobile telecom networks. In *Edbt* (p. 668-677).

DiPasquo, M. C. D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the fifteenth national conference on artificial intelligence (aaai), madison, wi* (pp. 509–516).

Dorogovtsev, S. N., & Mendes, J. F. F. (2002). Evolution of networks. In *Advances in physics* (pp. 1079–1187).

Eldardiry, H., & Neville, J. (2012). An analysis of how ensembles of collective classifiers improve predictions in graphs. In *Proceedings of the 21st acm international conference on information and knowledge management* (pp. 225–234).

Erdos, P., Pach, J., Pollack, R., & Tuza, Z. (1989). Radius, diameter, and minimum degree. *Journal of Combinatorial Theory, Series B*, *47*(1), 73–39.

et al., K. M. (2014). *Jgap - java genetic algorithms and genetic programming package.*

Fast, A., & Jensen, D. (2008). Why stacked models perform effective collective classification. In *Eighth ieee international conference on data mining* (pp. 785–790).

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *science*, *315*(5814), 972–976.

Fronczak, A., Fronczak, P., & Holyst, J. A. (2004). Average path length in uncorrelated random networks with hidden variables. *Phy. Rev. E*, *70*(5), 056110, 1-7.

Getoor, L., Friedman, N., Koller, D., & Taskar, B. (2003). Learning probabilistic models of link structure. *The Journal of Machine Learning Research*, *3*, 679–707.

Giles, C., Bollacker, K., & Lawrence, S. (1998). Citeseer: An automatic citation indexing system. In *Acm digital libraries* (pp. 89–98).

Guyon, I., & Elisseeff, A. (2003, March). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, *3*, 1157–1182.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009,

November). The weka data mining software: an update. *SIGKDD Explor. Newsl.*, *11*(1), 10–18.

Hanneman, R. A., & Riddle, M. (2005). *Introduction to social network methods*. University of California, Riverside, CA.

Hein, O., Schwind, M., & Konig, W. (2006). Scale-free networks: The impact of fat tailed degree distribution on diffusion and communication processes. *Wirtschaftsinformatik*, *48*(4), 267–275.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, *13*(2), 415–425.

Jensen, D., & Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. In *In proceedings of the 19th international conference on machine learning* (pp. 259–266). Morgan Kaufmann.

Jensen, D., Neville, J., & Gallagher, B. (2004). Why collective inference improves relational classification. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining* (pp. 593–598).

Ji, M., Sun, Y., Danilevsky, M., Han, J., & Gao, J. (2010). Graph regularized transductive classification on heterogeneous information networks. In *Machine learning and knowledge discovery in databases* (pp. 570–586). Springer.

Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th european conference on machine learning* (pp. 137–142). London, UK, UK: Springer-Verlag. Retrieved from `http://dl.acm.org/citation.cfm?id=645326.649721`

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *In icml* (pp. 290–297).

Kahng, G. O., Oh, E., Kahng, B., & Kim, D. (2003). Betweenness centrality correlation in social networks. *Phys. Rev. E*, *67*, 01710–1.

Knerr, S., Personnaz, L., & Dreyfus, G. (1990). Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing* (pp. 41–50). Springer.

Kou, Z., & Cohen, W. W. (2007). Notes on stacked graphical learning for efficient inference in markov random fields. In *Cmu technical report, cmu-ml-07-101.*

Kuncheva, L. I. (2004). *Combining pattern classifiers: Methods and algorithms*. Wiley-Interscience.

Kuncheva, L. I., & Jain, L. C. (2000). Designing classifier fusion systems by genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, *4*(4), 327–336.

Kuwadekar, A., & Neville, J. (2011). Relational active learning for joint collective classification models. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 385–392).

Leskovec, J., & Faloutsos, C. (2006). Sampling from large graphs. In *In proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining* (p. 631-636).

Lu, Q., & Getoor, L. (2003). Link-based classification. In *Icml* (p. 496-503).

Macskassy, S. A. (2007). Improving learning in networked data by combining explicit and mined links. In *Proceedings of the national conference on artificial intelligence* (Vol. 22, p. 590).

Macskassy, S. A., & Provost, F. (2003). A simple relational classifier. In *Workshop on multi-relational data mining (mrdm-2003)* (p. 64).

Macskassy, S. A., & Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, *8*, 935–983.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, *3*(2), 127-163.

McDowell, L., Gupta, K., & Aha, D. (2009). Cautious collective classification. *Journal of Machine Learning Research*, *10*, 2777-2836.

McDowell, L., Gupta, K., & Aha, D. (2010). Meta-Prediction for Collective Classification.

McDowell, L., Gupta, K. M., & Aha, D. W. (2007). Cautious inference in collective classification. In *Aaai* (pp. 596–601). AAAI Press.

McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 415–444.

Milgram, J., Cheriet, M., Sabourin, R., et al. (2006). "one against one" or "one against all": Which one is better for handwriting recognition with svms? In *Tenth international workshop on frontiers in handwriting recognition.*

Mitchell, M. (1996). An introduction to genetic algorithms, 1996. *PHI Pvt. Ltd., New Delhi*.

Murrugarra-Llerena, N., & de Andrade Lopes, A. (2011). A graph-based bagging. *Proceedings of the CoLISD: Collective Learning and Inference on Structured Data*, 25–36.

Neville, J., Gallagher, B., & Eliassi-Rad, T. (2009). Evaluating statistical tests for within-network classifiers of relational data. In *Icdm* (p. 397-406).

Neville, J., & Jensen, D. (2000). Iterative classification in relational data. In *Proc. aaai-2000 workshop on learning statistical models from relational data* (pp. 13–20).

Neville, J., & Jensen, D. (2008). A bias/variance decomposition for models using collective inference. *Machine Learning*, *73*(1), 87–106.

Newman, M. E. (2000). Models of the small world. *Journal of Statistical Physics*, *101*(3-4), 819–841.

Newman, M. E. J. (2003a). Mixing patterns in networks. *Phy. Rev. E*, *67*(2), 026126.

Newman, M. E. J. (2003b). The structure and function of complex networks. *SIAM Rev.*, *45*(2), 167-256.

Nguyen, M. N., & Rajapakse, J. C. (2003). Two-stage support vector machines for protein secondary structure prediction. *Neural, Parallel & Scientific Computations*, *11*(1 & 2), 1–18.

Oh, H.-J., Myaeng, S. H., & Lee, M.-H. (2000). A practical hypertext catergorization method using links and incrementally available class information. In *Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval* (pp. 264–271).

Ou, G., & Murphey, Y. L. (2007). Multi-class pattern classification using neural networks. *Pattern Recognition*, *40*(1), 4–18.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 1226–1238.

Perlich, C., & Huang, Z. (2005). Relational learning for customer relationship management. In *In proceedings of international workshop on customer*

relationship management: Data mining meets marketing.

Perlich, C., & Provost, F. (2006). Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, *62*, 65-105.

Polat, K., & Güneş, S. (2009). A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications*, *36*(2), 1587–1592.

Popescul, A., & Ungar, L. H. (2003). Structural logistic regression for link analysis. *Departmental Papers (CIS)*, 133.

Popescul, A., & Ungar, L. H. (2004). Cluster-based concept invention for statistical relational learning. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining* (pp. 665–670).

Preisach, C., & Schmidt-Thieme, L. (2008). Ensembles of relational classifiers. *Knowl. Inf. Syst*, *14*(3), 249–272.

Provost, F., Perlich, C., & Macskassy, S. (2003). Relational learning problems and simple models. In *Proceedings of the ijcai-2003 workshop on learning statistical models from relational data.* Retrieved from `http://pages.stern.nyu.edu/~fprovost/`

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, *77*(2), 275-286.

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research*, *5*, 101–141.

Rossi, R. A., McDowell, L. K., Aha, D. W., & Neville, J. (2012). Transforming graph data for statistical relational learning. *Journal of Artificial Intelligence Research*, *45*(1), 363–441.

Salton, G. (1989). Automatic text processing. 1989. *ed: Addison Wesley*.

Sen, P., & Getoor, L. (2006). Empirical comparison of approximate inference algorithms for networked data. In *Icml workshop on open problems in statistical relational learning (srl2006).*

Sen, P., & Getoor, L. (2007). Link-based classification. In *Um computer science department, technical report, cs-tr-4858.* University of Maryland.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3).

Senliol, B. (2010). *Feature and node selection for collective classification* (Unpublished master's thesis). Istanbul Technical University, Ayazaga, Istanbul.

Senliol, B., Aral, A., & Cataltepe, Z. (2009). Feature selection for collective classification. In *Computer and information sciences, 2009. iscis 2009. 24th international symposium on* (pp. 286–291).

Senliol, B., Gulgezen, G., Yu, L., & Cataltepe, Z. (2008). Fast Correlation Based Filter (FCBF) with a different search strategy. In *International symposium on computer and information sciences.*

Slattery, S., & Craven, M. (1998). Combining statistical and relational methods for learning in hypertext domains. In *Inductive logic programming* (pp. 38–52). Springer.

Snijders, T. A. B. (1992). Estimation on the basis of snowball samples: How to weight? *BMS: Bulletin of Sociological Methodology/Bulletin de M thodologie Sociologique*, *36*(1), 59–70.

Solla, S. A., Leen, T. K., & Müller, K.-R. (2000). *Advances in neural information*

processing systems. The MIT Press.

Tresp, V., Bundschus, M., Rettinger, A., & Huang, Y. (2008). Towards machine learning on the semantic web. In *Ursw (lncs vol.)* (p. 282-314). Springer.

Vapnik, V. N. (1998). Statistical learning theory.

Wang, F., & Zhang, C. (2008). Label propagation through linear neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on*, *20*(1), 55–67.

Wang, W., & Zhou, Z.-H. (2007). Analyzing co-training style algorithms. In *Machine learning: Ecml 2007* (pp. 454–465). Springer.

Waske, B., & Benediktsson, J. A. (2007). Fusion of support vector machines for classification of multisensor data. *Geoscience and Remote Sensing, IEEE Transactions on*, *45*(12), 3858–3866.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*, 440–442.

Xiang, R., Neville, J., & Rogati, M. (2010). Modeling relationship strength in online social networks. In *Proceedings of the 19th international conference on world wide web* (pp. 981–990).

Yang, Y., Slattery, S., & Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, *18(2/3)*, 219–241.

Yaslan, Y., & Cataltepe, Z. (2010). Co-training with relevant random subspaces. *Neurocomputing*, *73*(10), 1652–1661.

Yonghong, T., Tiejun, H., & Wen, G. (2006). Latent linkage semantic kernels for collective classification of link data. *J. Intell. Inf. Syst.*, *26*(3), 269–301.

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the twentieth international conference on machine learning* (p. 856-863). AAAI Press.

Zhou, D., Scholkopf, B., & Hofmann, T. (2005). Semi-supervised learning on directed graphs. In *In nips* (pp. 1633–1640). MIT Press.

Zhu, X. (2008). Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, *2*, 3.

Zhu, X., Ghahramani, Z., Lafferty, J., et al. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Icml* (Vol. 3, pp. 912–919).

APPENDICES

APPENDIX A.1 : Graph Properties Plots

APPENDIX A.1 : Graph Properties Plots

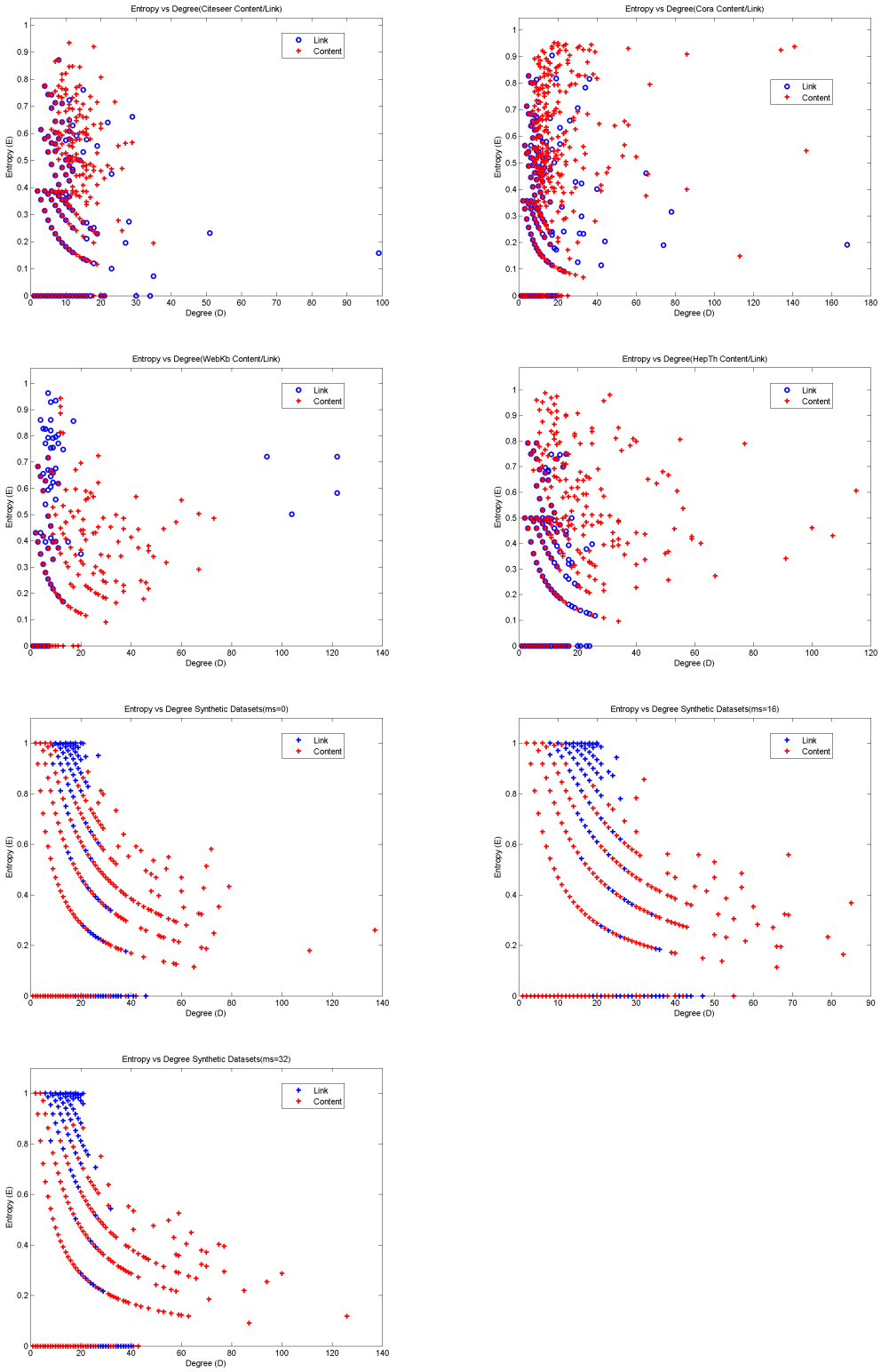


Figure 1 : Entropy vs degree dot plot.

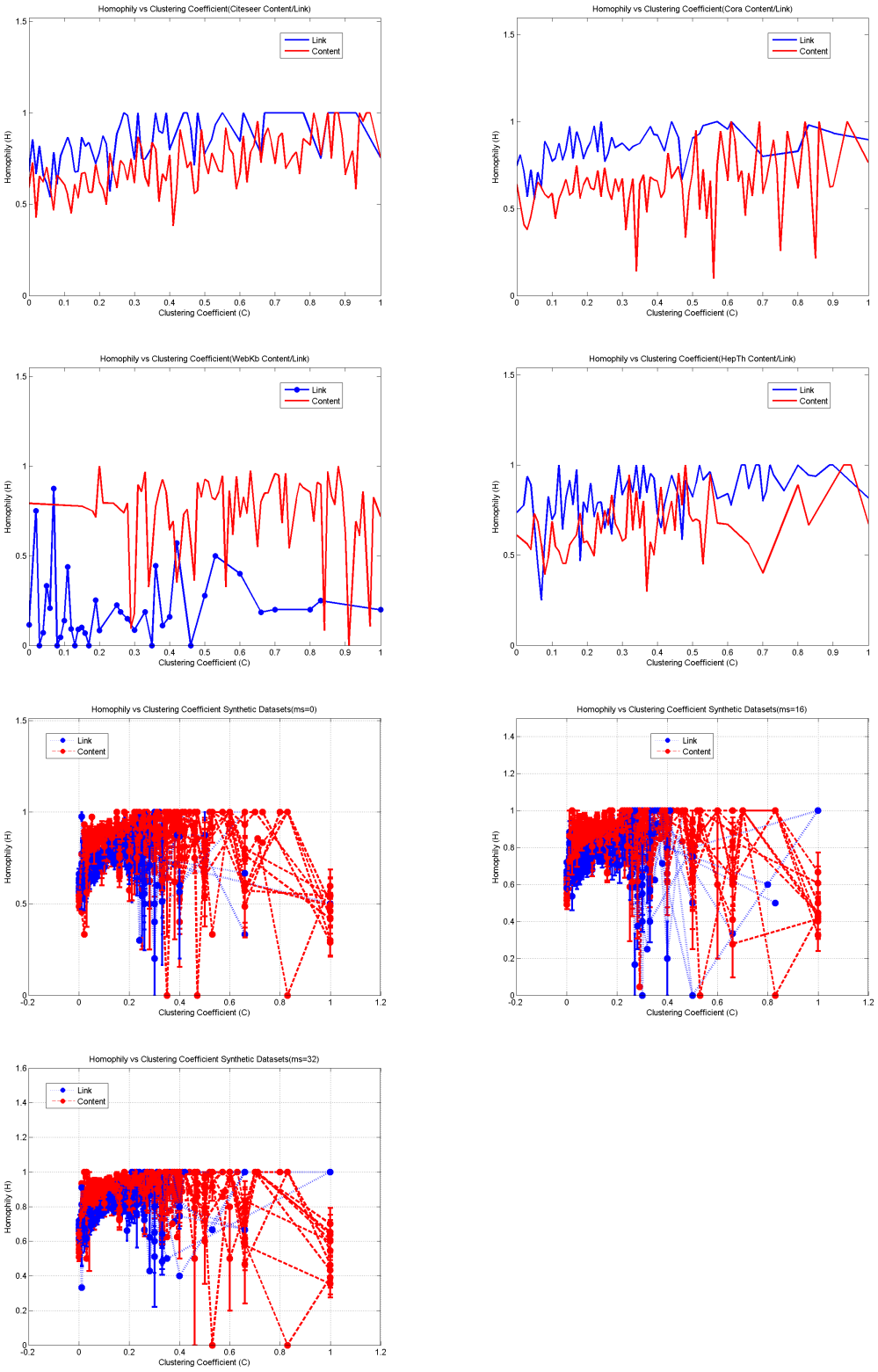


Figure 2 : Homophily vs clustering coefficient.

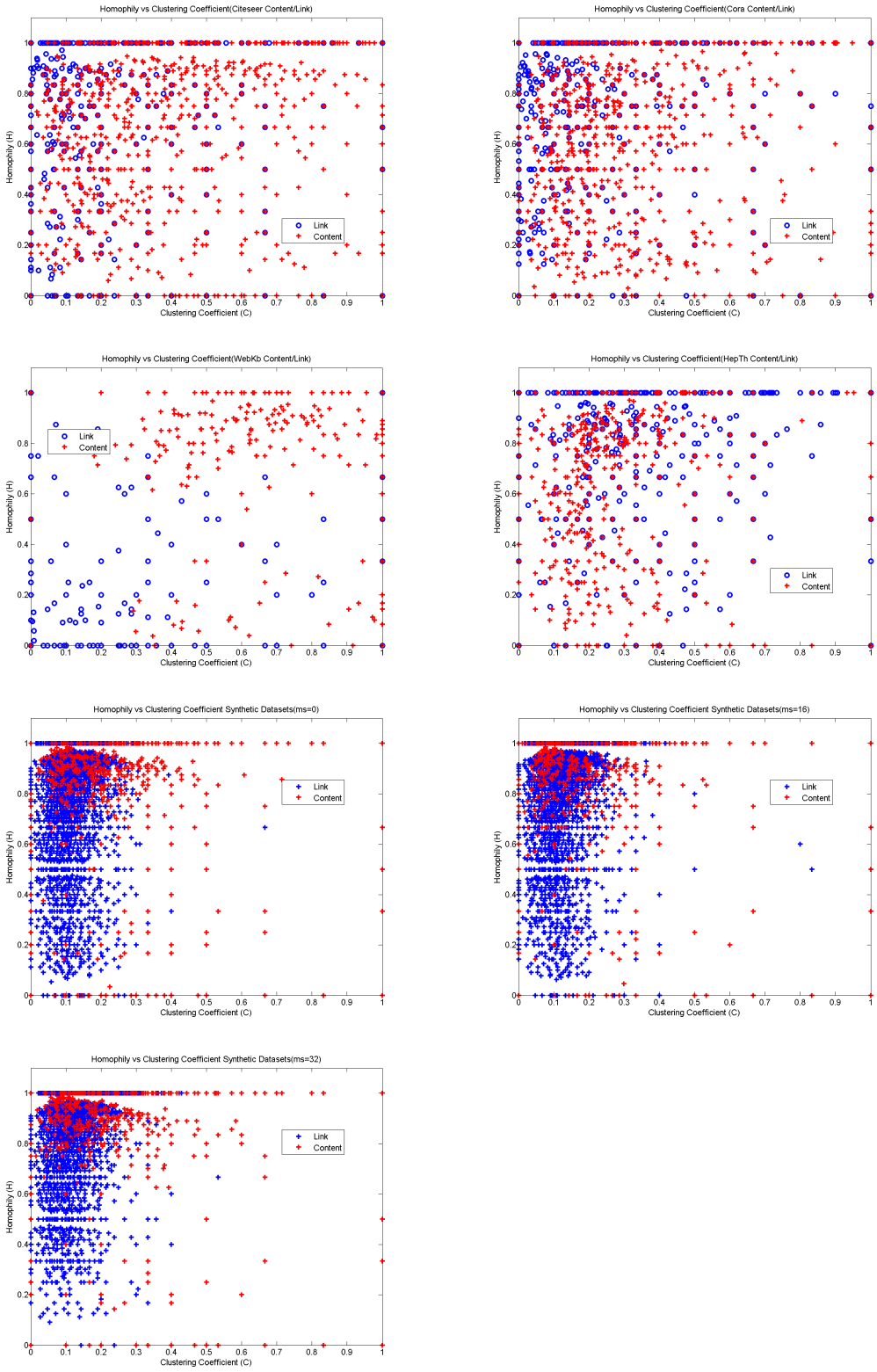


Figure 3 : Homophily vs clustering coefficient dot plot.

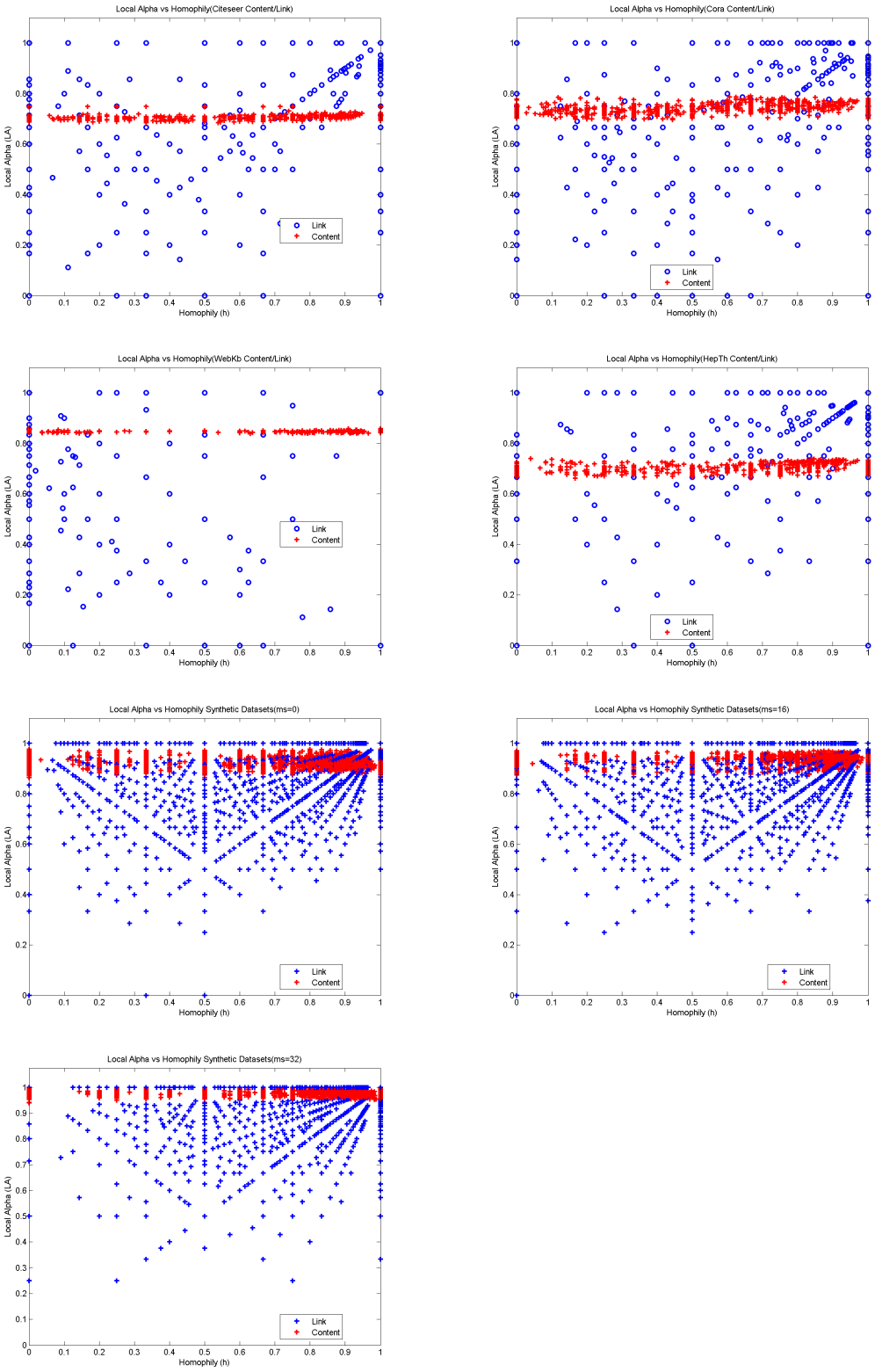


Figure 4 : Local alpha vs homophily dot plot.

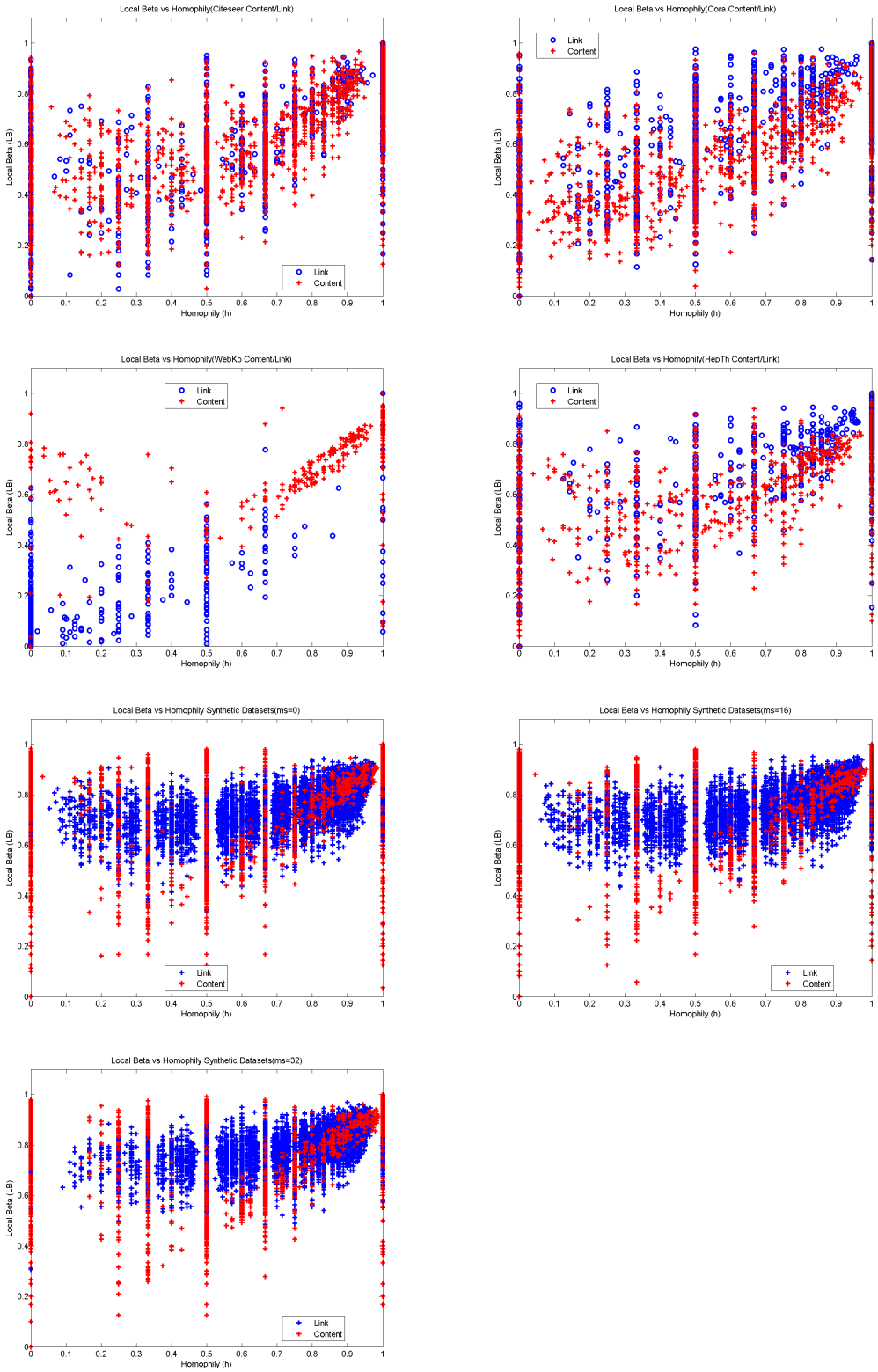


Figure 5 : Local beta vs homophily dot plot.

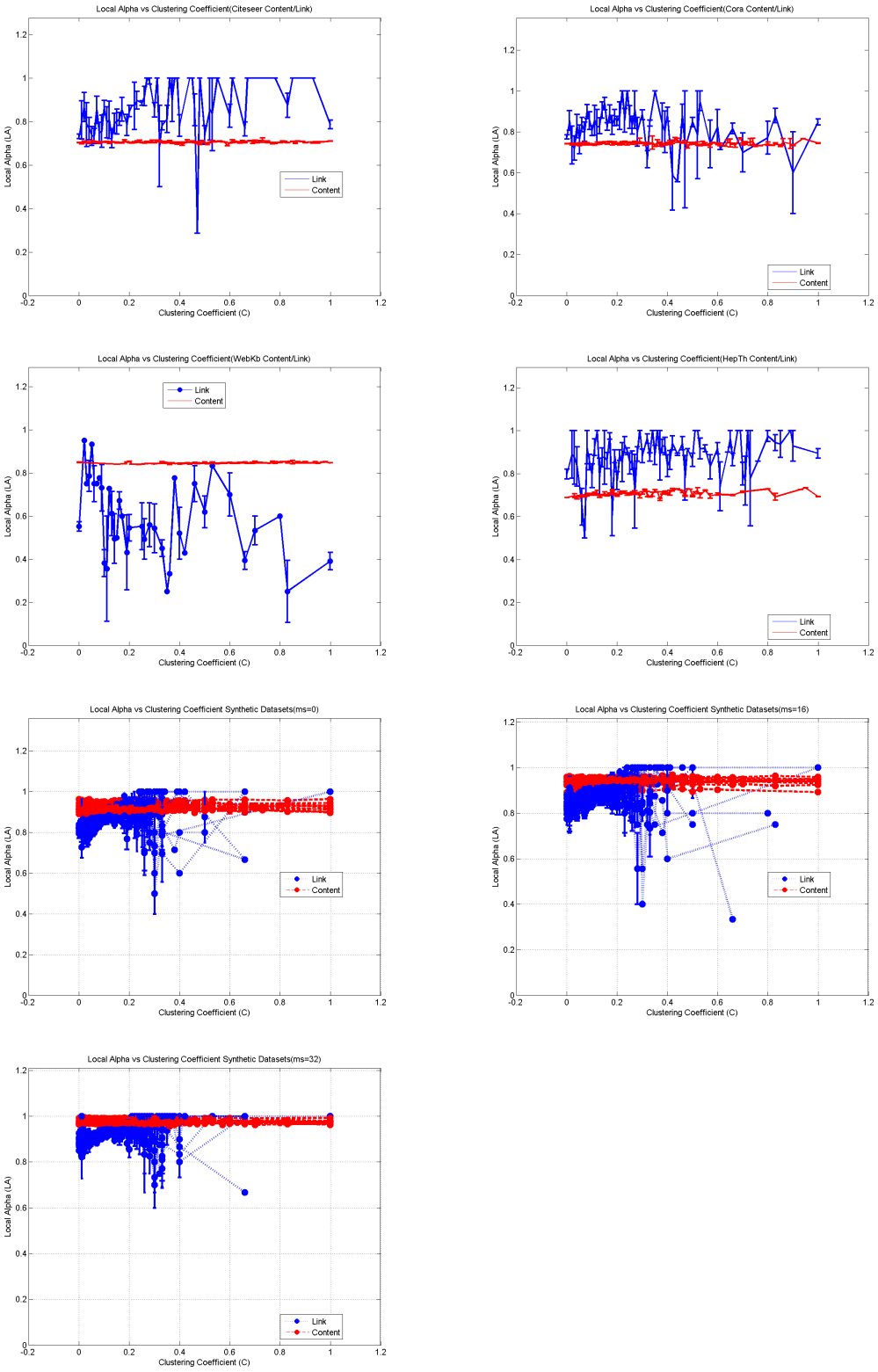


Figure 6 : Local alpha vs clustering coefficient.

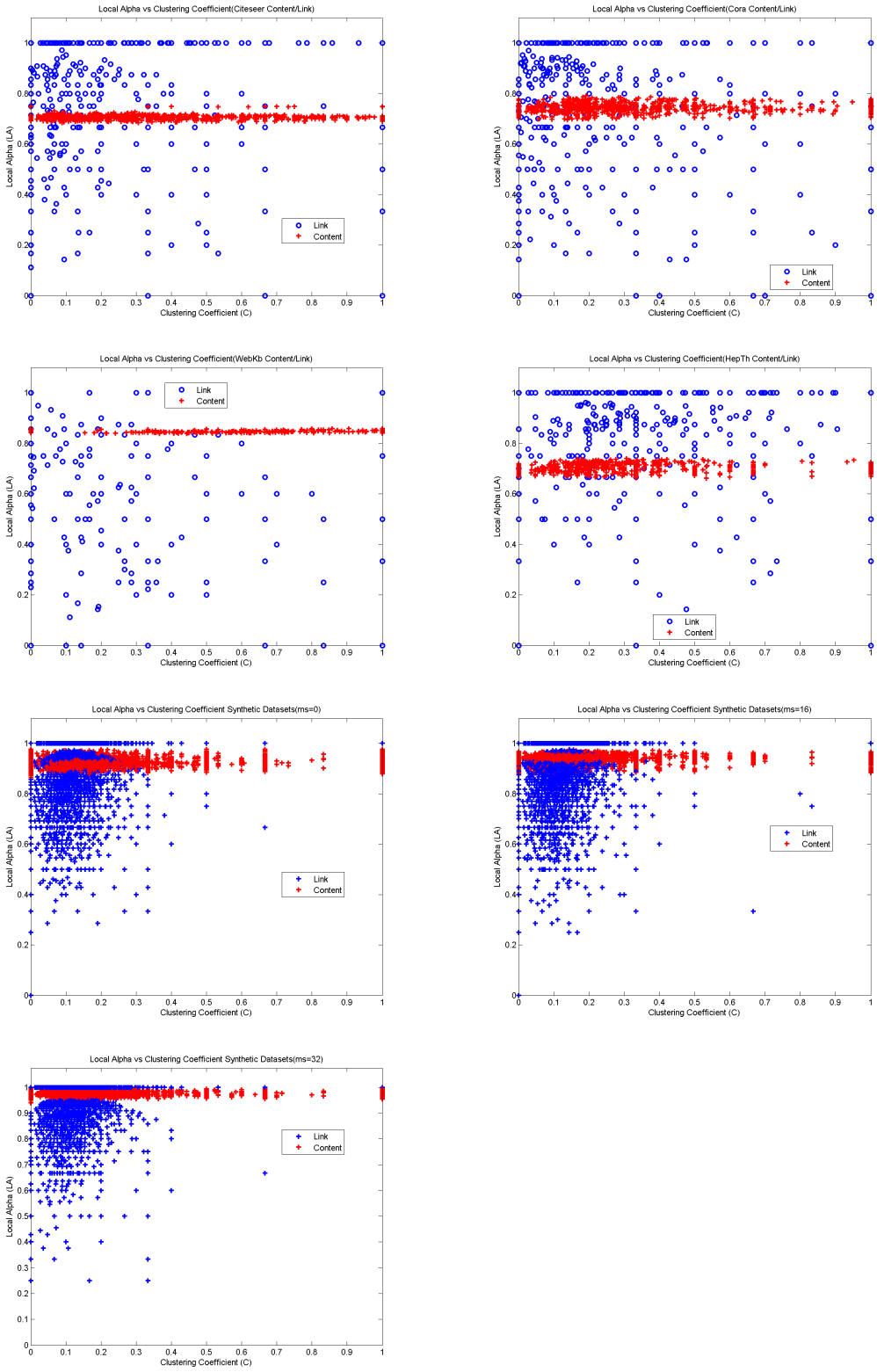


Figure 7 : Local alpha vs clustering coefficient dot plot.

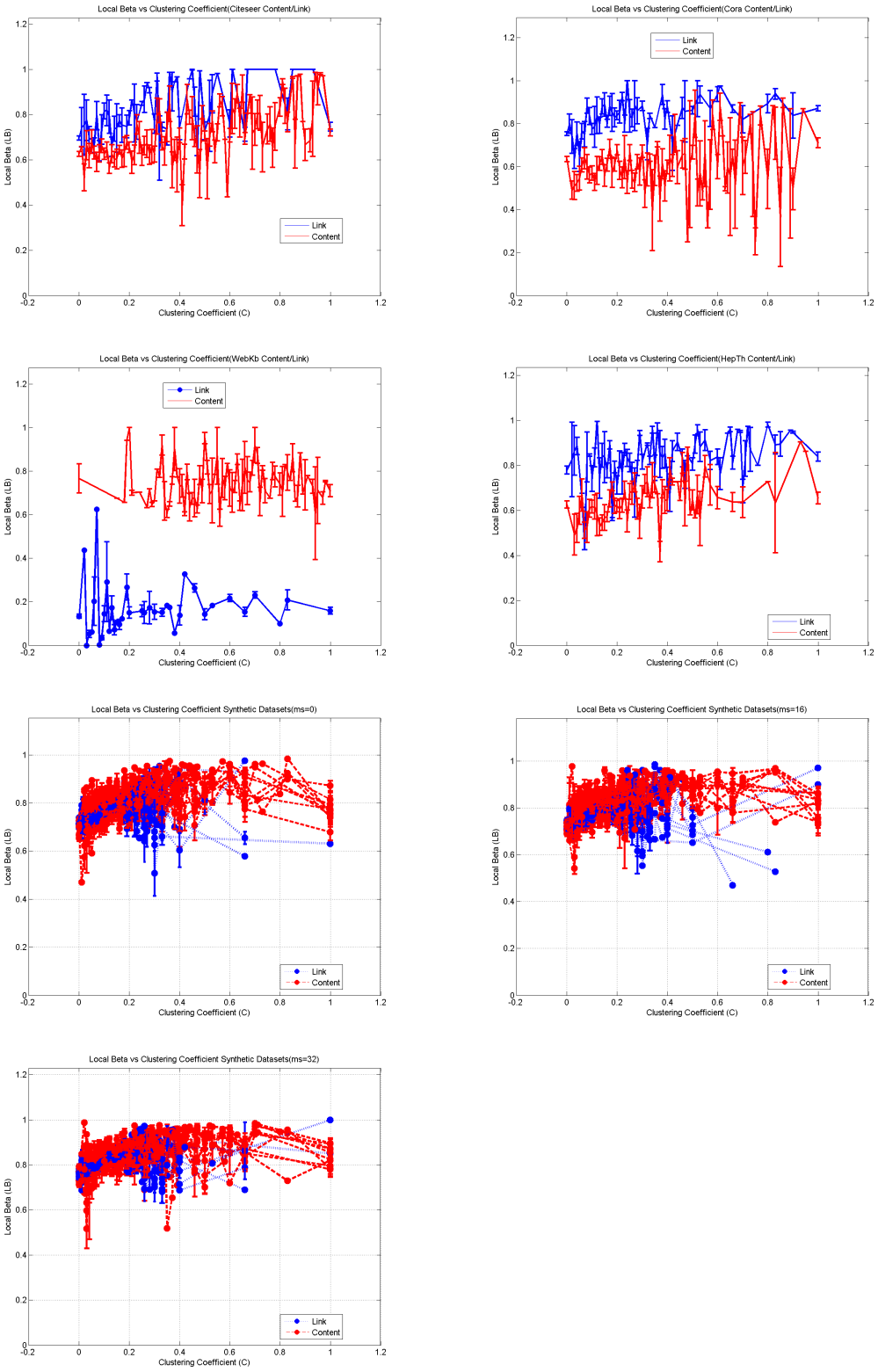


Figure 8 : Local beta vs clustering coefficient.

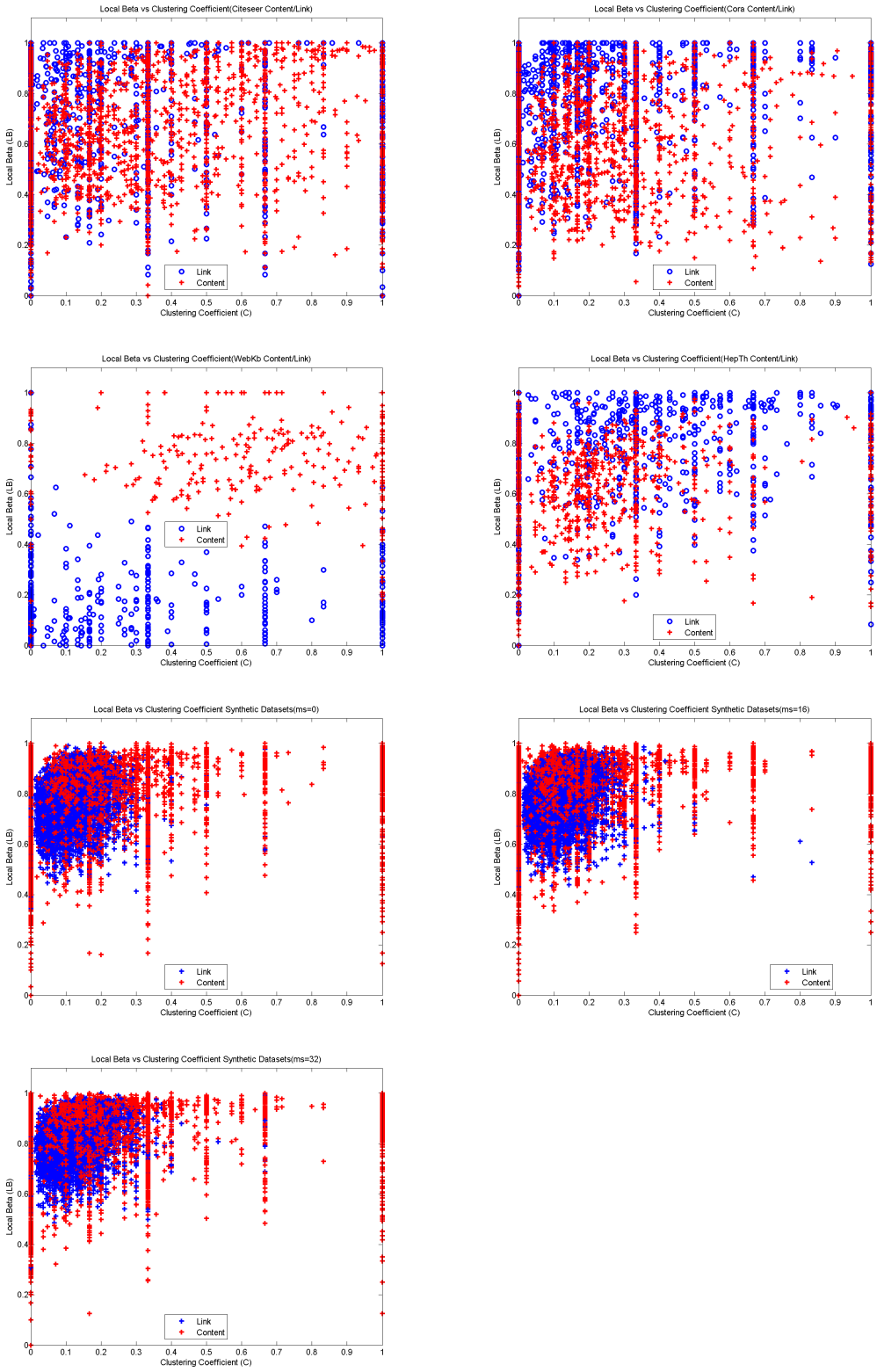


Figure 9 : Local beta vs clustering coefficient dot plot.

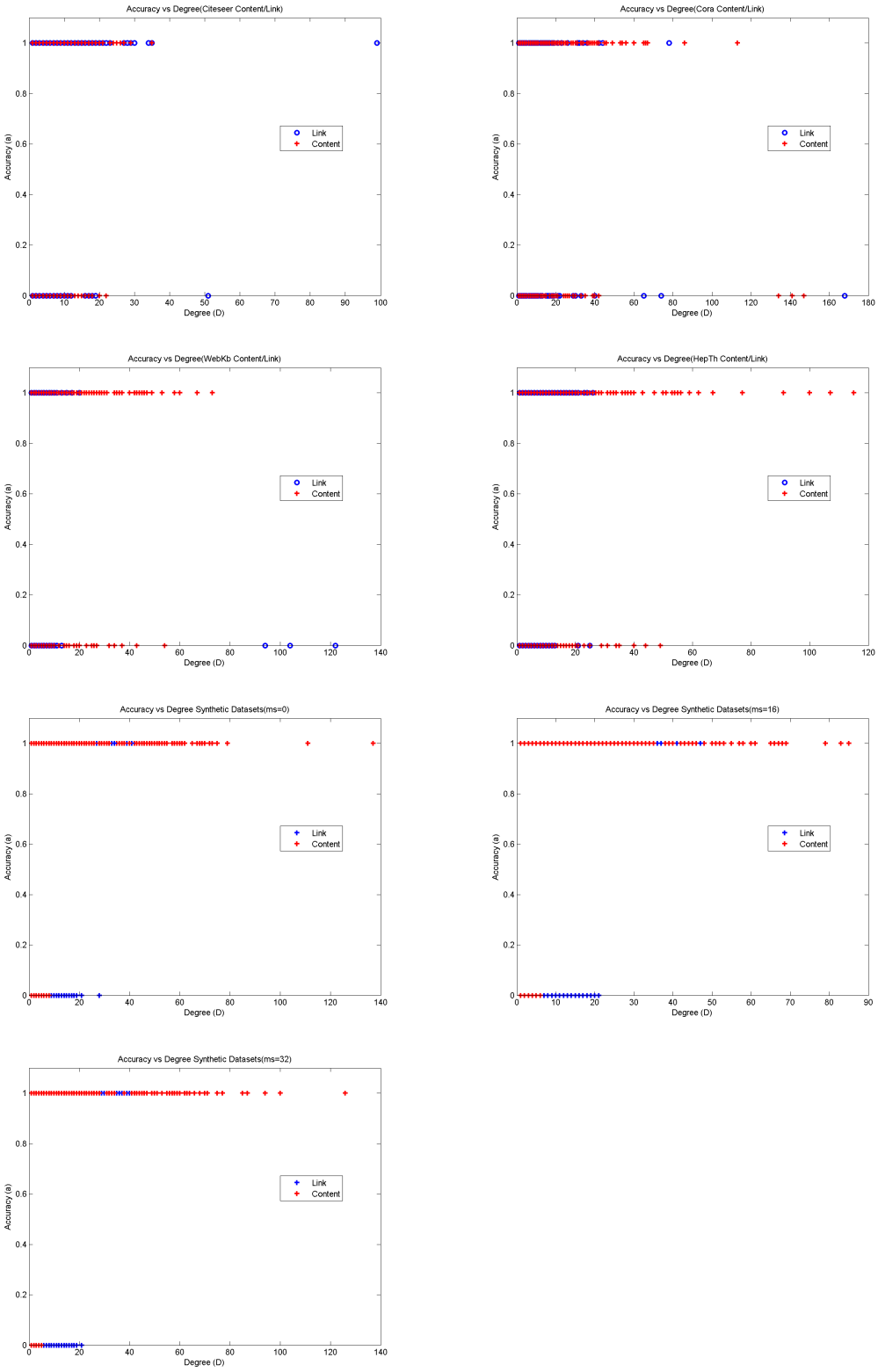


Figure 10 : Accuracy vs degree dot plot.

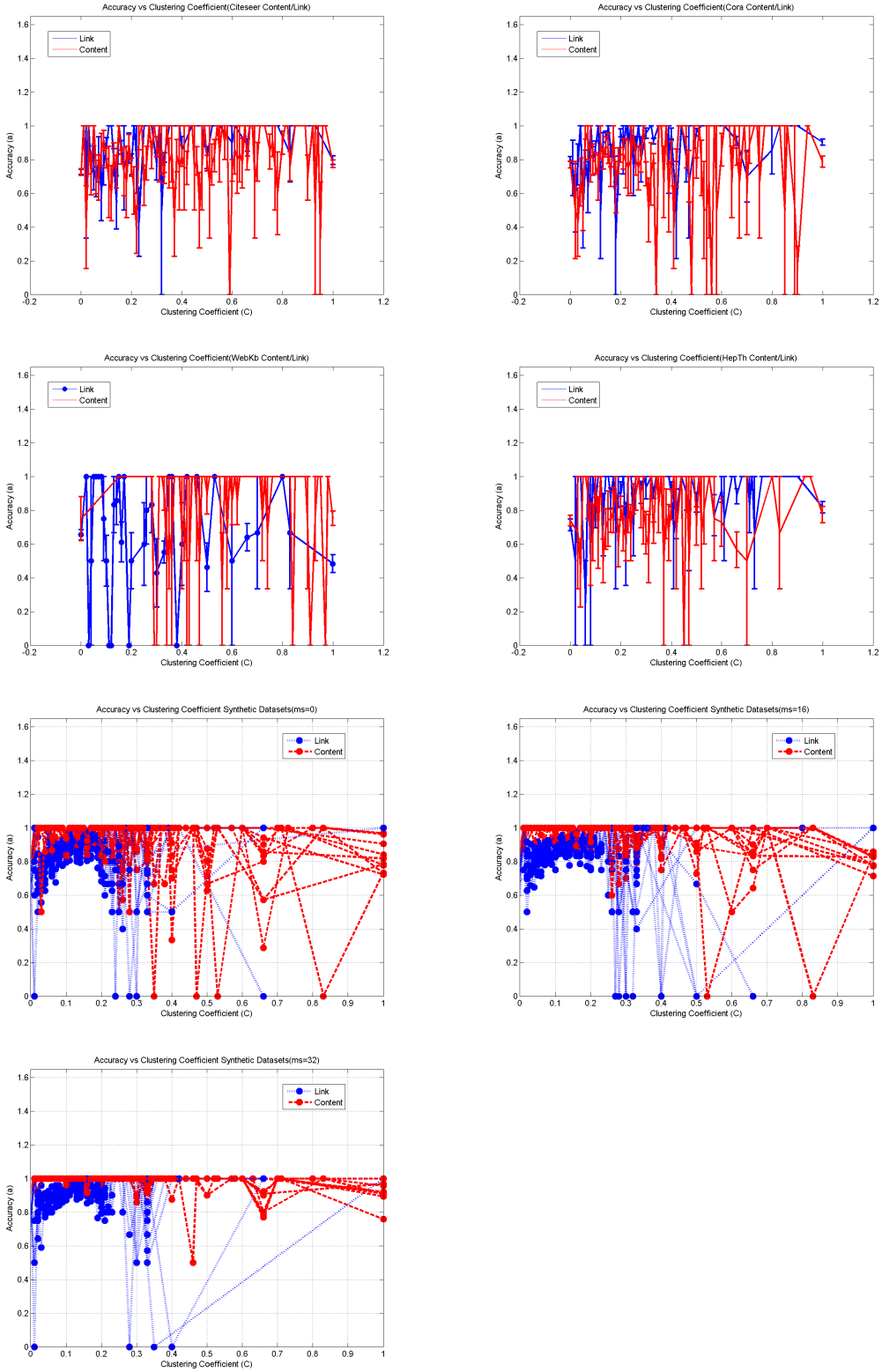


Figure 11 : Accuracy vs clustering coefficient.

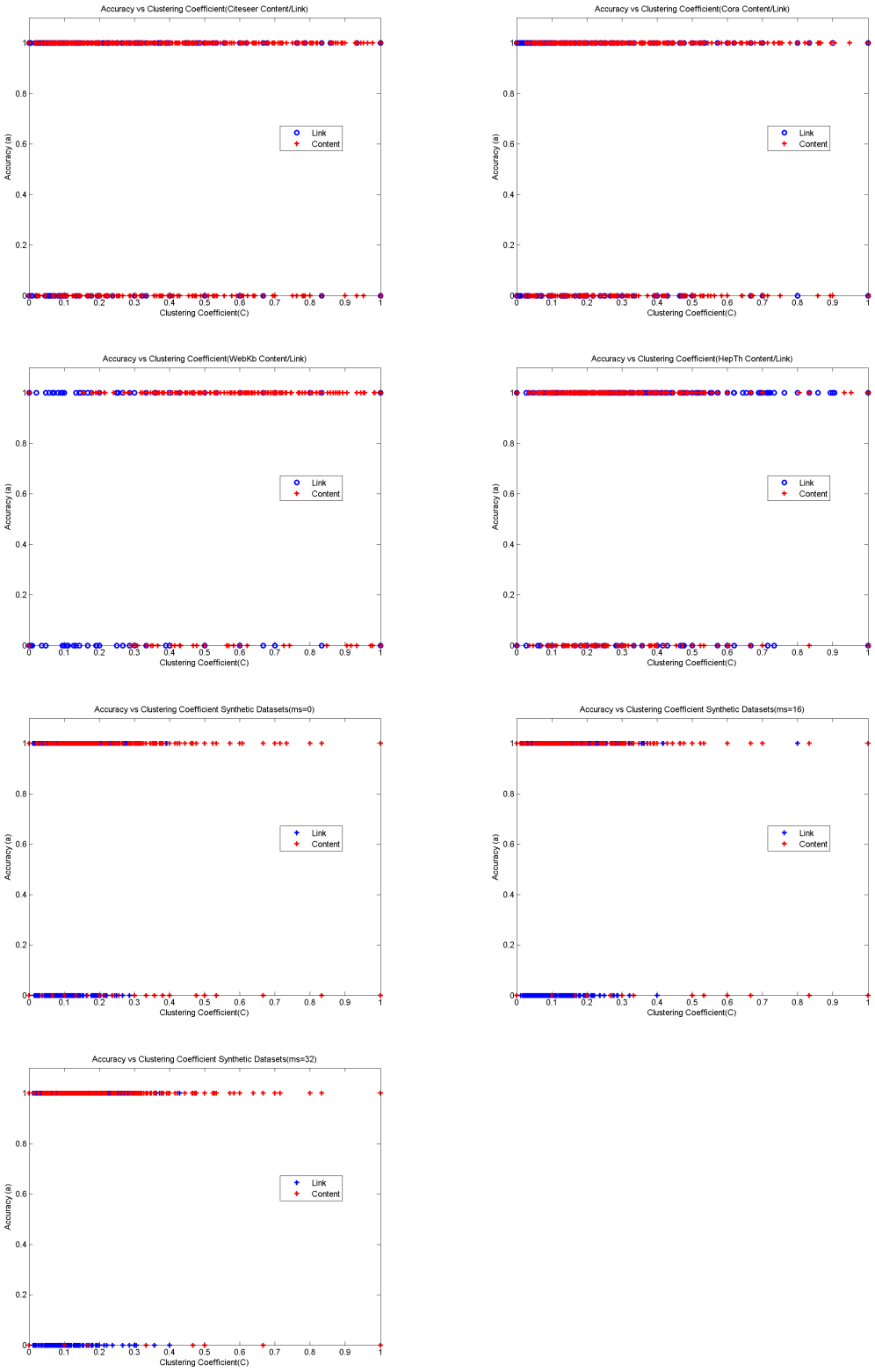


Figure 12 : Accuracy vs clustering coefficient dot plot.

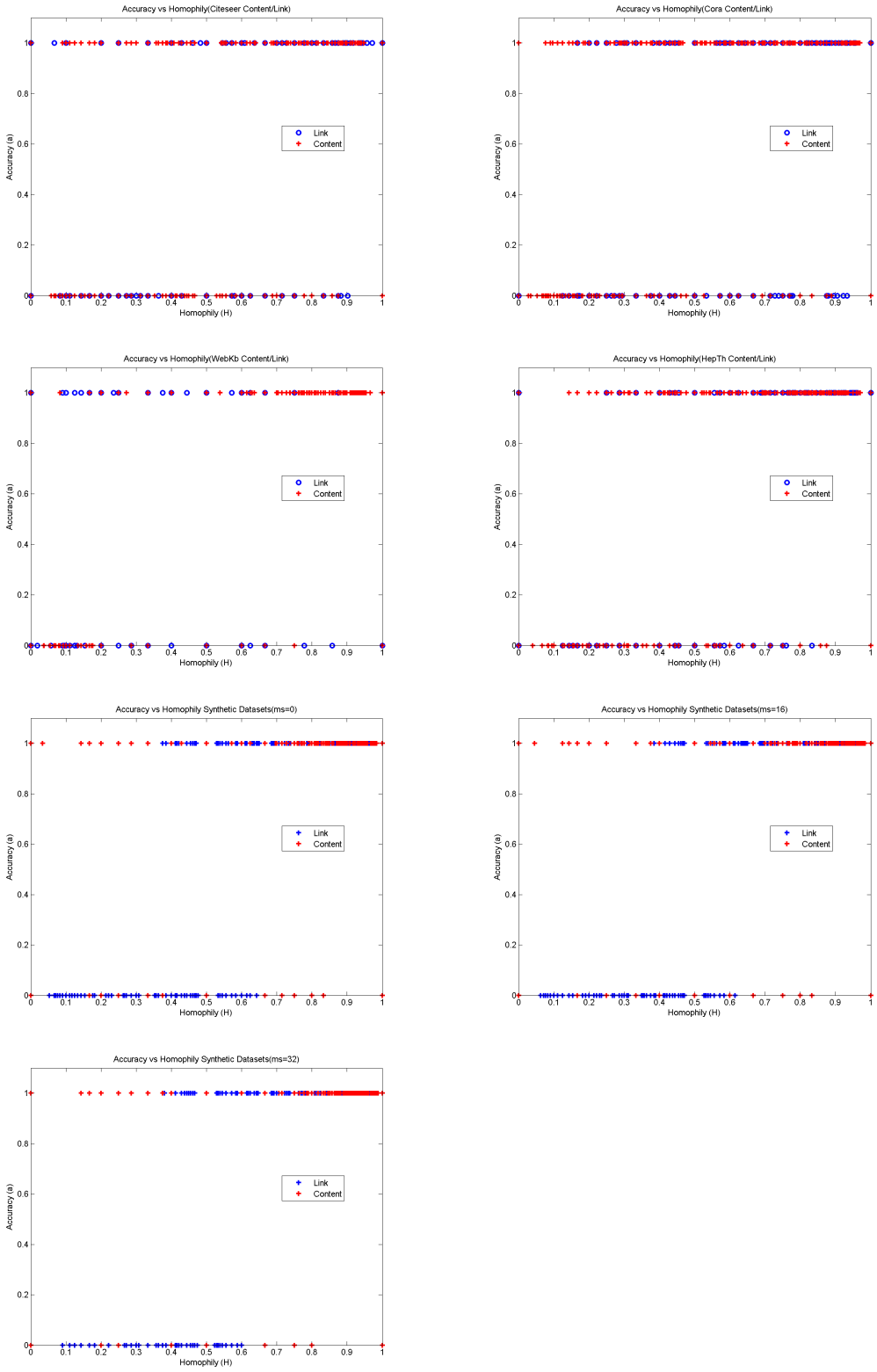


Figure 13 : Accuracy vs homophily dot plot.

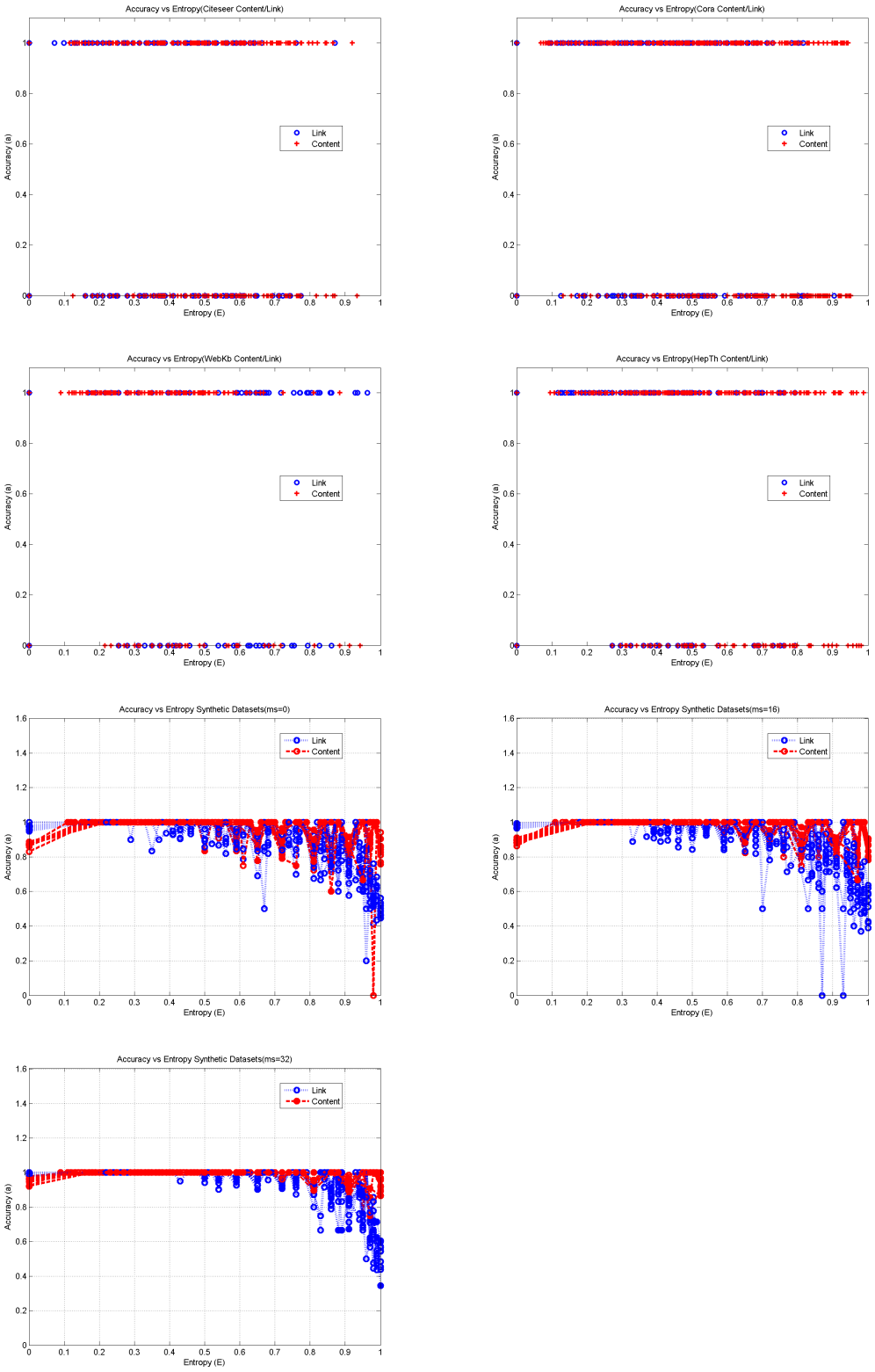


Figure 14 : Accuracy vs entropy dot plot.

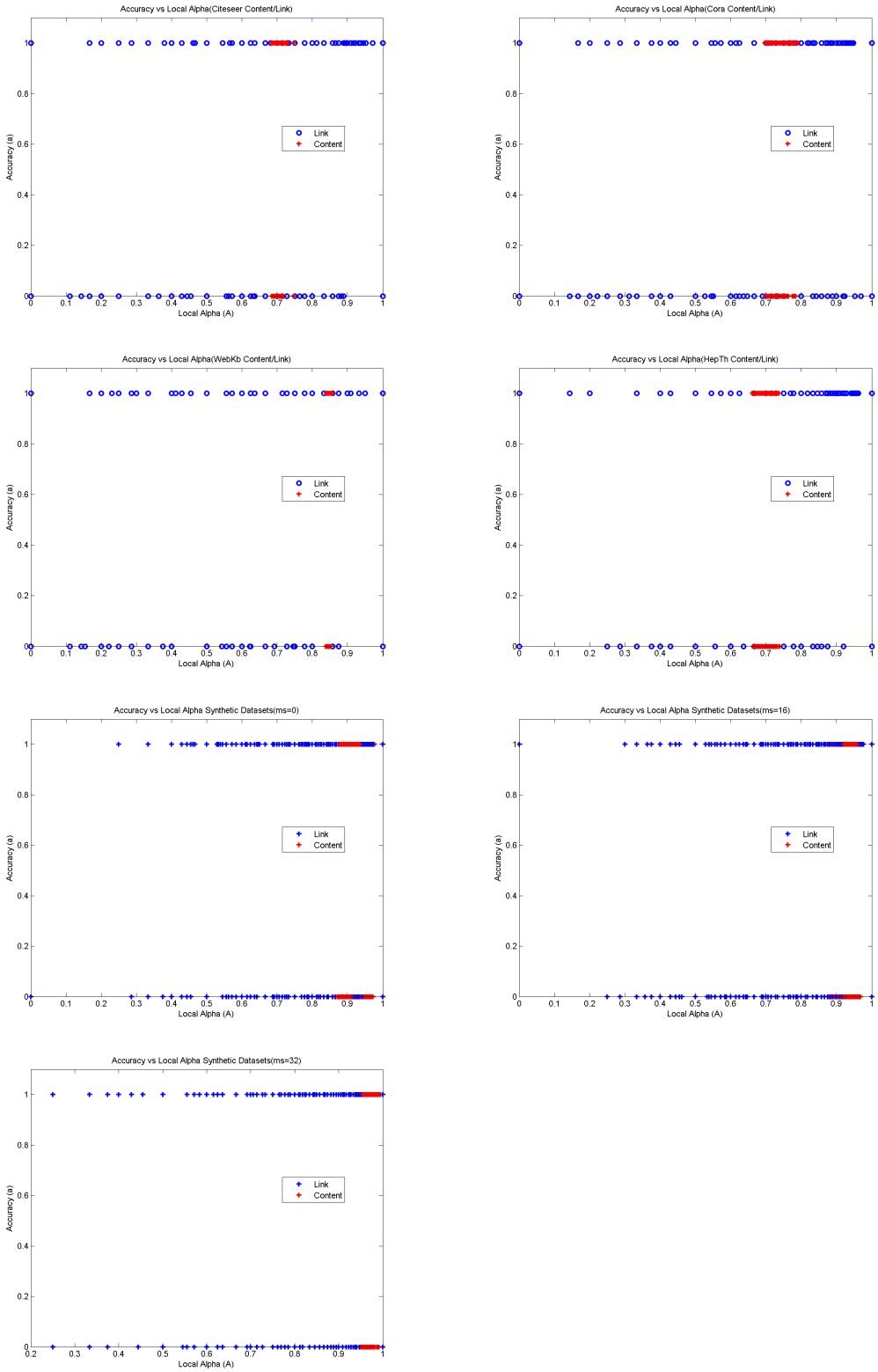


Figure 15 : Accuracy vs local alpha dot plot.

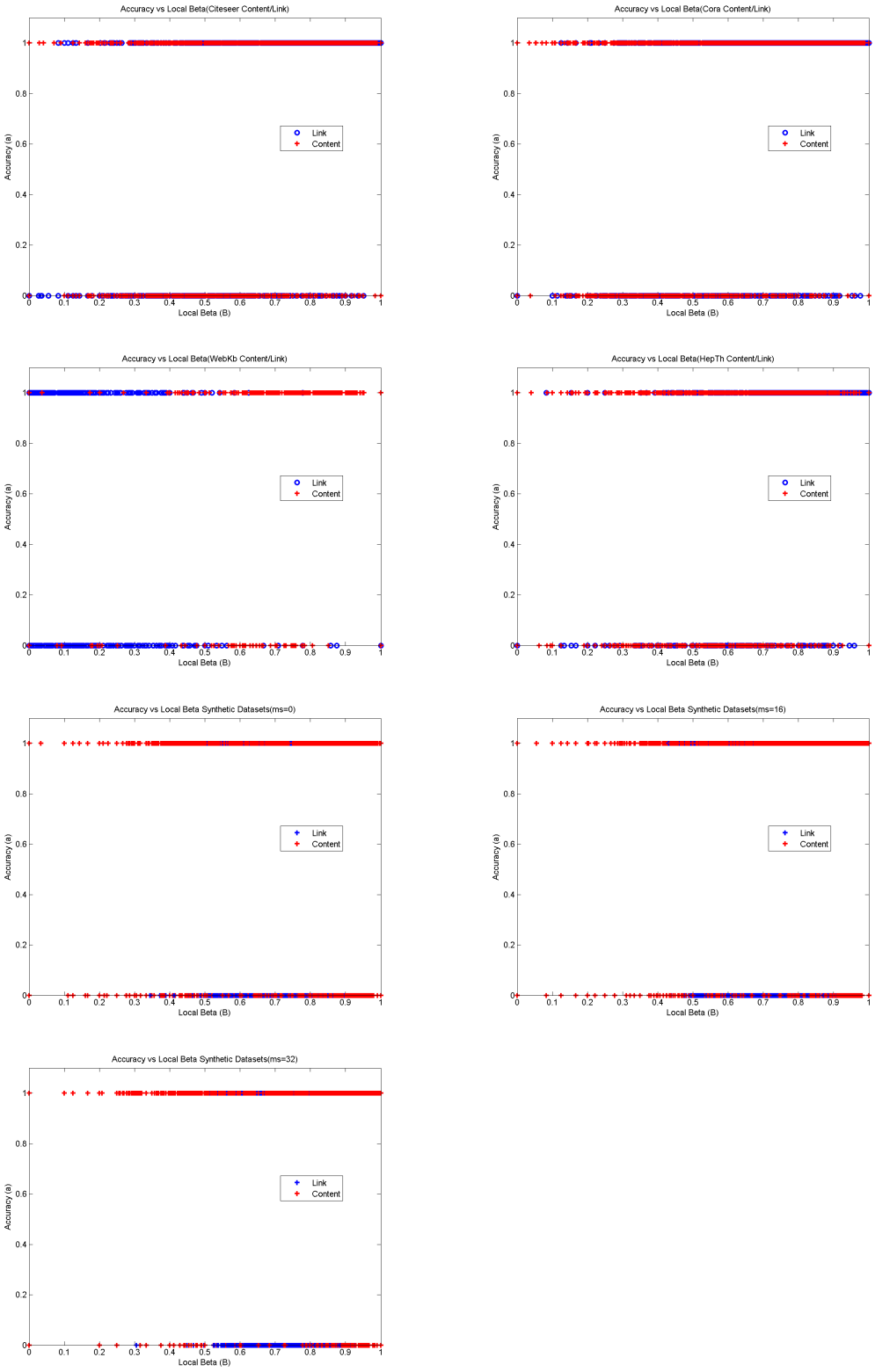


Figure 16 : Accuracy vs local beta dot plot.

CURRICULUM VITAE

Name Surname: Abdullah Sönmez

Place and Date of Birth: İstanbul, 07.03.1980

E-Mail: sonmezab@itu.edu.tr, abdullah.sonmez@gmail.com

B.Sc.: Computer Engineering at Istanbul Technical University, 2002
Thesis Title: "Camera Control Project"
Thesis Advisor: Prof. Dr. Coşkun Sönmez

M.Sc.: Istanbul Technical University, Computer Engineering, 2005
Thesis Title: "Music Genre and Composer Identification by Using Kolmogorov Distance"
Thesis Advisors: Doç. Dr. Zehra Çataltepe, Prof. Dr. Eşref Adalı

List of Publications and Patents:

- Bax E., Li J., **Sonmez A.**, Cataltepe Z.,, 2013: Validating Collective Classification Using Cohorts, *NIPS 2013 Workshop on Frontiers of Network Analysis: Methods, Models, and Applications*, December 9-10, 2013 Nevada, USA.

- Li J., **Sonmez A.**, Cataltepe Z., Bax E., 2012: Validation of Network Classifiers, *Structural, Syntactic, and Statistical Pattern Recognition Lecture Notes in Computer Science*, 7626, 2012, pp 448-457.

- Cataltepe Z., Yaslan Y., **Sonmez A.**, 2007: Music Genre Classification Using MIDI and Audio Features, EURASIP Journal of Advances in Signal Processing, vol. 2007 (January), Article ID 36409, 8 pages.

- Cataltepe Z., **Sonmez A.**, Adali E., 2005: Music Classification Using Kolmogorov Distance, *Representation in Music/Musical Representation Congress*, Istanbul, Turkey, October 2005.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- Cataltepe Z., **Sonmez A.**, 2013: Classification in Social Networks, Book Chapter in: Social Network Analysis:Tutorials and Case Studies, S. Gunduz-Oguducu, S. Etaner-Uyar, eds., submitted.

- Cataltepe Z., **Sonmez A.**, Senliol B., 2013: Feature Enrichment and Selection for Transductive Classification on Networked Data. Pattern Recognition Letters, (DOI: http://dx.doi.org/10.1016/j.patrec.2013.07.009).

▪ Cataltepe Z., **Sonmez A.**, Baglioglu K., Erzan A., 2011: Collective classification using heterogeneous classifiers. In Machine Learning and Data Mining in Pattern Recognition (pp. 155–169). Springer Berlin Heidelberg.

▪ Senliol B., Cataltepe Z., **Sonmez A.**, 2009: Collective Classification with Content and Link Noise *NIPS 2009 Workshop on Analyzing Networks and Learning with Graphs*, December 11, 2009 Whistler, BC, Canada.