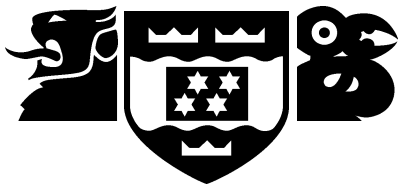
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

Citation-based Text Classification

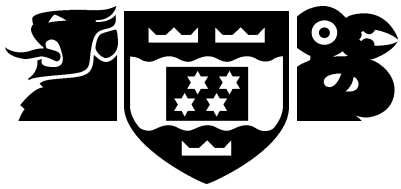
Minh Duc Cao, Xiaoying Gao and Mengjie Zhang

Technical Report CS-TR-05/7
November 2005

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Citation-based Text Classification

Minh Duc Cao, Xiaoying Gao and Mengjie Zhang

Technical Report CS-TR-05/7
November 2005

Abstract

The paper describes a number of approaches to exploiting citation structures for scientific document classification. Three methods, linear labelling update, probabilistic labelling update and neural networks, are developed. The results show that those methods significantly improve classification in comparison with using only document contents. Even though linear labelling update is a static model, it performs well especially when few or no training documents available. Both probabilistic labelling update and neural networks methods require training documents to train a Bayesian network and a neural network, respectively. Experiments show that they outperform linearly labelling update when sufficiently large training set is used.

Author Information

Minh Duc Cao is a postgraduate student in computer science and Xiaoying Gao and Mengjie Zhang are academic staff members in computer science. All the authors are in the School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, New Zealand.

1 Introduction

Document classification (aka *document categorisation* or *topic identification*), the task of assigning text documents to predefined categories, attracts much of research interest in recent years. This is largely due to the increasing availability of tremendous amount of documents in electrical format. Classifying documents from a collection into categories is helpful for people to retrieve useful information more effectively and more efficiently.

Research on document classification dated back to early 1960s, and became more popular in 1990s, when study in *machine learning* became active. Many machine learning methods have been applied to document classification and reported to gain an accuracy comparable to human experts while no human intervention is required [26]. These methods generally automatically train a classifier from a set of pre-classified documents, which are different from *knowledge engineering* methods used in 1980s, where expert knowledge is required to define a set of classification rules.

The set of pre-classified documents used to build the classifier is called the training set. During training process, a set of discriminative characteristics, often from the document text, are generated from the training set and encoded into the classifier to produce classification rules. In research setting, another set of documents called the test set, also pre-classified, is used to evaluate the performance of the classifier. Each test document is fed to the classifier, and the class assigned by the classifier is compared with the true label of the test document. The rate of matches is the measurement of the accuracy of the classifier.

1.1 Issues & Motivations

Scientists spend much of their time on reading papers related to their fields of research. With the increasing volume of scientific literature on-line nowadays, classification of scientific documents is becoming more and more important in research communities. A system of automatic paper classification and recommendation would be very helpful for scientists in organising and collecting their paper database. Conference organisers, librarians and journal publishers would find such a system considerably useful in dealing with an enormous number of papers submitted and published. It is obvious that, manual categorisation of those documents is very time consuming and requires much of human effort. Thus, it is essential to develop a scientific paper classification system that can work automatically and adaptively.

Several scientific paper search engines such as Google Scholar[2] and CiteSeer[1] provide very good tools for researchers for searching for scientific papers. However, searching of papers on those engines are primarily based on keyword matching and therefore, a search would often result in a large number of hits, majority of which may not be relevant to what users look for. A suggested solution to improve the search results is to categorise indexed documents into predefined topics. Users can select the topics they are interested in when performing a paper search. The search engines need to search for documents in selected topics only. That mechanism not only reduces searching time as only documents in those topics are searched but also produces more accurate hit list. Therefore, a scientific paper classification system would be a significant improvement for those scientific document search engines.

Literature has shown many machine learning approaches for general document classification. The most common methods are naive Bayes [14], k-nearest neighbours [28], nearest centroid [10], decision trees [15], support vector machines [11], maximum entropy [19] and neural networks [23]. Generally, those methods extract features from document contents and train a classifier for classification.

Scientific papers, different from general documents, do not exist in isolation but are linked

together by a citation network. A paper normally cites other related published papers which are likely to have similar topics. In other words, the citations link similar papers together. Hence, besides information from document content, the citation structure provides another source of clue that could be exploited for a better classification. If we could combine the document contents information and the citation information, it is expected that a better classification accuracy can be obtained in comparison with using content information only.

This research aims to investigate the effectiveness of citation structure in scientific document classification and develop a system so that citation information can be combined with document contents. The combination system is expected to perform better than that of using only content information for classification.

Three methods are developed for exploitation of citation structure of scientific document domain. The first method, *linear labelling update* uses a linear function to combine citation links of connected documents. The second method, *probabilistic labelling update* applies semantic of Bayesian networks. The last method trains a neural network to model the citation relationship.

The rest of the paper is organised as follows: section 2 presents some essential background and related work. The subsequent section 3 describe our methods for using citation links for document classification. Section 4 shows results from three methods and presents discussions. Section 5 summaries our work and discusses further work.

2 Background

Scientific documents are linked together by a citation network. The connectivity nature of documents hints that, there is extra information rather than document contents that can be used for classification. The integration of linkage information and content information is expected to have a better classification than using contents only.

Link analysis has been researched intensively since the birth of the world wide Web. Brin and Page exploited hypertext mining for PageRank, the technology behind the success of Google [4]. Their method iteratively computes the "importance" of web pages to rank the relevance of web pages which is used to prioritise search results. The anchor texts, i.e. texts of the links are taken into consideration as they well describe the pages the links lead to. Also, the rankings of pages are propagated through links in the PageRank system.

Chakrabarti et al. [5] explored the combining of words from connected documents into the feature vector of a document for classification. Their work shows that the naive use of text form neighbouring documents even degrades accuracy of the classification. Their explanation for the decrease is that link information is noisy and the distribution of terms from neighbouring document is not sufficiently similar to the distribution of the "true" class.

Instead of using contents from neighbouring documents, they incorporated the category information such as some initial guesses of categories. They then applied *Markov random field* model to iteratively update the category of documents. To make calculation manageable, they limited range of influence to 1 or 2 radius, which were equivalent to first and second order Markov random field. The approach is shown to improve the classification accuracy.

Another study by Oh et al. [20] reported similar finding: naive incorporating contents from neighbouring documents is generally not helpful while predicted class information is helpful. Their algorithm for hyperlink mining limits the influence of neighbouring documents to only *trustable* documents, which are documents similar enough to the target document.

Recently, work from Getoor et al. [8] applied the relational model [27] for link mining and Craven and Slattery [7] proposed combining statistical text analysis with a relational learner such as FOIL [22].

3 Methods for Citation Classification

3.1 Overview

The problem of document classification can be stated formally as follows. Given a corpus of documents (representing in feature vectors) $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ and a set of predefined categories or classes $C = \{c_1, c_2, ...c_m\}$, assign each of document d_j to one of the classes c_i.

More generally, a classifier is a function ϕ from document domain $\mathcal{D}$ to R^n where $\phi(d_j) = (s_{j1}, s_{j1}, .., s_{jn})$ and s_{ji} is the category score of document d_j assigning to class c_i. The document d_j will be assigned to the category which has the highest score:

$$\mathcal{C}(d_j) = arg\ max_i\{s_{ji}\} \tag{1}$$

The category score vector $V_j = (s_{j1}, s_{j1}, .., s_{jn})$ output from function ϕ, $V_j = \phi(d_j)$ is called the labelling of document j by function ϕ or by the corresponding classifier. If a document class is known, i.e. document from training set, we also define the labelling of the document as the vector $V_j = (s_{j1}, s_{j1}, .., s_{jn})$ where $s_{ji} = 1.0$ if the document is in class i, otherwise $s_{ji} = 0.0$.

The classifier is trained by a set of training examples S. A good training set should contain examples from all categories, so that the characteristics of all categories could be extracted by the classifier trainer. The training set S could be divided into disjoint sets S_i, which contains the set of training documents for class i.

In the scientific paper domain, citations are clearly a rich source of information to identify topic of documents. There is a correlation of topics between two citing documents. It is observable that the topic of a paper is related to that of papers it links to. This feature suggests that, information in the citation structure could help the classification of scientific papers. This section describes our investigation of combination of citation links and document contents for scientific document classification.

As reported by [5, 20], integration of contents of connected document is not helpful. Instead of using text from neighbouring documents (i.e. documents having links to or from), we explore the update of labelling of a document from the labellings of its neighbouring documents. If the surrounding documents' categories of a document are known, the class of that document can be derived from them. However, that is not always the case. The set of labelled documents in the corpus is limited as labelling of document is expensive. Instead of using "true" labelling, we first apply one of the content-based classification method to classify all documents in the corpus as a starting point. The labellings are then updated to give a better classification result.

Similar to work from [5] we use a boottraping approach. A content-based classifier is trained to predict the classes of documents. A link-based classifier is then trained and used to classify based on the predicted category information. Formally, the labelling of each document d_j by a classifier is a vector $V_j = (s_{j1}, s_{j2}, ..., s_{jm})$ in which s_{jk} is the likelihood of that paper in class c_k. We need to design an updating function $\mathcal{F}$ to update the labelling of a document from its neighbouring predicted labellings. Let N_j be the set of documents that connect with document d_j, the general form of the updating function $\mathcal{F}$ is:

$$V_j = \mathcal{F}(V_j, V_{j1}, V_{j2}, ..., V_{j|N|}) \tag{2}$$

or

$$V_j = \mathcal{F}(V_j, V_k) \tag{3}$$

for updating from an individual neighbour document d_k.

We developed three methods for updating labellings in this work. The first one *linear labelling update*, described in Section 3.2, defines a linear function to combine labellings of

neighbouring documents. Sections 3.3 presents the second method, *probabilistic labelling update*, which is based on Bayesian networks to define the updating function. Section 3.4.1 presents the third method which uses a neural network to learn the updating function.

3.2 Linear Labelling Update

The rationale behind the linear labelling update (LLU) is that, the information to identify topic of paper comes from the document itself and its neighbouring documents. Therefore, the labelling of a document is the combination of its own labelling (output from a content-based classifier) and labellings of connected documents. The method assumes that, each neighbouring document has an equal influence on the document. The updating function therefore adds a fraction of those labellings into the document's labelling.

$$V_j = (V_j + \eta \sum_{k:d_k \rightarrow d_j} V_{jk}) \tag{4}$$

where the updating rate η is a parameter that reflects the influence of linked papers. The higher η, the more influence of the categories of neighbouring documents on the target document.

The updating procedure is performed iteratively until the labellings of all documents in the corpus become stable, i.e. the change of labellings is small enough.

3.3 Probabilistic Labelling Update

There are several limitations of the LLU approach above. Firstly, the model is static; the inference is dependent on the parameter η. The system may work well in some parameter values but may fail in others. Deriving a good parameter value requires an empirical search and the parameter value found is unlikely to be optimal. It would be desirable if parameters can be learnt from the training set. Secondly, the model is rather "naive" and may not capture the dependency well.

The dependence among documents in the corpus is presented by citation links. The citation structure can be modelled as a directed graph in which vertexes are documents and edges are citation links. As a paper can only cite other papers already published, there should not be a circle in the graph. The graph is similar to Bayesian network or belief network [21]. Therefore, it is suggested that Bayesian network can be used to model citation link structure. We call the method derived the Bayesian network probabilistic labelling update (PLU).

3.3.1 Modelling Citation Links by Bayesian Network

Bayesian network is a directed graph representing dependencies among random variables [9]. Each node in the graph represents a random variable. An edge from node X to node Y represents dependency in the form of conditional probability $P(X|Y)$. In this case, variable X is said to be a parent of variable Y. Each node is associated with a conditional probability distribution $P(X|parents(X)$ that represents the effects of the parents on the node.

A Bayesian network allows calculation of the probability of some node given that some others nodes are observed. The well-known algorithm for exact inference in Bayesian network is *probability propagation* or sometime called *sum-product algorithm* or *message passing* [21, 12, 29, 6]. The idea behind the algorithm is that, belief is passed across the network to update probabilities of unobserved nodes.

If the graph is a poly-tree i.e. between any two nodes, there is at most one indirected path, the complexity of exact inference in Bayesian network is linear to the size of the network. However, in most cases, the network is multiply-connected, inference is NP-hard [25]. If

that is the case, several tractable approximate inference methods could be used. Messages are still "sent anyway" between any two nodes which introduce some imprecision, or Monte Carlo method [17] is used to generate samples from the distributions in the network.

From the above observation, this approach models the link citation by a Bayesian network. An example of the network is shown in Figure 1. There are two types of nodes in the network. Each round node represents a document, whose category is directly influenced by its text and other neighbouring documents. A square node represents the text of a document. Parent nodes of a document are its text node and other documents that are cited by the document.

For the example, in Figure 1, document d_3 cites documents d_1 and d_2. Therefore, its parents are its text node T_3 and two document nodes d_1 and d_2. Document d_3 in turn, is the parent of documents d_4 and d_5 as it is cited by those two.

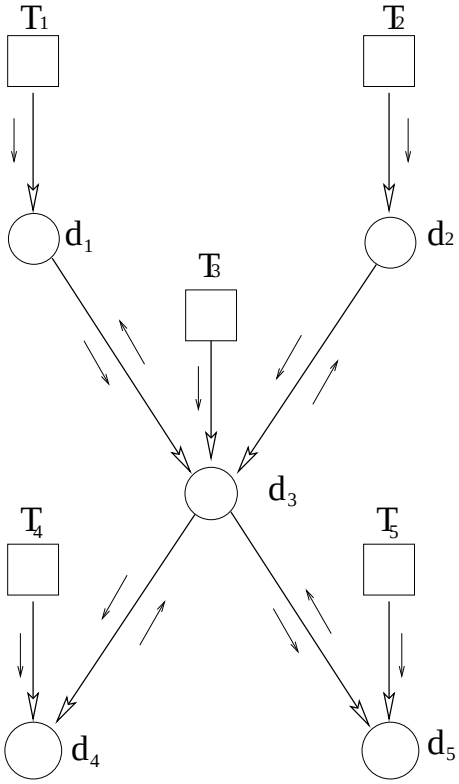


Figure 1: A sample citation Bayesian network.

In the citation Bayesian network described above, the text nodes are observed. As a text node has no parent node and only a child node, it sends probability message to is child node, which is the document having it as content. Each document, upon receiving belief probability from its text node, sends messages to its parents and its children document nodes. A detail of inference in Bayesian citation network is described in next sub section.

3.3.2 Inference in Citation Network

Assume we need to find the probability of a document d_j belonging to class c_i, $P(d_j = c_i)$ (a short notation for $P(\mathcal{C}(d_j) = c_i)$) based on available information from its text T_j and citation information N_j. We apply Bayesian rule for that probability as:

$$P(d_j = c_i|T_j, N_j) = \frac{P(T_j, N_j|d_j = c_i)P(d_j = c_i)}{P(T_j, N_j)} \tag{5}$$

Again we make another independence assumption, which states that, in a document, the content T and the neighbouring N are statistically independent. The independence assumption is interpreted by the equation:

$$P(T_j, N_j|d_j = c_i) = P(T_j|d_j = c_i)P(N_j|d_j = c_i) \tag{6}$$

We also can cancel out the normalisation factor$\frac{1}{P(T_j, N_j)}$, equation 5 can be written as:

$$\begin{aligned} P(d_j = c_i|T_j, N_j) &\propto P(T_j|d_j = c_i)P(N_j|d_j = c_i)P(d_j = c_i) \\ &= P(T_j|d_j = c_i)P(d_j = c_i)P(N_j|d_j = c_i) \\ &= P(d_j = c_i|T_j)P(N_j|d_j = c_i) \end{aligned} \tag{7}$$

$P(d_j = c_i|T_j)$ is the probability of document d_j in class c_i given its content. The probability is exactly the category score computed by a content-based classifier. Therefore the improvement would be obtained by multiplying with $P(N_j|d_j = c_i)$.

If we further assume the independence among information from each neighbouring documents n_k given d_j, we can write $P(N_j|d_j = c_i)$ as

$$\begin{aligned} P(N_j|d_j = c_i) &= \prod_{k:d_k \to d_j} P(n_k|d_j = c_i) \\ &= \prod_{k:d_k \to d_j} \frac{P(d_j = c_i|n_k)P(n_k)}{P(d_j = c_i)} \\ &= \prod_{k:d_k \to d_j} \frac{P(d_j = c_i, n_k)}{P(d_j = c_i)} \end{aligned} \tag{8}$$

where d_k is in the set of neighbouring documents of d_j and n_k is the information d_j obtains from d_k.

The node d_k is actually not observed. In fact, it can be computed by information from its text node, which is equivalent to $P(d_k = c_i|T_k)$, and its neighbouring $P(d_k = c_i|N_k)$. In other words, the categories information of d_k is obtained from the content-based classification and labelling updating from its neighbour. d_k itself does not generate information but encapsulate all messages it receives and passes on to d_j. Suppose messages received by d_k are from some sources of evident e_k (which are T_k and N_k as shown on Figure 2), message n_k from d_k to d_j is actually is e_k. Therefore we have

$$P(d_k|e_k) = P(d_k|T_k, N_k) \tag{9}$$

and

$$P(d_j = c_i, n_k) = P(d_j = c_i, e_k) \tag{10}$$

As can be seen from Figure 2, information from e_k is sent to d_j via d_k. Therefore, the joint probability $P(d_j = c_i, e_k)$ can be factored as

$$\begin{aligned} P(d_j = c_i, e_k) &= \sum_l P(d_j = c_i, d_k = c_l, e_k) \\ &= \sum_l P(d_j = c_i|d_k = c_l)P(d_k = c_l|e_k)P(e_k) \\ &= \sum_l P(d_j = c_i|d_k = c_l)P(d_k = c_l|T_k, N_k)P(e_k) \end{aligned} \tag{11}$$

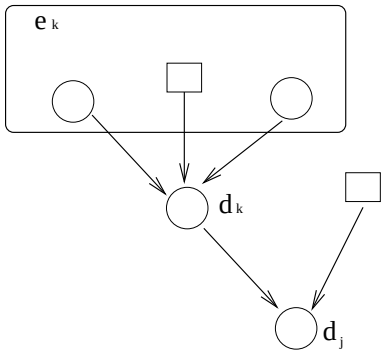


Figure 2: Information from d_k to d_j

where l are all possible values of class c_l.

$P(e_k)$ is again the common factor for all categories c_i and thus can be normalised. Plug equation 11 into equation 8 and equation 7, we obtain:

$$\begin{aligned}
&P(d_j = c_i|T_j, N_j) \propto \\
&\quad P(d_j = c_i|T_j) \prod_{k:d_k \to d_j} \frac{\sum_l P(d_j = c_i|d_k = c_l)P(d_k = c_l|e_k)P(e_k)}{P(d_j = c_i)} \\
&= P(d_j = c_i|T_j) \prod_{k:d_k \to d_j} \frac{\sum_l P(d_j = c_i|d_k = c_l)P(d_k = c_l|T_k, N_k)P(e_k)}{P(d_j = c_i)} \\
&\propto P(d_j = c_i|T_j) \prod_{k:d_k \to d_j} \frac{\sum_l P(d_j = c_i|d_k = c_l)P(d_k = c_l|T_k, N_k)}{P(d_j = c_i)}
\end{aligned} \tag{12}$$

In the above formula, $P(d_j = c_i|T_j)$ is calculated from the embedded content-based classification. $P(d_k = c_l|T_k, N_k)$ is the previous calculation on d_k. The term $P(d_j = c_i|d_k = c_l)$ and $P(d_j = c_i)$ can be learned from training set.

3.3.3 Learning the Model

The independence assumption we made above makes learning the network easier. Instead of learning the full conditional probability distribution $P(d_j|\text{parent}(d_j))$, the model needs only to learn the distribution $P(d_j = c_i|d_k = c_l)$ that can applied to all citations dependences.

As a citation link can be either in-link (cited) or out-link (citing), we need to estimate two conditional probability distributions, one for the case d_j cites d_k and the other for the case d_j is cited by d_k. Each distribution is a matrix $|C| \times |C|$. The entry (i, l) of the in-link matrix is estimated from training set by the maximum likelihood estimation:

$$\begin{aligned}
\text{in-link}_{il} &= P(d_j = c_i|d_k = c_l) \\
&= \frac{L_{li}}{\sum_k^{|C|} L_{lk}}
\end{aligned} \tag{13}$$

where L_{lk} is the number of citation links from a document in class c_l to c_k.

Similarly, the matrix out-link is also estimated as:

$$\begin{aligned}
\text{out-link}_{il} &= P(d_j = c_i|d_k = c_l) \\
&= \frac{L_{il}}{\sum_k^{|C|} L_{kl}}
\end{aligned} \tag{14}$$

Finally, the prior probability is estimated as:

$$P(d_j = c_i) = \frac{|S_i|}{|S|} \tag{15}$$

3.4 Learning Citation by Multilayer Neural Networks

Artificial neural network is a machine learning model based on the function of biological neurons. A neural network consists of a number of artificial neurons(nodes) which are highly interconnected to each others. Each connection is associated with a weight. The neural network is trained from a number of training patterns by adjusting the weights such that the error rate is minimised.

Neural networks have been applied to many classification and detection problems since 1980s' [24]. One of the most common architecture of neural network is feed-forward networks, which contain an input layer, one or more hidden layers and an output layer. Generally, the technique to train a feed-forward neural network is back-propagation. A randomly selected pattern is presented to the input layer of the neural network. The activation is propagated to the hidden layer(s) and then the output layer. The result is compared with the desired output of the training pattern. The discrepancy of the training pattern is then propagated back to hidden layer and then input layer. The weights are then adjusted so that the presentation of the same pattern to the neural network will generate less error.

In this approach, we used a neural network to learn the citation links of scientific documents. Specifically, the a neural network is trained to capture the citation link distribution. The neural network will compute the labelling of a document based on its labelling computed by the content-based classification and information from its neighbouring.

3.4.1 Neural Network Construction and Training

The neural network is designed to model the labelling update function of a document. The network takes the content-based labelling of an unknown document and of its neighbouring documents as input and outputs the labelling that utilises both content and citation link information.

To capture the link information, we need to incorporate some link features to the input vector of the neural network. Link features are source of information for classification obtained from neighbouring documents. As reported by [5, 20], categories information of neighbouring is useful for classification. From this, we explore a number of link feature models namely *count-link* and *sum-labeling*.

Count-link features of a document represent the numbers of neighbouring documents in every categories. If the majority of the neighbouring documents are in a certain category, the document is likely to be in that category too. This *count-link* model is similar to that introduced in work by Lu and Getoor [16].

The second link feature model we use is *sum-labelling* features, which represent the summation of all the labellings from neighbouring documents. In other words, the value of each feature is the sum of all scores of the corresponding category from the labellings of neighbouring.

Both of the feature models are the combination of content-based classification of all neighbouring documents. The number of link features is the same as the number of categories.

In order to incorporate both content and citation information for classification, the input vector consists of the labelling from the content-based classifier and the vector of link features. That gives the size of the input vector to the neural network double the number of categories. If the number of classes is n then the input vector size will be $2n$. First n elements of the

input vector are the labelling of the document by the content-based classifier, and the rest are n elements from link features. Therefore, input layer the neural network has $2n$ nodes.

The output of the neural network is also a vector of scores for all categories. In this approach, we used a single hidden layer in the network architecture. The number of nodes in hidden layer is a trade-off between generalisation and training accuracy. In this work, we use a heuristic size of hidden layer which is three times the size of output layer.

The other customisations of the neural network are as follows. The activations function associated with network nodes are logistic function. All weights of connections are initialised randomly. The error function which the neural network aiming to minimise is sum of square errors from all training patterns.

The neural network needs to be trained from a set of training patterns. The training patterns should be similar to the pattern of desired computation. As the neural network is designed for updating the labelling from the content-based classifier, we use the output of the content-based classifier on training document set as the input for the neural network. The output of the training patterns are the real labelling of the training documents.

3.4.2 Summary of Neural Network Approach to Citation Link

The overview of the classification system using neural network is summarised as follows:

- Train a content-based classifier using training set. The content-based classification method in this approach, similar to the other approaches, is naive Bayes classification.
- Apply the content-based classifier to classify the training set. The labellings of documents from training set by the content-based classifier are used to form input patterns to train the neural network.
- Prepare training set for neural network. For each training document, create a training pattern. The input vector of the training pattern is the combination of the labelling from the content-based classifier and the link features as described above. The output vector is the true labelling of the document.
- Train the neural network from the training patterns created. The training process terminates when the number training epochs reaches a predetermined number of the error rate becomes smaller than a threshold.
- To classify the test set, the system first applies the content-based classifier to classify all documents of the corpus, then the trained network to improve/update the classification.

4 Experiment Results

4.1 Dataset

For experiments throughout this study, we use Cora, a real world scientific paper corpus collected by McCallum et al [18]. A subset of Cora collection is selected as a test bed for the work.

The test bed contains 3098 papers in seven subjects of machine learning: case based, probabilistic methods, learning theory, genetic algorithms, reinforcement learning, neural networks, and rule learning. We selected only documents that cite or are cited by the others. The proportion of papers in each topic ranges from 7% in rule learning topic to 32% in neural network topic.

These papers are connected by a network of 11713 citation links. Examine the citation network, we find that 76% of citations are between by papers of same class, while 24% of citations are cross topic.

For each experiment, we randomly select an equal portion of the papers from each of the seven categories for training a classifier, and use the rest as the test set to evaluate the classifier. We carry out experiments on different training set sizes, which are 20%, 30%, 40%, 50% and 60% of the collection to examine the effect of each method. For the ease of notation, we name each configuration of data set as CORAXX where XX is the percentage of training set. For example, the set CORA40 refers to using 40% of the collection as training set and 60% as test set.

Each configuration of the system is tested 10 times on different set of training set and test set. As an example, when experiment with training set of size 20%, on each run, the system select randomly 20% of each class to make up the training set. The size of training set on 10 runs are the same but the documents selected are different. The mean accuracy of the 10 runs are recorded and reported.

The training set is first used to train a content-based classifier. We choose to use naive Bayes for its simplicity and efficiency. The training set is then used to train the Bayesian citation network and neural network. For testing, we first apply the content-based naive Bayes classifier to compute the initial labellings, then use one of three methods to update labellings.

4.2 LLU Results

We experiment LLU in different updating rates. Their results are shown in tables 1. On the table, the left most column shows the configuration of dataset, which specifies the size of the training set. The second column shows the accuracy of the content-based classifier. The results of each setting are shown in three columns. The first column shows the mean accuracy of the classification system. The second column shows the improvement in accuracy in compared with using only NB for content-based classification and the last column shows the number of average iterations for the labelling updating process to converge.

Dataset	Cont. Acc	$\eta = 0.10$			$\eta = 0.15$			$\eta = 0.20$		
		Acc.	Impr.	Iter.	Acc.	Impr.	Iter.	Acc.	Impr.	Iter.
CORA20	67.79%	84.42%	16.63%	14	84.40%	16.61%	12	84.40%	16.61%	10
CORA30	74.57%	85.40%	10.83%	11	85.39%	10.04%	10	85.39%	10.82%	7
CORA40	77.25%	85.98%	8.73%	9	85.97%	8.72%	8	85.97%	8.72%	6
CORA50	83.35%	86.52%	3.17%	8	86.51%	3.16%	6	86.54%	3.20%	5
CORA60	85.05%	86.75%	1.70%	6	86.74%	1.69%	5	86.76%	1.71%	4
Dataset	Cont. Acc	$\eta = 0.25$			$\eta = 0.30$			$\mu = 0.35$		
		Acc.	Impr.	Iter.	Acc.	Impr.	Iter.	Acc.	Impr.	Iter.
CORA20	67.79%	84.40%	16.61%	10	84.41%	16.62%	9	84.41%	16.62%	8
CORA30	74.57%	85.39%	10.81%	6	85.39%	10.04%	7	85.39%	10.82%	6
CORA40	77.25%	85.96%	8.71%	5	85.92%	8.68%	5	85.95%	8.70%	5
CORA50	83.35%	86.51%	3.16%	4	86.49%	3.14%	5	86.51%	3.16%	5
CORA60	85.05%	86.75%	1.70%	4	86.71%	1.66%	4	86.73%	1.68%	3

Table 1: Improvement by LLU.

In general, experiments show that LLU with any settings always improves the classification system. Particularly, on CORA20 dataset, LLU improves the classification by 16.61% (from 67.79% to 84.40%). However, the improvement is less in the dataset with larger training set.

This is because the performance of the content-based NB classification performs well on large training set.

4.3 PLU Results

Table 2 shows results of PLU, the format of the table is similar to the tables described in the previous sub section.

Dataset	Content Acc	Acc.	Impr.	Iter.
CORA20	67.79%	78.86%	9.07%	186
CORA30	74.57%	82.95%	8.38%	153
CORA40	77.25%	85.43%	8.18%	81
CORA50	83.35%	87.82%	4.27%	75
CORA60	85.05%	88.08%	3.03%	62

Table 2: Improvement by PLU

The PLU method also produces significant improvement of classification accuracy whenever it converges. On CORA20 dataset, an improvement of more than 9% is gained while on CORA60, the improvement is about 3%. In small training set(CORA20 and CORA30), the PLU is inferior to LLU, but for larger training set, PLU outperforms LLU.

Comparing with LLU, the PLU requires much more iterations to converse. As an example, on CORA40 dataset, PLU updating method requires 81 iterations while LLU converses after average less 10 iterations.

4.4 Neural Networks Results

Table 3 shows results of neural network, the format of the table is similar to the tables described in the previous sub section, except that the neural network requires only one iteration for update the labellings.

Dataset	Content Acc	Acc.	Impr.
CORA20	67.79%	76.17%	8.38%
CORA30	74.57%	82.44%	7.87%
CORA40	77.25%	84.28%	7.03%
CORA50	83.35%	86.52%	3.17%
CORA60	85.05%	87.22%	2.17%

Table 3: Improvement by neural network

4.5 Further Analysis

Table 4 shows the comparison of the best results in terms of classification accuracy of the three methods LLU and PLU. The best results of LLU are taken when updating rate $\eta = 0.1$.

Comparing the three methods, we see that LLU is superior to the other two while the training set is small. It is suggested that, the LLU is an approximated modelling of the citation structure. While only a small training set or no training document is available, LLU is a good choice. It is noted that, LLU does not need any training examples to learn the model.

Dataset	Cont. Acc.	LLU		PLU		NN	
	Acc.	Impr.	Acc.	Impr.	Acc.	Impr.	
CORA20	67.79%	84.40%	16.61%	%	9.07%	%	8.38%
CORA30	74.57%	85.39%	10.81%	%	8.38%	%	7.87%
CORA40	77.25%	85.96%	8.71%	%	8.18%	%	7.03%
CORA50	83.35%	86.51%	3.16%	%	4.27%	%	3.17%
CORA60	85.05%	86.75%	1.70%	%	3.03%	%	2.17%

Table 4: Comparison of LLU and PLU.

Nevertheless, the PLU and NN outperform LLU while the training set is sufficiently large. The two machine learning methods is able to generate the hidden patterns of the citation structure, which are better model the scientific citation than the static LLU. The reason for these two methods' inferiority to LLU is that, a small training set may not enable PLU and NN to generate a sufficiently good rules for the citation structure.

Examining training experiments of PLU, we observe that the parameters learnt by PLU in CORA50 and CORA60 are quite similar while there are differences between those parameters and those learnt from CORA20 and CORA30. It is suggested that the CORA50 is about enough for training BLU while CORA20 and CORA30 are too small for PLU. That also explains the poor performance of PLU on small training set.

While PLU does make improvement, the improvement is not as big as expected. We are expecting that, the message passing "send message anyway" causes imprecision of information passing. It is expected that some exact inference algorithms such as junction tree [13], cutset conditioning [12] or clustering [25] would perform better in information propagation.

Probably Approximately Correct (PAC) is the study about conditions necessary for valid generalisation in machine learning. The measurement of learning capacity of a system is Vapnik-Chervonenis (VC) dimention, which can be defined as the number of weights in the neural network [3]. A heuristic guideline for the size of training set is the VC dimension times the inverse of the expected error rate. For example, to train the above neural network to reach the error rate of 10%, the number of training patterns required will be:

$$\frac{1}{10\%} \times (7 \times 14 + 14 \times 21) = 4410 \tag{16}$$

As in the system, the numbers of training examples are much smaller than that number (CORA60 set has only 1800 training examples). Therefore, if more training examples are available, the system accuracy for neural network could increase.

5 Conclusions

Our investigation of citation links shows that the citation structure can be combined with one of the content-based classification method. Our framework for combining document contents and citation links includes a content-based classifier and an updating function. The labellings of papers computed by the content-based classier are updated by the updating function using categories information from neighbouring documents.

We developed three labelling updating methods, the linear labelling update(LLU), the probabilistic labelling update(PLU) and the neural network(NN) method. The LLU method updates the labelling of a document by adding the labellings of its neighbours multiplied by a constant. The PLU applies the Bayesian network to model the citation structure and uses message passing algorithm for propagate category information around the citation network

while the NN method model the updating function by a neural network. The LLU and PLU updates labellings iteratively until system converses while the NN updates only round.

For learning citation model, the PLU and NN requires some training samples while LLU is a static model and does not need training data. Both PLU and NN need to have sufficiently enough training to perform well. They only outperform LLU when a large training set is used.

While the work does achieve its goals, there is much room for further development. Some of possible directions to be considered are:

- The successes of PLU and NN suggest that the hidden characteristics of citations can be learned. Some other learning approaches such as genetic programming could be investigated to learn the citation structure.
- The inference approach currently applied by PLU, which is "send message anyway", is not generally considered good in Bayesian network inference as it introduces imprecision and divergences. Other more reliable inference algorithms such as junction tree, cutset conditioning and clustering could be investigated to design a better updating mechanism.
- Instead of using message passing algorithm, an approximate inference algorithm could also be used. Some examples of these are sampling, variational methods and loopy propagation.
- A number of temporal approaches such as hidden Markov model or dynamic Bayesian network are suggested to be used for modelling the citation structure.

References

[1] Citeseer, http://citeseer.ist.psu.edu/.

[2] Google Scholar, www.scholar.google.com.

[3] Baum, E. B., and Haussler, D. What size net gives valid generalization? *Neural Comput. 1*, 1 (1990), 151–160.

[4] Brin, S., and Page, L. The anatomy of a Large-scale Hypertextual Web search Engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117.

[5] Chakrabarti, S., Dom, B. E., and Indyk, P. Enhanced hypertext categorization using hyperlinks. In *Proceedings of SIGMOD-98, ACM International Conference on Management of Data* (Seattle, US, 1998), L. M. Haas and A. Tiwary, Eds., ACM Press, New York, US, pp. 307–318.

[6] Cowell, R. Introduction in Inference in Bayesian Networks. MIT Press, Cambridge, MA, USA, 1999, pp. 9–26.

[7] Craven, M., and Slattery, S. Relational Learning with Statistical Predicate Invention: Better Models for Hypertext. *Mach. Learn. 43*, 1-2 (2001), 97–119.

[8] Getoor, L., Segal, E., Taskar, B., and Koller, D. Probabilistic Models of Text and Link Structure for Hypertext Classification. In IJCAI Workshop on Text Learning: Beyond Supervision, 2001.

[9] Ghahramani, Z. Graphical Models: Parameter Learning. In *Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed., 2 ed. MIT Press, 2003, pp. 486–490.

[10] Han, E.-H., and Karypis, G. Centroid-Based Document Classification: Analysis and Experimental Results. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery* (London, UK, 2000), Springer-Verlag, pp. 424–431.

[11] Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (Chemnitz, DE, 1998), C. Nédellec and C. Rouveirol, Eds., Springer Verlag, Heidelberg, DE, pp. 137–142. Published in the "Lecture Notes in Computer Science" series, number 1398.

[12] Jordan, M. I., and Weiss, Y. Graphical Models: Probabilistic Inference. In *Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed., 2 ed. MIT Press, 2003, pp. 490–496.

[13] Lauritzen, S. L., and Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. 415–448.

[14] Lewis, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning* (Chemnitz, DE, 1998), C. Nédellec and C. Rouveirol, Eds., Springer Verlag, Heidelberg, DE, pp. 4–15. Published in the "Lecture Notes in Computer Science" series, number 1398.

[15] Lewis, D. D., and Ringuette, M. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval* (Las Vegas, US, 1994), pp. 81–93.

[16] Lu, Q., and Getoor, L. Link-based classification. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)* (Washington DC, 2003), ACM Press, New York, US.

[17] MacKay, D. J. C. Introduction to Monte Carlo Methods. In *Learning in graphical models*. MIT Press, Cambridge, MA, USA, 1999, pp. 175–204.

[18] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval 3*, 2 (2000), 127–163.

[19] Nigam, K., Lafferty, J., and McCallum, A. Using Maximum Entropy for Text Classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering* (1999), pp. 61–67.

[20] Oh, H.-J., Myaeng, S. H., and Lee, M.-H. A practical hypertext categorization method using links and incrementally available class information. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (Athens, GR, 2000), N. J. Belkin, P. Ingwersen, and M.-K. Leong, Eds., ACM Press, New York, US, pp. 264–271.

[21] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann publishers, 1988.

[22] Quinlan, J. R. Learning logical definitions from relations. *Mach. Learn. 5*, 3 (1990), 239–266.

[23] Ruiz, M. E., and Srinivasan, P. Hierarchical neural networks for text categorization. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, US, 1999), M. A. Hearst, F. Gey, and R. Tong, Eds., ACM Press, New York, US, pp. 281–282.

[24] Rumelhart, D., and McClelland, J. *Parallel Distributed Processing*, vol. 1 and 2. MIT Press, 1986.

[25] Russell, S., and Novig, P. *Artificial Intelligence: A Modern Approach*, 2 ed. Prentice Hall, 2005.

[26] Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys 34*, 1 (2002), 1–47.

[27] Taskar, B., Segal, E., and Koller, D. Probabilistic Classification and Clustering in Relational Data. In *Proceeding of IJCAI-01, 17th International Joint Conference on Artificial Intelligence* (Seattle, US, 2001), B. Nebel, Ed., pp. 870–878.

[28] Yang, Y. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval* (Dublin, IE, 1994), W. B. Croft and C. J. Van Rijsbergen, Eds., Springer Verlag, Heidelberg, DE, pp. 13–22.

[29] Yedidia, J. S., Freeman, W. T., and Weiss, Y. Understanding Belief Propagation and its Generalizations. Tech. Rep. TR-2001-22, Mitsubishi Electric Research Laboratories, Inc., jan 2002.