Using CBR techniques to detect plagiarism in computing assignments

Pádraig Cunningham

Department of Computer Science, Trinity College Dublin

Alexander N. Mikoyan

Dept. of Mathematics and Theoretical Mechanics, Moscow State University

Abstract. The problems of case retrieval in CBR and plagiarism detection have in common a need to detect close but not exact matches between exemplars. In this paper we describe a plagiarism detection system that has been inspired by ideas from CBR research. In particular this system can detect similarities between programs without performing exhaustive comparisons on all exemplars. Our analysis of similarity in this well controlled domain offers some insights into the kinds of profiles that can be used in similarity assessment in general. We argue that the choice of a perspicuous profile is crucial to any classification task and determining the best predictive features may require significant analysis of the problem domain.

1 Introduction

The problem of detecting plagiarism in computing assignments depends on being able to identify similar programs in large populations. This emphasis on similarity, on identifying close matches, is reminiscent of the problem of case retrieval in CBR. In this paper we will concentrate on the application of CBR techniques in Cogger[*], a system for detecting plagiarism. We will discuss what this novel domain informs us about retrieval in CBR and about the automatic assessment of similarity in general. Our considerations on similarity in this well controlled domain offer some insights into the alternatives of statistical and knowledge based classification.

Since the idea of similarity can be considered along several dimensions it is often difficult for humans to agree on when cases, or programming assignments, are similar. In this research the programs under consideration have complicated structure and programs are considered to be similar if their function call structure is similar. This involves the determination of the similarity of function call trees; the mechanisms we use are described in Appendix I.

Before examining the problem of plagiarism for a CBR perspective we will introduce some theoretical issues in CBR that are relevant. In section 3 we discuss similarity in general and in section 4 we consider the issue of problem representation that must be considered before any similarity can be determined. We believe that a basic tenet of the majority of CBR research is that similar cases can be retrieved from the case-base inexpensively; in section 5 we consider what kinds of representations are required to support this.

2 Theoretical Issues

Currently in AI there is a view that knowledge representation is unsuccessful and knowledge acquisition is fraught with problems. Consequently there is a move towards an AI paradigm that avoids these issues. This new AI is based on statistics and weights rather than symbolic knowledge representation [1]. The current popularity of connectionism is evidence of this. Closer to CBR, Memory Based Reasoning (MBR) is an approach to AI that wishes to avoid knowledge acquisition and domain modelling [2]. The great attraction of neural networks and MBR is the contention that expert performance can be achieved without knowledge level analysis of the problem domain. This is in sharp contrast with the conventional view in AI; the view that "In the knowledge lies the power" and the knowledge must be represented explicitly.

CBR is a methodology that can serve both of these paradigms. Case-Based Reasoning systems can be **information theoretic** or **knowledge-based**. CBR systems for simple tasks like diagnosis or property valuation can be set up with little analysis of the problem domain. At the other end of the spectrum systems for more complex tasks like design require a complex domain model in order to process retrieved cases.

The main theme of this paper is the implications that these issues have on determining similarity in case retrieval. Is it possible to establish the similarity of two cases in a system that does not have a strong domain model? How far can we go with shallow index features in case retrieval? To this end we will analyse similarity in the context of detecting plagiarism in computing assignments. This is not really a CBR problem but we will argue that the issues of similarity are the same nonetheless.

---

[*] "Cogging" is an anglo-irish slang word for copying homework or other exercises.

3 Similarity in CBR and in Plagiarism Detection

The standard approach to the problem of detecting plagiarism is to produce a profile reflecting the use of keywords and identifiers and to use this signature to produce a table describing the 'distance' between different programs (see [3][4][5] for instance). Research in CBR and machine learning has developed methods for assessing similarity in large populations that are more sophisticated than this so in this paper we apply CBR insights on similarity to the plagiarism detection problem.

In CBR the objective in noticing similarity is to allow for reuse of old solutions in new situations. In plagiarism the object is to identify similarity that betrays a common origin. In CBR similarity may be based on surface features or on features that are more abstract and structural. In plagiarism detection attempts will have been made to conceal superficial similarity and detection must be able to identify systematic or structural similarities.

3.1 A Brief overview of CBR

In attempting to apply CBR techniques in this domain our understanding of the stages in CBR are as follows:-

- Case Representation
- Case Indexing/Retrieval
- Mapping
- Adaptation

A fundamental idea in much of CBR research is that the identification of the best matching case from the case-base should be a two stage process [6]. **Base filtering** is the first stage where a set of candidate cases are selected from an indexed case-base. The case-base will often be organised as a discrimination net to facilitate this. The second stage (**Case Selection**) will select a case from this candidate set based on a more detailed comparison of the cases. A mapping between the base and target cases may also be produced at this stage. Such a selection that does not involve an exhaustive search of the case-base is a novel idea in plagiarism detection.

For plagiarism detection we will concentrate first on producing a representation of the programs for mapping, before analysing the mapping process itself. More than anything this exercise in plagiarism detection highlighted the importance of this parameterisation process.

4 Representation

The first phase in the development of a CBR system involves deciding on a representation of the cases in the system (Fig. 1). This phase is crucial because the perspicuousness of the representation greatly influences the success of the subsequent phases. It is important to note that the contribution of neural networks in tasks of this type is in the classification process that operates on the chosen representation (see the mushroom classification task in [7] for instance. However we are arguing that the crucial phase in this problem solving process is choosing the correct representation in the first place. Indeed given a good predictive representation it may be possible to classify inputs using traditional cluster analysis techniques.

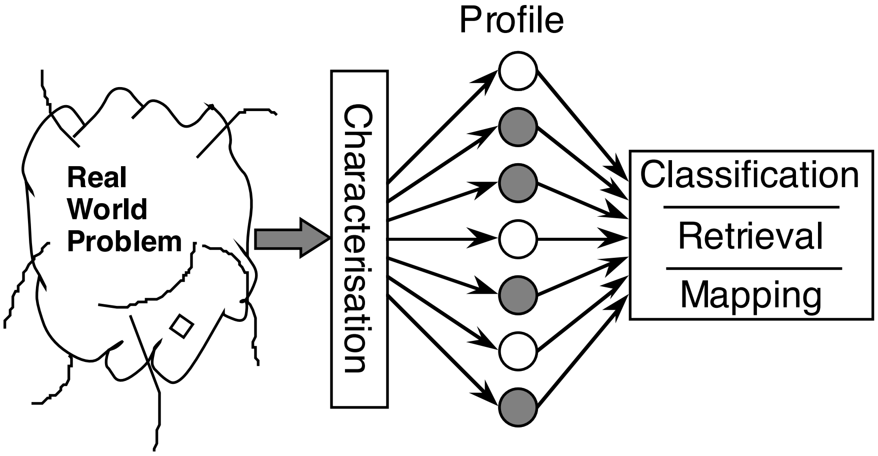

Fig.1. Characterisation involves producing a profile that represents real world problem for use in classification, etc.

From a problem solving perspective it is useful to characterise indexing features along a continuum from shallow to deep (Fig. 2). Shallow features are the obvious surface features of a case and can be determined without much analysis. Deep features are more predictive in the context of the problem in hand but require more analysis to determine. It should be clear that when two cases are very similar they will share surface features. However, when the similarity is more abstract shallow indexes may be different and the similarity may only be indicated in more abstract features. We believe that the plagiarism detection problem that we will describe offers some useful insights in this regard.

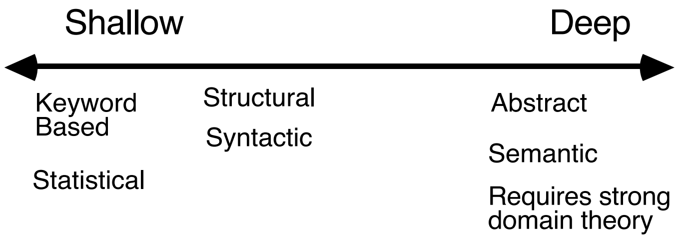

Fig.2. Indexing features can be shallow or deep depending on their semantic content.

5 Base Filtering

What we require form base filtering is a computationally inexpensive means of selecting a small set of candidates from the case-base that are likely to be similar to our target case. This is essentially a classification problem and as such there are several different approaches that can be adopted. One important criteria in categorising techniques is whether classification involves an exhaustive search of the case-base. Nearest neighbour techniques are of this type and involve comparing the target case with each base case in turn using a global distance metric. In these cases retrieval is O(n) where n is the number of cases in the case-base. More promising case retrieval techniques structure the case-base as a decision tree and retrieval is O(log (n)) since it does not involve visiting every case in the case-base.*

The most common means of supporting base filtering in CBR is to organise the case-base as a decision tree that will support rough remindings without the need for exhaustive comparisons. Two approaches were considered for Cogger:-

- **Information theoretic**: Using Gennari's Classit algorithm [8]. This method takes a case represented as a vector of numeric attributes and can incrementally locate the case in a classification hierarchy. In case retrieval this classification can produce the candidate set required from base filtering. The main advantage of Classit is that it is incremental, unlike other cluster analysis techniques in statistics.

- **Knowledge based**: Discrimination networks (D-nets). The indexing features are ordered according to importance and the case-base is structured as a taxonomy based on this feature ordering. Cases are classified by locating them in this taxonomy.

D-nets have the advantage that, with redundancy, they can support different types of remindings. However, the use of D-nets commits the user to a knowledge level evaluation of the problem domain and to an ordering of the indices to reflect their priorities. For this reason Gennari's incremental clustering algorithm was used to perform the base filtering in Cogger.

Experiments with this base filtering indicate that no cases are slipping though the net. In no situation has the base filtering failed to capture known similar cases in the candidate set.

5.1 Experiments in Cogger

One of the very simplest profiles that can be used as the basis for similarity assessment in plagiarism detection is an ordered frequency count of identifiers in the programs. Matching is done based on comparing ranked frequency counts. Table 1 shows an example of two such profiles. This profile is at the very shallow end of the continuum shown in Figure 2. It is easy to construct without any knowledge of what is going on in the program. This profile is adequate for spotting blatant plagiarisms as is the case here. However, a few simple changes to the programs will fool any similarity metric based on this profile.

---

* There are also connectionist techniques for classification and similarity that have not been considered here. In particular spreading activation and constraint satisfaction have been used in ARCS [9] and SAARCS [10]. These are localist connectionist systems and are different to other approaches mentioned in this paper.

Table 1. Frequency counts of identifiers in two programs

Program stu1		Program stu3	
19	close	20	close
11	pipea	10	pipea
10	pipeb	10	pipeb
4	hold	4	num
4	dup	4	dup
3	wait	3	wait
3	fork	3	fork
3	execlp	3	execlp
2	squasherpd	2	squasherpd
2	readerpd	2	readerpd
2	pipe	2	formatterpd
2	formatterpd		

A rudimentary understanding of the programming language suggests a improved profile based on counts of reserved keywords only. This profile better reflects the actual structure of the program since keywords like `do` and `if` indicate specific control structures. Similarity metrics based on this profile can spot less obvious similarities.

Table 2. Profiles based on counts of reserved identifiers.

Program sta05		Program sta09	
4	char	2	char
		1	do
38	else	41	else
1	float	1	float
19	for	14	for
52	if	56	if
4	int	9	int
21	return	22	return
1	sizeof	1	sizeof
2	struct	2	struct
22	void	27	void
9	while	8	while

In the terms introduced in Fig. 2 this is an improved surface profile or a rudimentary structural profile. The profile that we actually used in Cogger is a further improvement on this. Some of the less predictive keywords have been dropped and some structural parameters of the programs have been included. For instance; the first parameter (`top width`) is the number of function calls from the top level of the program, the `depth` is the maximum depth of function calls.

Table 3. Profiles based on structural paramters and counts of reserved identifiers.

Program sta05		Program sta09	
39	top width	9	top width
6	depth	6	depth
39	max width	31	max width
13	user defined	14	user defined
13	system def.	12	system def.
0	recursive	1	recursive
0	do	1	do
19	for	14	for
52	if	56	if
9	while	8	while
21	return	22	return
38	else	41	else
0	case	0	case
0	switch	0	switch

6 Evaluation and Conclusions

We have tested these profiles on three data sets containing from 6 to 35 program profiles. When programs are very similar it shows up in all profiles as would be expected. When the plagiarism is more subtle and surface features have been altered the similarity is not evident in the surface profile but shows up in the structural profile (see Table 3). Determining the appropriate features for this profile required some analysis. So we conclude that for more difficult classification tasks surface profiles are not adequate and the more abstract profiles are more expensive to set up.

From the perspective of the plagiarism detection task, the main novelty of this system is that it operates without doing exhaustive comparisons. Cogger performs similarity assessment as a two stage process; the first stage uses the Classit algorithm [8] to produce the candidate set of cases. The second stage performs expensive comparisons of the program structures to produce a metric of similarity (see Appendix I for details). The main conclusions from this exercise are as follows:-

- Effective profiling is crucial: no amount of cleverness in matching and retrieval can compensate for poor case representation. Figure 1 depicts the characterisation process that produces the regular case representation for classification etc. Settling on the best predictive features was a non-trivial task in Cogger and would be expected to be more difficult in less formal domains.

- Cogger performs very well at identifying similarities in programs; however, it must be acknowledged that using the function tree structure as the basis of the mapping process is crucial to this success.

References

1. Stanfill C., Waltz D., "Statistical Methods, Artificial Intelligence, and Information Retrieval", in