Intelligent GP Fusion from Multiple Sources for Text Classification

Baoping Zhang, Yuxin Chen, Weiguo Fan, Edward A. Fox
Virginia Tech
Dept. of Computer Science
Blacksburg, VA, USA
{bzhang, yuchen, wfan, fox}@vt.edu

Marcos Gonçalves, Marco Cristo
Federal University of Minas Gerais
Dept. of Computer Science
Belo Horizonte, MG, Brazil
{mgoncalv, marco}@dcc.ufmg.br

Pável Calado
IST/INESC-ID
Lisbon, Portugal
pavel@tagus.ist.utl.pt

ABSTRACT

This paper shows how citation-based information and structural content (e.g., title, abstract) can be combined to improve classification of text documents into predefined categories. We evaluate different measures of similarity – five derived from the citation information of the collection, and three derived from the structural content – and determine how they can be fused to improve classification effectiveness. To discover the best fusion framework, we apply Genetic Programming (GP) techniques. Our experiments with the ACM Computing Classification Scheme, using documents from the ACM Digital Library, indicate that GP can discover similarity functions superior to those based solely on a single type of evidence. Effectiveness of the similarity functions discovered through simple majority voting is better than that of content-based as well as combination-based Support Vector Machine classifiers. Experiments also were conducted to compare the performance between GP techniques and other fusion techniques such as Genetic Algorithms (GA) and linear fusion. Empirical results show that GP was able to discover better similarity functions than GA or other fusion techniques.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; I.5.4 [Pattern Recognition]: Applications–*Text processing*

General Terms: Algorithms, Measurement, Performance, Experimentation

Keywords: Classification, Genetic Programming

1. INTRODUCTION

In recent years, automated classification of text into predefined categories has attracted considerable interest, due to the increasing volume of documents in digital form and the ensuing need to organize them. However, traditional content-based classifiers are known to perform poorly when documents are noisy (e.g., digitized through speech recognition or OCR) and/or contain scarce textual content (e.g., metadata records in digital library (DL) catalogs) [3].

DLs offer both (1) the opportunity to explore the complex internal structured nature of documents and metadata records in the classification task; and (2) the social networks occuring in specific communities, as expressed for example by citation patterns in the research literature. On the other hand, many DLs, which are created by aggregation of other sub-collections/catalogs, suffer from problems of quality of information. One such problem is incompleteness (e.g., missing information). This makes it very hard to classify documents using traditional content-based classifiers like SVM, or Naive Bayes. Another quality problem is imprecision. For example, citation-based information is often obtained with OCR, a process which produces a significant number of errors. In this work we try to overcome these problems by applying automatically discovered techniques for fusion of the available evidence. Particularly, we investigate an inductive learning method – Genetic Programming (GP) – for the discovery of better fused similarity functions to be used in the classifiers, and explore how this combination can be used to improve classification effectiveness.

One motivation of this work is to enhance teaching and learning in the computing field, by improving CITIDEL [5], part of the National Science Digital Library (NSDL). Many of the records in CITIDEL lack classification information. That makes it difficult for users to browse, even with the support of our advanced multischeming approach [26]. It also limits the scope of advanced interfaces to CITIDEL, which make use of category information [21]. Further, the lack of classification information reduces broader downstream impact that could result from use of records harvested from CITIDEL and similar systems, such as into NSDL or NDLTD (e.g., to support browsing of computing dissertations [37]).

This paper provides background in Section 2, especially about GP. Section 3 highlights challenges in classification. Section 4 describes our experiments. Section 5 discusses related work, while Section 6 gives conclusions and describes plans for ongoing research.

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

2. BACKGROUND

2.1 Genetic Programming

Genetic algorithms (GAs) [18] and genetic programming (GP) [25] are a set of artificial intelligence search algorithms designed following the principles of biological inheritance and evolution. The solution to a problem is represented as an individual (i.e., a chromosome) in a population pool. The population of individuals evolves generation by generation through genetic transformation operations – such as reproduction, crossover, and mutation – with the aim of creating more diverse and better performing individuals with better fitness values in subsequent generations. A fitness function is available to assign the fitness value for each individual.

The difference between GA and GP is the internal representation – or data structure – of the individual. In GAs, each individual is commonly (though not always) represented by a fixed-length bit string, like (1101110 . . .) or a fixed-length sequence of real numbers (1.2, 2.4, 4, . . .). In GP, more complex data structures, e.g., trees, linked lists, or stacks, are used [27]. Moreover, the length or size of the data structure is not fixed, although it may be constrained by implementation to within a certain size limit. GAs are often used to solve difficult optimization problems, while GP is typically used to approximate complex, nonlinear functional relationships [25]. Because of the intrinsic parallel search mechanism and powerful global exploration capability in a high-dimensional space, both GA and GP have been used to solve a wide range of hard optimization problems that oftentimes have no best known solutions.

2.2 The Application of GA/P to IR

Both GP and GA have been applied to the information retrieval field [12,17] as well as for data classification. In [4], Cheang et al. proposed a Genetic Parallel Programming classifier to evolve parallel programs for data classification problems. Eggermont et al. [10] proposed several methods using techniques from the field of machine learning to refine and reduce search space sizes for evolving decision trees for data classification. They showed how classification performance improves when shrinking the space in which a tree-based Genetic Programming (GP) algorithm searches. Kishore et al. [23] proposed a methodology for GP-based n-class pattern classification. They modelled the given n-class problem as n two-class problems; further, a GP classifier expression was evolved as a discriminant function for each class. In [24], they integrated the GP classifier with feature space partitioning for localized learning to improve pattern classification. Genetic Programming also has been applied to text classification through the use of a parse-tree. In [6], Clack et al. used GP to route in-bound documents to a central classifier which autonomously sent documents to interested research groups within a large organization. The central classifier used a parse tree to match the aspects of a document to nodes of the tree, which ultimately leads to a single numerical value – the classification or "confidence value" – during evaluation.

Castillo et al. [9] developed a multistrategy classifier system for document classification. They applied different types of classifiers (e.g., Naive Bayes, Decision Trees) to different parts of the document (e.g., titles, references). Genetic algorithms are applied for feature selection as well as for combining the output of the different classifiers.

2.3 Rationale of using GP for Text Classification

To fuse multiple sources of evidence for text classification, we could use techniques like manual selection and majority voting. However, manual selection and combing is very tedious and time-consuming, while simple majority voting may not work in all contexts. A better way is to use machine learning techniques to intelligently combine evidence, by optimizing a given objective function.

The decision to use GP for fusing multiple sources of evidence to discover similarity functions for text classification was motivated by four factors:

1. The large size of the search space;

   A similarity function can be represented in a tree structure. For our trees we have used 14 terminals (see Section 2.4), four functions, trees of depth up to 16, and real constants which can vary between 0 and 1. Langdon et al. [28] show that for these parameters the search space for a tree is very large – a needle-in-a-haystack problem. Other search mechanisms like random search and exhaustive search would take inordinate amounts of time. GP has been shown to perform well under such conditions.

2. The characteristics of the objective function;

   Many performance measures in IR are discrete. GP does not require that objective functions be continuous in nature as long as it can distinguish good solutions from bad ones [25].

3. Previous success on the application of GP in the IR field as well as in data classification;

   Fan et al. previously reported success in applying GP to discover ranking functions for both individual queries (information routing tasks) [12] and multiple queries (ad-hoc retrieval tasks) [13]. The success of GP, in discovering nonlinear functions and structures in other data mining domains [25], also motivated us to use GP for similarity function discovery.

4. Little prior work on applying GP to large scale text classification.

   We seek to fill the near-void that exists in this area; our approach appears to have promise for broad impact.

2.4 GP Configurations

In order to apply GP to the problem of classification, several required key components of a GP system need to be defined: Terminals are the leaf nodes in the tree structure. Functions are the non-leaf nodes combining the leaf nodes. Numerical operations like +, -, *, /, and log are commonly used. A fitness function is the objective function GP aims to optimize. Reproduction is a genetic operator to breed new individuals, while mutations are deviations during this reproduction process. Crossover takes two individuals (parents) to breed one new individual that shares some attributes with each parent.

When setting up the configurations of the GP system for similarity function discovery, we combined features regarding content-based structural information and features regarding citation-based information to use as the terminals.

To determine the similarity between two documents we used three different similarity measures applied to the content of abstract, title, and abstact-plus-title (denoted as full) of documents separately: Bag-of-Words (BOW), Cosine and Okapi. Also, we used five different citation-related similarity measures: co-citation [32], bibliographic coupling [22], Amsler [1], and Companion (authority and hub) [8]. More information about these measures can be obtained in [36]. This gave us fourteen similarity measures, represented as document × document matrices. Through GP, we intend to discover a single similarity function, for each class, that combines all or several of the similarity measures described here. Table 1 shows the mapping of all these measures to their abbreviations which are used later in the paper.

Similarity Measures	Abbreviations
Abstract_BagOfWords	Abs_BOW
Abstract_Cosine	Abs_Cos
Abstract_Okapi	Abs_Okapi
Full_BagOfWords	Full_BOW
Full_Cosine	Full_Cos
Full_Okapi	Full_Okapi
Title_BagOfWords	Title_BOW
Title_Cosine	Title_Cos
Title_Okapi	Title_Okapi
Amsler	Amsler
Bibliographic Coupling	Bib_Coup
Co-citation	Co-Cit
Companion_Authority	Comp_Aut
Companion_Hub	Comp_Hub

Table 1: Measures and corresponding abbreviations.

The functions used are +, *, /, sqrt. These functions are selected because they provide meaningful operations on relations. For example, the + and * operations are summing and reinforcing the relations, respectively. On the other hand, the / operation accomodates inverse relationships, while the sqrt operation serves to scale the values. The algorithm below details our fitness evaluation function which GP should optimize within a particular class.

```
Let R = 0
For each document D in class C
     Find the K documents most similar to D
     Predict a class for D according to the kNN
       algorithm (Section 3) using the K
       documents as the K nearest neighbors
     If this is a correct prediction, R = R + 1
end for
Let F = R/|C|
```

A good similarity function, i.e., a similarity function with a high fitness value, is one that, when applied to a document d_i of class C, ranks documents from class C in such a way that those with greater similarities to d_i are top-ranked. The higher the value of F, the better the function. It is worth noticing that the choice of fitness function can have a huge impact on the final classification performance [11]. Experiments with different fitness functions are currently underway.

3. CLASSIFICATION FRAMEWORK

3.1 Framework

Along with the settings discussed in the previous section, the overall classification framework has 5 steps, as follows:

1. For each class, generate an initial population of random trees, each representing a similarity function.
2. For each class, perform the following sub-steps on training documents for N_{gen} generations.
   (a) Calculate the fitness of each similarity tree.
   (b) Record the top N_{top} similarity trees.
   (c) Create a new population by: reproduction, crossover, and mutation.
3. Apply the recorded (N_{gen} * N_{top}) candidate "similarity trees" to a set of validation documents and select the best performing tree b_C as the unique best discovered similarity tree for each class C.
4. For each class C, use b_C as similarity function in a *kNN* classifier (see below) and apply this resulting classifier to a set of testing documents.
5. Combine the output of each classifier through a simple majority voting.

Steps 1, 2, and 3 concern the training process within GP which intends to discover better similarity functions for each class. However, the discovered functions only can be used to calculate the similarity between a pair of documents. In order to evaluate the performance of those functions in the classification task, we used a strategy based on a nearest neighbor classifier – *kNN* [34]. This classifier assigns a category label to a test document, based on the categories attributed to the k most similar documents in the training set. The *kNN* algorithm was chosen since it is simple and makes direct use of similarity information.

In the *kNN* algorithm, to a given test document d is assigned a relevance score $s_{c_i,d}$ associating d with each candidate category c_i. This score is defined as:

$$s_{c_i,d} = \sum_{d' \in \mathcal{N}_k(d) \wedge class(d') = c_i} similarity(d, d') \qquad (1)$$

where $\mathcal{N}_k(d)$ are the k nearest neighbors (the most similar documents) of d in the training set. In Step 4 of our framework the generic similarity function of *kNN* is replaced by the functions discovered by GP for each class.

In multi-classification problems with n classes, we effectively end up with n *kNN* classifiers using the described framework. In order to produce a final classification result, we combine the output of all n classifiers using a simple *majority voting* scheme, whereby the class of a document d_i is decided by the most common class assigned by all the n classifiers. In case of ties, we assign d_i to the larger class. Because of its simplicity we chose to use majority voting in our framework (Step 5) to: 1) help alleviate the common problem of overfitting found in GP training [12] and; 2) help boost performance by allowing *kNN* classifiers to apply different similarity functions which explore and optimize the characteristics of each particular class in different ways.

3.2 Theoretical and Practical Properties

The classification framework suffers from scalability, a common problem with GP. That is, it takes a relatively long time to find an optimal solution. In fact, the time complexity of an experiment based on the above framework is O(N_{gen} * N_{ind} * T_{eval}), where N_{gen} is the number of generations for evolution, N_{ind} is the number of individuals in a

population pool, and T_{eval} is the fitness evaluation time for an individual. Since T_{eval} is determined by the complexity of an individual (Size) and the number of training samples ($N_{samples}$), then the total time complexity of an experiment is O(N_{gen} * N_{ind} * Size * $N_{samples}$) [11]. Nevertheless, scalability and computing efficiency are not the main issue in our experiments; we focus more on the effectiveness of the classifiers. The speed of the learning process can improve through parallel processing as well as through optimization, e.g., as described in Section 4.1. Further, in a practical application, the learning process need occur only occasionally; the frequent operation will be the actual classification of incoming new documents.

4. EXPERIMENTS

To test the hypotheses that GP is able to adapt itself to find the best similarity functions, we ran two sets of experiments following the framework of the previous section on both the first level (11 categories, A to K) and the second level (44 categories, though some small categories were removed, see Section 4.1) of the ACM Computing Classification System (CCS, `http://www.acm.org/class/1998/`). For the first level experiments, a subset of the ACM collection with 30K metadata records corresponding to those classified under only one category in the first level was used. Correspondingly, for the second level experiments, another ACM collection subset was used, with 19K metadata records drawn from those classified under only one second level category.

The ACM Digital Library suffers from most of the problems we mentioned in the Introduction. For example, only 42.3% of the records in the first level collection and only 41.7% of the records in the second level collection have abstracts, which makes it very hard to classify them using traditional content-based classifiers. For these records, the only available textual content is title. But titles contain normally only 5 to 10 words. Citation information was created with OCR and had a significant number of errors. A very imprecise process of matching between the citation text and the documents, using adaptations of techniques described in [29], had to be performed. This introduced noise and incompleteness in the citation-based similarity matrices computed with measures such as co-citation or bibliographic coupling. Other concerns were the large search space and skewed distributions observed for some categories.

Stratified random sampling (cf. Section 4.1) was used to create training and test collections. The combination of these experiments provides us with insights about the capability of the GP-based discovery framework.

4.1 Sampling

The collection used in our experiments has 30,022 documents belonging to 11 categories for the first level and 18,664 documents belonging to 44 categories for the second level. Figure 1 shows the distributions for the first level and second level categories, respectively. Each terminal or feature described is a similarity matrix which contains the similarity between each pair of documents. Using half or more of the whole collection as our training data, the required resources – in CPU time and amount of memory – would be enormous. The time required to discover a proper classification framework also would be significant. To reduce the high cost of resources and at the same time improve efficiency, sampling was used. So, we started by considering the documents belonging to each category of CCS as different population strata. We then randomly selected from each stratum a given number of units based on a proportion, like 15%, for each category. However, special attention was paid to skewed categories. For example, category E in the first level only has 94 documents while the average size of the other categories is in the range of thousands. 15% of 94 only gives us 14 documents and this was too small to serve as the sample to discover the whole category's characteristics. The number of documents for some second level categories is too small (< 50) for classification purposes, so those categories were dropped from the experiments. Also, because of its size, category E was not subdivided in the second level and all of its ACM sub-categories were considered as a unique second-level one (denoted E.x). Classification statistics for both the first level and the second level collection were used to control the sampling procedure. That is, baselines from the samples for each level were compared with baselines for that level's corresponding whole collection, to ensure that the samples mirror that level's whole collection as well as possible.

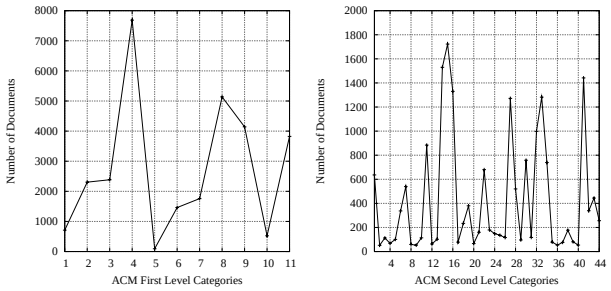

Figure 1: Compared distribution for the first and the second level ACM collections.

4.2 Experimental Data Set

We follow a three data-sets design [12, 30] in our experiment. We randomly split the data into training, validation, and test parts. The introduction of the validation dataset is to help alleviate the problem of overfitting of GP on the training data and select the best generalizable similarity function. We generate two sets of training samples using a stratified sample strategy for both the first level and the second level in the CCS. The first set used a random 15% sample for large classes and 50% for skewed classes. The second set used a random 30% sample for large classes and 50% for skewed classes[1]. In the 30% sample, one specific strategy was used for the biggest class D in the first level to reduce time. Since class D is huge, even using 30% for training in GP would require a long time. Thus, instead of a 30% sample, we save time by using 20% for training for class D; we think that sample is enough for GP to discover this class' best fusion of features for classification purposes. For the 15% (50% for skewed classes) training sample, a random 10% (25% for skewed classes) sample of the collection is used for validation and the rest of the samples are used for testing. Correspondingly, for the 30% (20% for class D

[1]In the remainder, we use 15% to refer to the first sample set and 30% to refer to the second sample set for each level, respectively.

and 50% for skewed classes) training sample, a random 20% (15% for class D and 25% for skewed classes) sample of the collection is used for validation and the rest of the samples are used for testing. All approaches reported in later sections use the same training and test sets. Results are based on test data sets only.

4.3 Baselines

In order to demonstrate that the combination of different features by GP is able to provide better classification results, we need to compare it with the classification statistics of each feature in isolation (baselines). We used F1 as our comparison criteria. F1 is a combination of precision and recall. Precision p is defined as the proportion of correctly classified records in the set of all records assigned to the target class. Recall r is defined as the proportion of correctly classified records out of all the records having the target class. $F1 = \frac{2pr}{p+r}$. It is worth noting that F1 is a balanced combination of precision and recall. It reduces the risk that you can get perfect precision by always assigning zero categories, or perfect recall by always assigning every category. The result we want is to assign the correct categories and only the correct categories, maximizing precision and recall at the same time, thereby maximizing F1. Macro-averaging (F1 results are computed on a per-category basis, then averaged over categories) was applied to get a single performance value over all categories. Tables 2 and 3 show the type of evidence that performs the best, according to macro F1, when applied to a *kNN* algorithm for a specific category in isolation in the test collections for the first level and the second level, respectively. Table 4 shows the average macro F1 over all categories for both the first level and the second level for each similarity evidence, also in isolation.

Class	Macro F1/ class(15%)	Best Evidence (15%)	Macro F1/ class(30%)	Best Evidence (30%)
A	27.76	Full_BOW	37.31	Title_BOW
B	70.34	Full_Okapi	73.55	Full_Cos
C	67.15	Full_Okapi	67.53	Full_Okapi
D	73.97	Full_Okapi	77.38	Full_Okapi
E	43.75	Full_Okapi	27.59	Full_Cos
F	51.59	Full_Okapi	56.96	Full_Cos
G	67.75	Full_Cos	70.88	Full_Cos
H	72.83	Full_Okapi	72.59	Full_Okapi
I	65.97	Full_Okapi	67.86	Full_Okapi
J	22.34	Full_Cos	20.12	Full_Okapi
K	73.19	Full_Okapi	73.74	Full_Okapi

Table 2: Best baseline for each category (first level).

From Tables 2 and 3 it can be seen that full-based evidence is most important for the majority of the classes. For those classes whose best performer was citation-based evidence, 1) Amsler had the highest value for category I.3 for both the 15% and 30% samples; 2) Amsler, Bib_Coup, and Comp_Hub were the most frequent evidence for the 15% sample; and 3) Amsler and Comp_Aut were the most frequent evidence for the 30% sample. From Table 4 it can be seen that the best types of evidence are the full-based ones, followed by title-based, citation-based, and abstract-based evidence, respectively. This should be expected since: full is the evidence which appears in all the documents and contains the most complete information, the titles are relatively short, many documents are missing abstracts, and the information provided by the citation structure is very incomplete and imprecise.

Class	Macro F1/ class(15%)	Best Evidence (15%)	Macro F1/ class(30%)	Best Evidence (30%)
A.0	48.05	Full_Okapi	55.57	Full_Okapi
A.1	9.52	Full_BOW	0.00	-
B.1	0.00	-	44.44	Full_Okapi
B.3	28.13	Abs_Okapi	39.13	Full_Okapi
B.5	24.00	Bib_Coup	31.25	Abs_Cos
B.6	53.53	Full_Cos	55.56	Full_Cos
B.7	56.29	Full_Okapi	56.19	Full_Okapi
B.8	15.38	Bib_Coup	9.52	Full_Okapi
C.0	0.00	-	12.50	Full_Okapi
C.1	21.05	Full_Okapi	27.78	Full_Cos
C.2	68.29	Full_Okapi	70.91	Full_Okapi
C.4	5.88	Comp_Hub	6.67	Comp_Hub
D.1	17.54	Full_Okapi	17.78	Full_Cos
D.2	59.10	Full_Okapi	60.91	Full_Okapi
D.3	65.37	Full_Okapi	66.37	Full_Okapi
D.4	64.19	Full_Okapi	65.57	Full_Okapi
E.x	24.24	Full_Okapi	44.44	Title_Cos
F.1	37.59	Bib_Coup	38.33	Comp_Aut
F.2	24.29	Amsler	37.26	Amsler
F.3	14.29	Comp_Aut	22.86	Comp_Aut
F.4	32.14	Amsler	43.18	Full_Okapi
G.1	66.94	Full_Cos	69.46	Full_Cos
G.2	23.30	Abs_BOW	44.04	Full_Cos
G.3	37.50	Abs_BOW	45.16	Comp_Aut
G.4	17.65	Comp_Hub	32.50	Full_Okapi
H.1	12.99	Full_Cos	8.33	Amsler
H.2	71.44	Full_Cos	75.07	Full_Cos
H.3	56.01	Full_Okapi	61.07	Full_Okapi
H.4	26.67	Full_Okapi	32.43	Title_Okapi
H.5	50.30	Full_Cos	54.88	Full_Okapi
I.1	40.00	Full_Cos	43.33	Full_Cos
I.2	59.77	Full_Okapi	66.44	Full_Okapi
I.3	69.42	Amsler	70.58	Amsler
I.6	72.33	Full_Okapi	72.97	Full_Okapi
I.7	32.65	Full_Okapi	25.00	Title_Cos
J.1	10.00	Comp_Hub	0.00	-
J.3	45.00	Full_Cos	60.61	Full_Cos
J.5	18.35	Full_Cos	13.51	Title_Cos
J.6	32.00	Full_Cos	55.81	Full_Okapi
K.1	33.33	Title_BOW	21.05	Title_BOW
K.3	75.37	Full_Cos	76.03	Full_Cos
K.4	19.94	Full_Cos	27.15	Title_BOW
K.6	23.42	Full_Okapi	24.92	Full_Okapi
K.7	20.00	Full_BOW	38.98	Full_Okapi

Table 3: Best baseline for each category (second level).

4.4 Experimental Set Up

We ran experiments on the two training samples using different parameters. We noticed that a larger population size and different rate for the genetic transformation operations like crossover, reproduction, and mutation produce better results. On the other hand, they have a huge effect on the training time. We used 400 as population size, 65% crossover, 30% reproduction, 5% mutation, and 30 generations as our GP system experimental setting. In the next section, we only report performance of the best tree in the validation sets applied to the test sets.

4.5 Experimental Results

We demonstrate the effectiveness of our classification framework in several ways: 1) by comparing its performance against the best baselines per class in isolation; 2) by comparing it against a majority voting of classifiers using those best baselines as similarity functions; 3) by comparing our experimental results with the results achieved through a linear combination of both the content-based and structure-based information through SVM; and 4) by comparing our experimental results with the results achieved through a content-

Class	Majority Best Evidence		Linear Fusion		GA		Majority GP		Content-based SVM		Combination-based SVM	
	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)
A	24.54	34.68	22.01	31.81	20.70	31.13	25.36	35.51	27.33	36.24	26.36	35.52
B	69.48	74.66	66.99	72.35	70.44	74.27	78.48	79.90	69.29	70.94	69.80	71.79
C	67.95	69.22	66.96	68.11	68.34	69.04	72.18	71.45	64.29	64.98	64.93	65.74
D	68.14	78.04	70.73	77.66	72.98	78.87	76.55	81.93	72.66	77.37	74.04	77.95
E	34.48	22.22	24.24	20.00	29.41	20.00	37.84	21.62	14.29	15.69	12.34	15.39
F	48.32	57.42	46.81	55.89	50.50	58.23	58.80	64.58	53.06	54.07	54.11	55.30
G	66.81	72.77	68.38	72.53	69.89	75.03	73.55	77.29	68.05	69.31	68.21	70.05
H	71.37	73.10	69.37	70.48	71.65	72.78	77.56	77.86	71.11	71.91	71.22	72.49
I	60.25	67.86	62.45	66.95	65.14	69.83	71.62	74.27	64.62	64.80	64.75	65.42
J	16.39	14.01	24.19	21.71	24.22	21.84	29.54	29.03	13.26	20.56	13.55	24.65
K	72.63	74.73	71.22	72.20	73.15	73.36	76.09	75.81	71.48	71.15	72.55	72.14
Avg F1	54.58	58.07	53.94	57.24	56.04	58.58	61.60	62.66	53.57	56.09	53.80	56.95

Table 5: Comparison between Majority Voting using the best evidence per class in isolation, Linear Fusion, GA, combined majority GP, content-based and combination-based SVM for the first level.

Evidence	First Level		Second Level	
	(15%)	(30%)	(15%)	(30%)
Abs_BOW	18.31	24.22	11.36	9.50
Abs_Cos	32.41	34.97	18.80	20.82
Abs_Okapi	33.21	34.14	18.81	19.49
Full_BOW	40.01	45.76	22.87	23.31
Full_Cos	53.40	55.80	32.31	36.39
Full_Okapi	57.17	57.37	32.63	36.64
Title_BOW	44.87	49.34	26.27	30.39
Title_Cos	49.10	51.58	28.45	34.01
Title_Okapi	48.36	52.08	28.40	33.98
Bib_Coup	30.85	33.96	17.84	20.49
Amsler	36.90	41.09	21.21	24.41
Co-Cit	21.73	26.88	12.71	15.06
Comp_Aut	31.43	35.77	17.39	21.33
Comp_Hub	33.40	36.85	19.02	22.24

Table 4: Macro F1 on individual evidence.

based SVM classifier[2]. While the last comparison may seem inappropriate since the classifiers are trained and applied to different types of content, it does provide a good idea of the core performance of our method, clearly showing it as a valid alternative in classification tasks similar to the ones used in this paper.

The SVM classifier has been extensively evaluated for text classification on reference collections, thus offering a strong baseline for comparison. A content-based SVM classifier was first used in text classification by Joachims [19]. It works over a vector space, where the problem is to find a hyperplane with the maximal margin of separation between two classes. This hyperplane can be uniquely constructed by solving a constrained quadratic optimization problem, by means of quadratic programming techniques.

Joachims et al. [20] also have shown how to combine different similarity measures by means of composite kernels. Their approach consists of combining simple and well-understood kernels by a series of "kernel preserving" operations, hence constructing an increasingly matching feature space S. In order to apply their method, we started by noticing that each one of our types of evidence can be represented as positive document-by-document matrices, like those presented in Section 2.4. So, we can represent each type of evidence as a kernel matrix $\mathcal{K}_{evidence}$ with elements $\mathcal{K}_{evidence}(d_i, d_j)$, where d_i and d_j are document vectors. The kernel matrix for our final feature space S is obtained by means of a linear combination of our initial kernel matrices, as described in Eq. 2.

[2]For content we used a concatenation of title + abstract.

$$\mathcal{K}_{combined}(d_i, d_j) = \frac{1}{N} \sum_{\forall evidence} \mathcal{K}_{evidence}(d_i, d_j) \quad (2)$$

where N is the number of distinct kinds of evidence used. Finally, the classification decisions for the combined kinds of evidence are obtained by applying a linear SVM classifier in our final feature space S.

For completeness, we also compare our classification framework against a classifier using a simple linear fusion of evidence as the similarity function and against a classifier using the similarity function discovered through GA. The similarity function used in linear fusion is simply represented through a summation of all the evidence, while the similarity function discovered through GA is represented by a weighted linear combination of all evidence, where the weights are assigned by the GA training process.

In a comparison, class by class, for the first level, between the majority GP (Table 5 columns 8 and 9) and the best evidence in isolation (Table 2), the majority GP outperforms the best evidence in most of the classes in both samples (only the performance for class A and E are worse). It also can be seen from Table 6 (columns 8 and 9) and Table 3 that this is true for most of the classes in the second level. There are two major reasons that the majority GP didn't excel in a few classes (like classes A.1, B.8, C.0, C.4, D.1, F.3, J.1, and K.1) in the second level. One is that the numbers of documents belonging to these classes are too small, so GP could not discover enough characteristics of those classes for classification purposes. For example, class A.1 only has 52 documents while class C.0 only has 53 documents. The other reason is that the best baselines for those classes (see Table 3) are poor, as compared with other classes. This means that we do not have good types of evidence for those classes.

When comparing the majority GP against the majority using the best evidence on the first level (Table 5 between columns 2,3 and 8,9) it is clear that majority GP presents better performance: we obtain a gain of 12.86% in the 15% sample and 7.90% in the 30% sample. This is also true for the second level: we obtain a gain of 10.05% in the 15% sample and 10.42% in the 30% sample.

When majority GP is compared with the combination-based SVM classifiers, the gains are even higher: we obtain a gain of 14.50% in the 15% sample and 10.03% in the 30% sample for the first level and a gain of 23.73% in the 15% sample and 18.78% in the 30% sample for the second level.

Class	Majority Best Evidence		Linear Fusion		GA		Majority GP		Content-based SVM		Combination-based SVM	
	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)	(15%)	(30%)
A.0	47.43	53.50	46.56	53.52	46.27	52.89	48.55	55.50	57.41	59.33	57.43	57.43
A.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	11.11	10.52	0.00	11.11
B.1	36.84	42.55	36.11	44.44	39.44	40.91	44.19	50.85	21.62	27.78	21.33	38.46
B.3	28.17	40.00	26.47	46.67	26.47	42.42	30.00	42.86	20.00	38.71	26.47	39.02
B.5	26.97	24.56	19.07	14.04	18.71	13.11	25.64	30.30	16.13	12.12	20.69	19.45
B.6	45.91	53.07	42.16	49.33	40.73	45.82	45.09	57.59	26.84	38.89	42.54	46.62
B.7	57.31	59.75	52.86	53.88	50.21	49.46	59.56	62.22	45.29	58.89	51.18	55.35
B.8	5.41	0.00	0.00	0.00	0.00	0.00	5.56	9.52	4.16	0.00	0.00	6.90
C.0	0.00	0.00	0.00	9.52	0.00	19.05	0.00	11.76	0.00	0.00	0.00	11.11
C.1	28.57	23.81	19.28	18.60	24.39	19.05	24.24	21.74	14.71	8.89	16.92	20.93
C.2	71.43	74.57	68.48	70.22	66.52	68.27	73.62	77.11	69.02	71.53	70.70	71.74
C.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.25
D.1	10.34	6.90	11.76	12.12	8.20	12.12	12.35	15.79	14.63	26.09	13.74	17.91
D.2	57.95	61.43	57.32	54.95	54.60	57.65	63.58	65.19	55.08	57.41	56.90	60.72
D.3	63.90	66.76	63.20	65.53	61.73	64.37	70.72	69.33	59.35	61.05	64.20	65.48
D.4	66.38	68.75	62.65	65.07	61.07	63.51	68.65	71.65	59.85	61.21	61.11	64.25
E.x	17.14	53.33	25.64	38.89	16.22	43.75	35.00	47.06	7.14	10.52	13.12	10.26
F.1	34.89	44.76	29.25	38.46	26.47	35.84	33.73	40.99	24.78	35.82	27.35	40.25
F.2	21.36	37.74	15.05	28.46	16.36	29.96	26.01	45.99	10.17	21.88	18.84	31.19
F.3	5.26	0.00	0.00	11.11	0.00	11.11	8.51	17.39	0.00	0.00	0.00	0.00
F.4	35.94	46.38	29.27	46.58	30.40	44.74	30.88	50.63	37.33	39.44	28.77	43.18
G.1	67.57	74.06	67.88	72.33	65.76	69.21	73.54	75.59	62.33	67.96	66.38	69.12
G.2	22.03	46.32	20.47	42.59	19.05	43.56	23.18	41.58	23.12	27.69	21.10	33.59
G.3	40.00	49.38	30.87	34.67	31.79	34.57	31.37	46.34	45.98	33.34	35.18	34.48
G.4	23.14	34.48	16.67	31.82	13.79	28.85	17.39	37.21	13.43	36.36	21.66	34.41
H.1	10.53	0.00	9.52	0.00	6.45	0.00	13.70	5.41	6.78	5.72	4.21	4.88
H.2	66.24	78.63	66.21	71.11	68.06	70.85	78.40	81.26	76.62	76.48	75.18	75.81
H.3	59.81	63.16	58.07	59.83	55.43	57.14	63.89	67.92	51.26	55.96	59.11	61.84
H.4	15.69	7.14	24.24	18.75	26.67	22.86	31.43	23.53	5.27	0.00	9.68	8.89
H.5	49.61	57.82	41.33	48.18	42.08	49.85	54.79	61.30	45.78	51.45	41.91	49.59
I.1	39.18	46.15	28.32	39.29	28.57	40.68	43.37	49.12	29.63	19.51	23.64	38.81
I.2	61.93	70.51	55.91	64.60	54.84	64.31	65.19	73.06	54.59	56.43	52.27	62.27
I.3	74.74	75.02	68.92	69.24	63.79	67.51	76.20	77.10	59.83	64.73	58.75	63.76
I.6	74.73	76.41	74.34	71.47	72.44	70.71	70.73	79.63	72.62	73.96	75.67	74.86
I.7	16.67	14.29	21.05	23.08	21.28	14.29	32.65	32.26	14.29	8.00	18.19	23.53
J.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.40	0.00
J.3	36.36	37.04	38.89	40.00	44.44	29.63	56.41	48.28	18.19	18.19	13.34	4.17
J.5	14.04	9.38	23.53	9.52	10.67	9.84	22.47	19.18	22.59	11.77	10.42	18.92
J.6	24.56	62.50	29.27	54.05	24.10	45.45	36.62	60.00	16.67	14.81	17.30	36.73
K.1	23.53	0.00	25.00	11.76	30.30	0.00	24.24	12.50	24.00	26.09	10.52	11.11
K.3	76.25	75.88	73.82	74.70	71.49	72.87	76.45	78.42	74.38	75.41	77.64	77.68
K.4	20.25	31.09	8.42	22.42	12.16	28.46	25.51	33.20	17.95	21.82	12.46	32.86
K.6	18.32	25.83	21.13	20.07	21.43	21.62	21.63	29.83	10.76	19.52	23.61	37.58
K.7	19.16	43.10	22.01	39.64	22.43	40.65	22.43	40.68	26.74	51.67	20.82	41.25
Avg F1	34.44	39.46	32.52	37.28	31.70	36.29	37.90	43.57	30.17	33.11	30.63	36.68

Table 6: Comparison between Majority Voting using the best evidence per class in isolation, Linear Fusion, GA, combined majority GP, content-based and combination-based SVM for the second level.

The performance of content-based SVM is also worse than that of the majority GP, which suggests that we have a better classification method.

Finally, when we compare majority GP against both linear fusion and GA, we see that GP is able to discover better similarity functions.

We did a pair-wise t-test comparing GP with all other columns in Tables 5 and 6, respectively. Majority GP is statistically significantly different from all the others, with $p<0.05$.

5. RELATED WORK

In the World Wide Web environment, several works have successfully used link information to improve classification performance. Different information about links, such as anchor text describing the links, text from the paragraphs surrounding the links, and terms extracted from linked documents, has been used to classify documents. For example, Furnkranz et al. [15], Glover et al. [16], and Sun et al. [33] show that anchor text and the paragraphs and headlines that surround the links helped improve the classification result. Similarly, Yang et al. [35] claimed that the use of terms from linked documents works better when neighboring documents are all in the same class.

Other researchers applied learning algorithms to handle both the text components of the Web pages and the links between them. For example, Joachims et al. [20] studied the combination of support vector machine kernel functions representing co-citation and content information. Cohn et al. [7] show that a combination of link-based and content-based probabilistic methods improved classification performance. Fisher and Everson [14] extended this work by showing that link information is useful when the document collection has a sufficiently high density in the linkage matrix and the links are of high quality.

Chakrabarti et al. [3] estimate the category of test documents by studying the known classes of neighboring training documents. Oh et al. [31] improved on this work by using a filtering process to further refine the set of linked documents to be used. Calado et al. [2] proposed a Bayesian network model to combine the output of a content-based

classifier and the information provided by the documents' link structure.

6. CONCLUSION

In this paper, we considered the problem of classification in the context of document collections where textual content is scarce and imprecise citation information exists. A framework for tackling this problem based on Genetic Programming has been proposed and tested. Our experimental results on two different sets of documents from each level of the ACM Computing Classification System have demonstrated that the GP framework can be used to discover better similarity functions that, when applied to a *kNN* algorithm, can produce better classifiers than ones using individual evidence in isolation. Our experiments also showed that the framework achieved results better than both traditional content-based and combination-based SVM classifiers. Comparison between the GP framework and linear fusion as well as GA also showed that GP has the ability to discover better similarity functions.

Future work will include an extensive and comprehensive analysis of the reasons why GP-based fusion works and outperforms other similar techniques. We want to explore parallel computation to address the scalability issue. We also want to test this framework with different document collections (e.g., the Web) to assess its applicability. Besides that, we want to improve our current evidence, for example, using better methods for citation matching, by trying to fix some OCR errors and using different matching strategies. Finally, new terminals (features) representing additional evidence may be explored; one example is further evidence such as anchor/citation text or patterns of authorship.

7. ACKNOWLEDGEMENTS

This work was funded in part by NSF through grants DUE-0136690, DUE-0121679, and IIS-0086227. Additional support was provided by AOL, CNPq, MCT/FCT (grant SFRH/BD/4662/2001), and Fucapi Technology Foundation.

8. REFERENCES

[1] R. Amsler. Application of citation-based automatic classification. Technical report, The University of Texas at Austin, Linguistics Research Center, Austin, TX, 1972.

[2] P. Calado, M. Cristo, E. S. de Moura, N. Ziviani, B. A. Ribeiro-Neto, and M. A. Gonçalves. Combining link-based and content-based methods for Web document classification. In *Proc. of CIKM-03*, pages 394–401, New Orleans, US, 2003.

[3] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of SIGMOD*, pages 307–318, Seattle, 1998.

[4] S. M. Cheang, K. H. Lee, and K. S. Leung. Data classification using genetic parallel programming. In *GECCO-03*, volume 2724 of *LNCS*, pages 1918–1919, Chicago, 2003.

[5] CITIDEL. Computing and Information Technology Interactive Digital Educational Library, www.citidel.org, 2004.

[6] C. Clack, J. Farringdon, P. Lidwell, and T. Yu. Autonomous document classification for business. In *AGENTS-97*, pages 201–208, 1997.

[7] D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS 13*, pages 430–436. MIT Press, 2001.

[8] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks*, 31(11–16):1467–1479, 1999. Also in Proceedings of the 8th International World Wide Web Conference.

[9] M. D. del Castillo and J. I. Serrano. A multistrategy approach for digital text categorization from imbalanced documents. *SIGKDD*, 6(1):70–79, 2004.

[10] J. Eggermont, J. N. Kok, and W. A. Kosters. Genetic programming for data classification: Refining the search space. In *Proc. of BNAIC-03*, pages 123–130, Nijmegen, 2003.

[11] W. Fan, E. A. Fox, P. Pathak, and H. Wu. The effects of fitness functions on genetic programming-based ranking discovery for web search. *JASIST*, 55(7):628–636, 2004.

[12] W. Fan, M. D. Gordon, and P. Pathak. Discovery of context-specific ranking functions for effective information retrieval using genetic programming. *TKDE-04*, 16(4):523–527, 2004.

[13] W. Fan, M. D. Gordon, P. Pathak, W. Xi, and E. A. Fox. Ranking function optimization for effective web search by genetic programming: An empirical study. In *Proc. of HICSS-04*, pages 105–112, Hawaii, 2004.

[14] M. Fisher and R. Everson. When are links useful? Experiments in text classification. In *Proc. of ECIR-03*, pages 41–56, 2003.

[15] J. Furnkranz. Exploiting structural information for text classification on the WWW. In *IDA-99*, pages 487–498, 1999.

[16] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake. Using Web structure for classifying and describing Web pages. In *Proc. of WWW-02*, 2002.

[17] M. Gordon. Probabilistic and genetic algorithms for document retrieval. *CACM*, 31(10):1208–1218, 1988.

[18] J. H. Holland. *Adaption in Natural and Artificial Systems.* MIT Press, Cambridge, MA, 1992.

[19] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of ECML-98*, pages 137–142, Chemnitz, Germany, 1998.

[20] T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proc. of ICML-01*, pages 250–257, Williams College, US, 2001.

[21] N. Kampanya, R. Shen, S. Kim, C. North, and E. A. Fox. CitiViz: A visual user interface to the CITIDEL system. In *Proc. of ECDL-04*, pages 122–133, Bath, UK, 2004.

[22] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25, 1963.

[23] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *IEEE TEC-00*, 4(3):242–258, 2000.

[24] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Genetic programming based pattern classification with feature space partitioning. *Information Sciences*, 131(1-4):65–86, 2001.

[25] J. R. Koza. *Genetic programming: On the programming of computers by natural selection.* MIT Press, Cambridge, 1992.

[26] A. Krowne and E. A. Fox. An architecture for multischeming in digital libraries. In *Proc. of ICADL-03*, pages 563–577, Kuala Lumpur, Malaysia, 2003.

[27] W. B. Langdon. *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!* Kluwer, Boston, 1998.

[28] W. B. Langdon and R. Poli. *Foundations of Genetic Programming.* Springer-Verlag, New York, 2002.

[29] S. Lawrence, C. L. Giles, and K. Bollacker. "Digital Libraries and Autonomous Citation Indexing". *IEEE Computer*, 32(6):67–71, 1999.

[30] T. M. Mitchell. *Machine learning.* McGraw Hill, New York, US, 1996.

[31] H.-J. Oh, S. H. Myaeng, and M.-H. Lee. A practical hypertext catergorization method using links and incrementally available class information. In *Proc. of SIGIR*, pages 264–271, 2000.

[32] H. G. Small. Co-citation in the scientific literature: A new measure of relationship between two documents. *JASIS*, 24(4):265–269, 1973.

[33] A. Sun, E.-P. Lim, and W.-K. Ng. Web classification using support vector machine. In *Proc. of WIDM-02*, pages 96–99, 2002.

[34] Y. Yang. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In *Proc. of SIGIR-94*, pages 13–22, Dublin, IE, 1994.

[35] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *JIIS*, 18(2-3):219–241, 2002.

[36] B. Zhang, M. A. Gonçalves, W. Fan, Y. Chen, E. A. Fox, P. Calado, and M. Cristo. Intelligent fusion of structural and citation-based evidence for text classification. Technical Report TR-04-16, Computer Science, Virginia Tech, 2004.

[37] B. Zhang, M. A. Gonçalves, and E. A. Fox. An OAI-based filtering service for CITIDEL from NDLTD. In *Proc. of ICADL-03*, pages 590–601, Kuala Lumpur, Malaysia, 2003.