# Competence-Driven Active Learning

Charles McCarthy
106380221
cmmc1@student.cs.ucc.ie
Supervisor: Dr. Derek Bridge

April 2012

**Abstract**

In this report, we:

- Present an overview of the active learning domain.
- Investigate the notions of competence models applied to the area of selection strategies in pool-based active learning, attempting to expand on the work of Delany [25].
- Develop and experiment on new selection strategies based on competence models, and present several, which also incorporate the idea of density, which perform favourably against more traditional strategies.
- Present the platform which we developed for the experimentation, which has the useful capability of operating in a distributed-computing fashion.

# Declaration of Originality

In signing this declaration, you are confirming, in writing, that the submitted work is your own original work, unless clearly indicated otherwise, in which case, it has been fully and properly acknowledged.

I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way towards an educational award;
- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Name : Charles McCarthy

Signature: _____

Date : April 5th 2012

# Acknowledgements

While much of the work carried out was based on external literature, a great deal of advice and direction was gained from the weekly meetings with my project supervisor, Dr. Derek Bridge.

The impact of casual conversations with classmates Pádraig Ó'Dúinn, Barry Hurley and Kevin Leo should also be acknowledged. Pádraig, in his research, encountered similar issues to myself on occasion, and would often share his insights into how he got around them. Barry was useful in offering LaTeX help, and was sometimes consulted while I was learning it. Kevin had a great deal of Python and general *nix experience, and offered advice and assistance whenever particularly frustrating issues were encountered (which would usually be prompted by me ranting about them during our lunchtime card games).

Barry and Pádraig also assisted in reviewing this document, checking it for ambiguities, grammar mistakes, etc.

# Contents

# Chapter 1

# Introduction

## 1.1 Classification

Many overlooked everyday tasks involve classification of one form or another.

- Bouncers at pubs / clubs try to determine if patrons should be allowed entry or not. This represents binary classification: classification where only two labels are involved.
- When we go to our cupboard and find bread gone past its expiration date, we might classify it as 'fine', 'iffy' or 'asking for trouble'. This represents multi-label classification: classification where more than two labels are involved.
- Spam / ham email classification - because general discussions of classification are not complete without mentioning this domain.

*Classification* involves assigning some form of a label to an object (be it physical or conceptual).

## 1.2 Building Classifiers

Taking the potentially stale bread example, we might wish to somehow automate the process of classifying the bread. Doing this would likely involve us defining some characteristics related to the bread for which we might be able to base our decision, for example:

- Number of days past expiration date
- Type of bread (soda / white sliced / brown sliced)
- Number of mould spots
- Maximum mould spot diameter

Measurable characteristics of a class of objects, in more technical terms, are called *attributes*.

From these attributes, we might come up with rules based on our personal intuition to determine the label:

- If number of days past expiration $\leq 0$: Fine
- Otherwise, if the number of mould spots $\leq 2$ and max mould diameter $\leq$ 1cm: Fine
- Otherwise, if it's soda bread: Iffy
- Otherwise: Asking for trouble

Manually creating classifiers can be a tedious process however, and while rule based systems are sometimes genuinely valid in certain domains like medical diagnosis, having a computer program capable of building a classifier is often very useful.

## 1.3 Case-Based Classification using KNN

### 1.3.1 Overview

One method of building a classifier is through the use of case-based reasoning systems. In this model, we have a relatively large set of cases (data instances within the domain, each characterized by attributes and a label). When a new unlabelled case is presented to the system, the system consults, in some fashion, its collection of labelled cases (collectively termed the *case-base*) to try to infer the presented case's label.

The consulting bit involves having a classifier trained from the case-base, and using the classifier to predict the new instance's label. The use of k-nearest neighbours classification fits the case-based reasoning concept quite well. There is no expense involved in the training phase since the classifier simply runs directly off of the case-base.

In k-nearest neighbours, when a new unseen case is presented to the classifier, the classifier finds cases in the case-base that are similar to the unseen case, and uses these cases (along with knowledge of their labels) to predict a label for the unseen case.

### 1.3.2 Similarity/Distance

The notion of *similarity* or *distance* between pairs of cases is necessary for KNN. As one would expect, the similarity between two cases is a numerical figure which represents how similar the cases are to one another. Inversely for distance.

### 1.3.3 Nearest Neighbours

KNN operates by retrieving a fixed quantity 'k' of the new case's nearest neighbours within the case-base. 'Nearest' is defined based on the distance metric used.

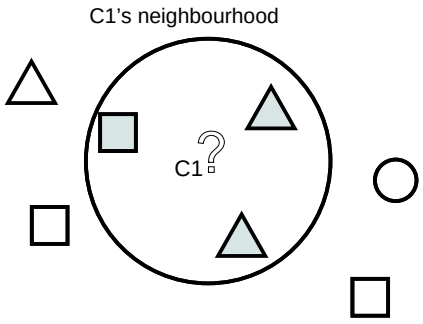



Figure 1.1: Example of a new unlabelled case $C1$ added to a case-base, placed at distances relative to the existing cases in the case-base. At $k = 3$, and KNN used, the shaded cases are retrieved as the 3-nearest neighbours. Assuming majority voting, it is classified a triangle.

In the example presented in Figure 1.1, the shaded cases are retrieved as the nearest neighbours of the new unlabelled case $C1$.

### 1.3.4 Performing the Classification

How the unlabelled case gets classified based on the retrieved nearest-neighbours depends on the voting method used, for example:

- In majority voting, the most frequent label of the nearest neighbours is used. In the example, two vote 'triangle', and one votes 'square'. 'Triangle' is the more frequent label of the retrieved neighbours, so it is classified 'triangle'.
- In distance weighted voting, votes are weighted based on the distance between the source of the label and new unlabelled case. Closer neighbours thus get a larger proportion of the vote than far-off neighbours.

Having k greater than one has the benefit of trying to minimize the effect of noise in the dataset. Using distance weighted voting as opposed to majority voting also shares this characteristic.

## 1.4 Getting Labelled Training Data

To perform classification using KNN (or any other type of machine learning based classification), there is an unavoidable need for labelled training data to initially train the system. Inevitably, how one goes about generating that labelled training data becomes a concern.

In general, the entity which the system may consult if it wishes to determine the true label for a case is termed the *oracle*.

While often, the oracle is a human expert familiar with the domain in question, the oracle may be any type of entity for which there is some expense[1] in using to determine a label.

There is a limit to how much data we can ask the oracle to label. If a human expert is needed, this human factor in itself limits the quantity of data which can be labelled. The financial cost might be significant, and the availability of such an expert can also be a limiting factor. As a result of all this, the question of which data is worthwhile labelling comes up.

## 1.5 Active Learning

The key concept of active learning is that a machine learning program has a say in its own learning process, by being capable of choosing the data from which it learns.

Not all data is of equal benefit in labelling if it's at the expense of not labelling some other data. We may already have sufficient data for one area of the domain, and be relatively confident in our classifications for that area, but might have very little for some other part of the domain. Given that we can only label so much, it would be preferable, for example, to explore the more unknown part of the domain.

The term *selection strategy* is used to denote the algorithmic approach taken to determine which unlabelled cases to query against the oracle for their true labels, and subsequently put in the case-base with their labels. The *stopping condition* of the system is the point at which the system must stop consulting the oracle.

[1] Here, we use the word 'expense' in its broadest possible meaning. It may be financially expensive, computationally expensive, or simply take an exceedingly long time to run. In any event, 'expense' refers to some undesirable operation for which we want to minimize the number of calls.

## 1.6 Active Learning Application Domains

There are many real-world domains in which a large quantity of unlabelled data is available, but labelling all of that data is not feasible.

Often the labelling process is in itself exceedingly slow compared to either the amount of unlabelled data available, or the rate at which new data is being collected.

### 1.6.1 Recommendation Engines

In recommendation engines, the user can be thought of as the oracle. It is from user feedback (either implicit or explicit) that the recommendation engine can know if an item was relevant to the user or not. It is likely that there is a vast library from which a recommendation can come, but the quantity of user feedback may be limited.

What is sometimes done in these systems is that the engine includes some results it is relatively certain will be relevant to the user, and others of which it is uncertain, but would like to gain feedback on from the user. Only comparatively few of these uncertain cases can be given to the user for feedback, as doing otherwise would negatively impact the user's experience.

Choosing which uncertain cases to present to the user to maximize the system's competence-gain is therefore an important problem for the system.

### 1.6.2 Medical Data

Medical data is another area in which active learning is of particular interest and relevance.

Vast quantities of patient data exists, but each patient data instance is unlikely to have results of tests for every possible disease / condition. It is obviously preferable though to detect any illness or condition a person has as soon as possible, so that appropriate measures are taken. It is not feasible however to test each individual for every possible illness.

When building a system to predict illnesses, one would want to minimize the number of physical tests to be carried out on patients for the purpose of training the system. It is in this way that active learning can have a useful role to play.

### 1.6.3 Computer Vision

Computer Vision is another area in which large quantities of data are available, and easy to generate, but manual labelling is incredibly tedious.

### 1.6.4 Text Classification

With the proliferation of the internet, and in particular social media, text classification has an increasing number practical applications. The traditional example is that of email spam analysis, but more and more, sentiment analysis is becoming an active area of interest.

Vast quantities of textual data are often available for the target domain, though initial labelling requires human involvement. Active learning can play a potentially very useful role in the process, helping in maximizing the system gain with the minimum amount of data.

### 1.6.5 Voice Analysis

Speech recognition is one area that can bring immense benefit to users, but for which manual labelling requires expert skills, and is painstakingly slow even with a trained expert's involvement. For such systems, it is highly desirable to reduce the quantity of data that must be manually labelled to bring the system to an acceptable rate.

## 1.7 Types of Active Learning

### 1.7.1 Stream Based Learning

In *Stream Based Learning*, the system does not just select at the outset the cases to request labelling by the oracle. Instead, it is presented with a stream of cases, and for each case, it may either itself guess the classification (if it is relatively sure it is correct), or it may defer the decision to the oracle.

### 1.7.2 Query Synthesis

Although it can cause hassle for a human annotator, another variation is known as *Query Synthesis*, in which the program actually constructs its own unlabelled data instances to present to the oracle (as opposed to the data being actual previously observed).

The program is thus able to actively explore the domain space, as opposed to just waiting for data to try to infer this information.

While unsuitable for many problem areas (in particular text), one could easily see how this model could be useful in the context of scientific experiments. The program may wish to make *guesses* as to what would be useful data to get classified, so that it may further refine its knowledge of the domain.

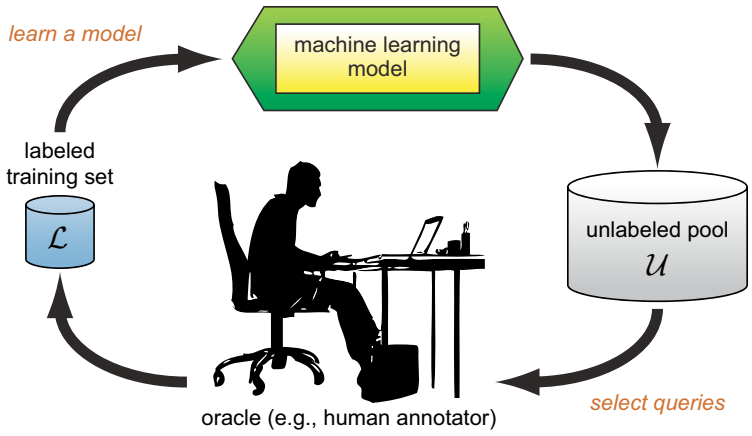
Settles [1] points to a good example involving a "robotic scientist" as described in King et al. [2].

Query Synthesis is not really necessary in traditional pool-based sampling areas - since a large pool of truly observed (and therefore valid) unlabelled examples are present that should be representative of the type and distribution of problems which will be later presented to the system.

### 1.7.3 Pool-Based Sampling

The facet of active learning we will subsequently consider is known as *pool-based sampling* in other literature [1], and is based on the idea that there is a large quantity of unlabelled cases available, but getting labels (which might involve consulting a human expert) is an expensive process, for which it is undesirable to apply to the entire unlabelled set.

The aim of the system is to choose a subset of these unlabelled cases for which to request a label. These are then used as the system's pool of labelled cases which it will consult in future. The system would like to maximize its classification accuracy for future cases, but resort to asking the oracle external to the system for a minimal number of labels.




Pool-Based Active Learning Cycle Diagram found in Settles [1]

## 1.8 Variations / Related Areas

Many variations on the concept of active learning exist. The following is intended to give just a small subset of examples, as found in Settles [1].

### 1.8.1 Case-Base Maintenance

*Case-base maintenance* is concerned with the task of how to maintain a case-base.

Over time, as new cases are added, the case-base size increases. This can have an adverse effect on system performance, thus it can be desirable to have the case-base be as minimal as possible, while maintaining classification accuracy.

Another consideration is that some cases may be adversely affecting the system's classification accuracy. It may be that these cases were initially mislabelled by the oracle, the underlying data was noisy, or it could be that the nature of the domain has changed and that similar cases now actually have a different label. The detection of these cases is therefore an important part to long-running case-based reasoning systems.

Overall, case-base maintenance can be summarised as the process of maintaining a case-base to a desirable level over time. This contrasts to what we will concern ourselves with - how to build a case-base.

### 1.8.2 Seeding in Active Learning

Many selection strategies have the assumption built in that there are already some cases in the case-base from which they may base their judgements. Thus the issue of how to initially seed the case-base crops up.

In our evaluations and experimentation, we don't attempt to initially seed the case-base - instead starting with an empty case-base. The reasoning for this is twofold:

1. The datasets we use are relatively simplistic, which even with random sampling, converge to a relatively high accuracy even after a small number of samples are added. This would make it very difficult to reasonably compare different selection strategies if seeding were also used.

2. Adding seeding would also add another factor for testing, as opposed to primarily testing the variance between different selection strategies.

### 1.8.3 Noisy Oracle

A *Noisy oracle* relates to the oracle being wrong in its classification some of the time.

In our investigation, we do not deal with this variation at all, assuming a perfect oracle.

### 1.8.4 Labelling Cost Variation

In many real-world applications, the cost associated with labelling a case can vary from case to case. Some cases can involve much more time to determine a label for than others. In text classification, articles vary in length and complexity. Similarly in video and audio.

In our investigation, we do not deal with case labelling cost variation, instead assuming a fixed cost.

### 1.8.5 Batch Learning

In practice, it is unlikely that cases would be presented to the oracle one-at-a-time. What is more likely is that batches of cases are presented to the oracle.

In our investigation, we do not perform batch learning - instead presenting cases one at a time to the oracle (equivalent to a batch size of 1).

## 1.9 Report Outline

- Chapter 2 will outline various basic selection strategies as found in existing active learning literature, as well as looking at some more complicated strategies.
- Chapter 3 will give an overview of the platform architecture developed during the course of this project for running and analysing selection strategy experiments.
- Chapter 4 will introduce the notion of *Competence Models*, outlining a particular model developed by a researcher from DIT.
- Chapter 5 will then proceed to present our attempts at expanding the models, and using them in an active learning selection strategy context.
- Chapter 6 will present the results of the experimental analysis carried out on the basic selection strategies, and our competence based strategies.
- Chapter 7 finishes with the conclusions to the project, and potential areas of future work that could follow.

# Chapter 2

# Literature Review

## 2.1 Introduction

Active learning, and case-based reasoning in general, have massive quantities of research backing them. In this chapter, we will try to give an overview of some basic selection strategies in use in active-learning, along with two examples of recent research efforts to the area.

## 2.2 Common Concepts

Certain concepts very frequently occur in the active learning literature, so these will be outlined first.

### 2.2.1 Density

The notion of *density* is extremely common. It simply represents how dense a region is with cases within a large group of cases by means of how close cases are to one another.

Density is usually measured across the entire dataset (i.e. both the unlabelled and labelled pool), as it is of less use to know of dense regions within the labelled dataset when trying to choose cases from the unlabelled data to request a label for[1].

The reasoning is that the entire dataset is hopefully representative of real-world cases, of which new cases will appear with similar ranges and distributions. If one selects from a dense region of the unlabelled dataset, it would be hoped that it would cover a large range of input problems.

Even with the commonality of the density notion, there is quite a bit of variance in the literature around how density is actually measured. Two examples of relatively straightforward measures:

1. **Global Density**: The average similarity from a case to every case in the dataset [3].
2. **Local KNN Based Density**: The average similarity to its k-nearest neighbours in the dataset [4].

Density is also sometimes measured for groups as opposed to on a case-by-case basis [5], and often involves the sum of the pair-wise similarities for the cases in the group.

[1] This, in particular, is true when building the case-base. For case-base maintenance tasks, dense regions within the case-base might indicate redundancy.

### 2.2.2 Diversity

*Diversity* (usually also mentioned with density) is another common measure used. It simply represents how different a case is compared to other cases in a group.

In contrast to density, diversity is usually not measured across the entire dataset, but instead just across the actual case-base[2]. The reasoning for this is that it would be beneficial to select new cases that are diverse from what is already present in the case-base, in hopes that a larger range of the domain may be solved as a result.

A relatively common metric for the diversity of a case is the similarity to that case's nearest neighbour within the group in question.

### 2.2.3 Solves / Classifies / Contributes

When it comes to analysing an individual case, often the notion of the other cases that the individual case can somehow classify comes into question. In the context of KNN classification though, an individual case will not be able to classify another case on its own, instead somehow only possibly contributing to another case's classification. This leaves room for different definitions of what one means when they consider a case capable of solving another case.

Cummins [6] highlights the interesting point that there are subtle but definitely significant differences to what different researchers mean when they use the term solves. Often, the term seems to be glossed over quickly in literature due to its apparent obviousness, but is still often core to the algorithms presented.

## 2.3 Basic Selection Strategies

### 2.3.1 Random Sampling

*Random sampling* is probably the most basic selection strategy. It simply selects cases at random from the unlabelled set. Even with its simplicity, it has the benefit of resulting in a moderately representative sampling of the domain[3] if a sufficient number of instances are chosen.

It is often used as one of a group of baseline strategies against which other strategies are compared.

### 2.3.2 Uncertainty Sampling

*Uncertainty sampling* [7] selects cases that a classifier trained on the case-base is most uncertain about (or said another way - is least certain about). Assuming the classifier can generate a probability for each label when provided with a new unlabelled case, the maximum of these for a case can be said to be the *certainty* with which the classifier can classify that case. Picking the minimum of these gives the least certain instance. Uncertainty sampling can thus be defined as, for each unlabelled case, calculating the probability of the most likely label (i.e. the label with the highest probability), and choosing the case with the smallest value to sample.

[2] Again, this might be different when one is dealing with case-base maintenance.

[3] This is assuming that the unlabelled cases are a representative sampling of the real world domain - which we usually assume that they are.

The rough intuition is that by selecting cases that the classifier is least certain about, you are making the biggest impact on the system by trying to ensure that there won't be new unseen instances that the classifier will be so uncertain of.

The benefit of uncertainty sampling is that it doesn't select redundant cases, since whatever it selects is the one that it is least capable of classifying with its current case-base. The downside is that it is very prone to querying outliers, since it does not consider information such as the density of the region from which the case comes.

Uncertainty sampling, like random sampling, is very commonly used as a baseline when evaluating new algorithms.

### 2.3.3 Uncertainty with Density

As discussed in the previous section, uncertainty sampling has the serious downfall of being prone to querying outliers, since it does not consider the regions surrounding the cases it selects.

A very straightforward attempt at solving this issue is by incorporating density information also, giving an additional weighting to the uncertainty measures based on how dense a region the case in question comes from. Uncertain cases from dense regions thus become more favourable than uncertain cases from very sparse regions (outliers).

One example of combining uncertainty with density can be found in Zhu et al. [4], though it uses an uncertainty measure based on Entropy as opposed to pure uncertainty alone.

### 2.3.4 Margin Sampling

*Margin sampling* tries to account for a shortcoming in pure uncertainty sampling for multi-label classification in that it only considers information about the most probable label, essentially throwing away the information about the other labels.

Margin sampling looks at the difference between the probability of the most probable label, and the second most probable label. The intuition is that if there is a large difference between the two, the classifier must be relatively certain. On the other hand, if there is very little difference, then with relatively little change, the classifier might have picked a different label, and so it really mustn't have been very certain about it.

Margin sampling only makes sense in classification where there are more than two possible label values. In binary classification, Margin sampling gives an identical strategy to uncertainty sampling.

### 2.3.5 Diversity Only

In *diversity only* sampling, the system selects cases that are as far away as possible from everything else in the case-base. This can result in an understandably diverse case-base, but is very prone to querying outliers like uncertainty sampling.

This motivates many authors to try and incorporate density information with diversity. One such attempt is described next.

## 2.4 Rong Hu's Work on EGAL: Exploration Guided Active Learning

### 2.4.1 Introduction

Many selection strategies have firm dependencies on a type of classifier confidence score (what Hu terms *exploitation*), while also incorporating measures such as density and diversity (what Hu terms *exploration*).

Hu, in her work on EGAL [8], experiments with developing an exploration only strategy by combining measures of density and diversity[4].

While some of the formula specifics feel slightly odd, the overall attempt at using an exploration only strategy is pretty interesting.

### 2.4.2 Hu's Density / Diversity Measures

#### Density

Hu defines a case's *density* as the sum of similarities within a case's neighbourhood.

$$density(x_i) = \sum_{x_r \in N_i} sim(x_i, x_r)$$

The neighbourhood $N$ of a case $x_i$ within the entire dataset $D$ she defines as:

$$N(x_i) = \{x_r \in D \mid sim(x_i, x_r) \geq \alpha\}$$

$\alpha = \mu - 0.5 \times \delta$ where $\mu$ is the mean, and $\delta$ is the standard deviation of the similarities. This value for $\alpha$ is somewhat arbitrary, in that it just seemed to work well in Hu's experiments.

Of particular note is that the density measure is calculated across the entire dataset $D$ (i.e. unlabelled and labelled).

#### Diversity

Diversity measures how different a case is from everything already in the case-base (all the labelled cases $L$).

$$diversity(x_i) = \frac{1.0}{max_{x_r \in L} sim(x_i, x_r)}$$

In this formula, similarity is inverted because distance is the inverse of similarity. So diversity is really the distance to the nearest neighbour in the labelled set.

[4] Rong Hu recently completed her PHD Thesis on "Active Learning for Text Classification" in D.I.T. with Dr. Sarah Jane Delaney as one of her supervisors [8]. In her thesis, she covers a broad range of topics, but of particular interest to us is her work on "EGAL", and also on using aggregate confidence measures.

### 2.4.3 The EGAL Strategy

#### Overview

Hu's approach is to build a candidate set of cases from the unlabelled set which one might want to add to the case-base. This candidate set is based roughly on the cases that are furthest away from all the cases in the case-base (i.e. the most diverse cases). From these candidates, one then chooses cases that are from dense regions of the case space.

Hu allows for the balance between the effect of density and diversity to be controlled through a "balancing factor".

#### The EGAL Algorithm

**Input**: An initial labelled set $\mathcal{L}$, an unlabelled pool $\mathcal{U}$ of $n$ examples, a stopping criterion $\mathcal{SC}$, a batch size $b$, a balancing parameter $w$
**Output**: A labelled dataset
Compute the similarity matrix $\mathcal{M}$ of $s(i,k)$ where $x_i, x_k \in \mathcal{L} \cup \mathcal{U}$;
Set $\alpha = \beta = \mu - 0.5 \times \delta$; $\mu$ and $\delta$ being the mean and standard deviation of the similarity matrix $\mathcal{M}$;
Calculate density for all the unlabelled examples $x_i, i \in I_u$
**while** $\mathcal{SC}$ *is not met* **do**
  $\mathcal{CS} = \emptyset$, $Selected = \emptyset$;
  Construct the candidate set $\mathcal{CS}$
  **while** $|\mathcal{CS}| = 0$ **do**
    Update $\beta$ ;
    Update $\mathcal{CS}$ ;
  **end**
  Rank examples in $\mathcal{CS}$ by descending density order;
  **foreach** $t, t = 1 \ldots b$ **do**
    **if** $|\mathcal{CS}| < b$ **then**
      $Selected = Selected \cup \mathcal{CS}$ ;
    **else**
      Select the top $b$ ranked examples from $\mathcal{CS}$ with highest density and add them into $Selected$;
    **end**
  **end**
  Label each example $x_i \in Selected$ ;
  $\mathcal{L} = \mathcal{L} \cup Selected$ , $\mathcal{U} = \mathcal{U}/Selected$ ;
**end**

Hu's Algorithm for the EGAL Selection Strategy

#### Candidate Set

$$CS = \left\{ x_i \in U \mid diversity(x_i) \geq \frac{1}{\beta} \right\}$$

This is presented in a slightly expanded (but equivalent) form in Hu's paper dedicated to EGAL [9]. Really, $\beta$ represents the maximum similarity allowed to its nearest neighbour. This formula forms the set of cases within the unlabelled set $U$ which do not have any other cases within a distance of $\frac{1}{\beta}$.

### Generating the Max Similarity Sets

Firstly, for each item in the unlabelled set $U$, the similarity to the *closest* element in the labelled set is calculated and stored[5].

$$S = \left\{ s_i = \frac{1}{diversity(x_i)} \mid x_i \in U \right\}$$

### Determining $\beta$

The setting of $\beta$ feels somewhat round-about - but roughly speaking, $\beta$ is set to the similarity value that would cause a certain pre-fixed percentage of the top-most diverse elements to be chosen as the candidate set.

Firstly, a proportioning parameter $w$ controls the proportion of the top-most diverse elements to include in the candidate set. From experimentation, Hu finds that 0.25 gives good results - i.e. selecting a quarter of the most diverse elements as the candidate set.

Because, in the algorithm, similarity values are spoken about instead of proportions, we must find the threshold similarity value ($s_w$) that would result in that proportion.

$$S_1 = \{s_i \in S \mid s_i \leq s_w\}$$

$$S_2 = \{s_j \in S \mid s_j > s_w\}$$

$$|S_1| = \lfloor (w \times |S|) \rfloor + 1, 0 \leq w \leq 1$$

$\beta$ is then set to that $s_w$ value.

### Algorithm Outline

1. Initially $\beta$ is simply set to $\alpha$, but is updated according to the formula above each time the candidate set becomes empty.
2. The candidate set is sorted from highest density to lowest density, and items are taken in order until the stopping condition is reached.

### Factor Summaries

- $\beta$ represents the maximum similarity that labelled examples should be from an unlabelled example to be thought of as *diverse*, and thus should be included in the *candidate set*.
- $\alpha$ relates to the distance (really similarity) within which to define the neighbourhood of a case for the purpose of calculating the density (total pairwise similarity) of the case.

  This is represented in terms of similarity - the minimum similarity value cases should have to be within the neighbourhood.

[5] Diversity is the distance to the closest element. The inverse of distance gives similarity. Thus the inverse of diversity is the similarity of an element to its closest element.

**$w$ as a Density/Diversity Balancing factor**

Hu explains that $w$ is a balancing factor between density and diversity [9].

- When $w = 0$, a pure diversity-sampling approach occurs.
- When $w = 1$, a pure density-sampling approach occurs.
- When $w$ is set to values between 0 and 1, a mixture of density and diversity sampling occurs.

### 2.4.4 Experimental Results

In Hu [8], Hu finds that $w = 0.25$ gives the best results for the datasets she used.

Hu also finds that the EGAL strategy with $w = 0.25$ compares favourably with random sampling, and also beats diversity-only and density-only sampling.

For comparison with other active learning strategies, Hu finds that EGAL performs generally better than the other strategies at the initial stages of the active learning process.

### 2.4.5 Comments

Even though $w = 0$ gives a diversity-only approach, and $w = 1$ gives a density-only approach, it is unclear the exact effect of having values in between the 0 to 1 range.

The approach of picking a similarity value to achieve that partitioning also feels unnecessarily complicated.

## 2.5 Rong Hu's Work on Aggregating Confidence Measures

### 2.5.1 Introduction

Before continuing, confidence probably needs to be initially distinguished from competence. *Confidence* relates to how confident the system is regarding the classification of an unseen instance (e.g. the probability of the most likely label as determined by the classifier), whereas *competence* relates to the overall capability of the system. It is confidence measures that are dealt with here.

In her thesis [8], Hu outlines a selection strategy based on using a single confidence measure (essentially, the same as standard uncertainty sampling, but using the confidence measure to determine what the case-base is most uncertain of), but expands on this to present an algorithm which attempts to aggregate multiple confidence measures, to produce an algorithm that performs better than using any one of the confidence measures alone.

### 2.5.2 Model & Measures

In Delany et al. [10], five confidence measures are proposed. Hu chooses the three of these which she determines experimentally to have the least correlation[6].

For the three, the following metrics are needed:

[6]The validity / appropriateness of this approach will not be considered further though - since this particular paper - and the piece of interest to us - is that of aggregating confidence measures. Further descriptions, explanations and justifications of the measures can be found in Delany et al. [10]

- $N_i(t)$ denotes the ith nearest neighbour of example t.
- $NLN_i(t)$ denotes the ith nearest like[7] neighbour to example t.
- $NUN_i(t)$ denotes the ith nearest unlike neighbour to example t.

**Average NUN Index**

$$AvgNUNIndex(t,k) = \frac{\sum_{i=1}^{k} IndexOfNUN_i(t)}{k}$$

where:

- $IndexOfNUN_i(t)$ is the index of the ith nearest unlike neighbour of target example t, the index being the ordinal ranking of the example in the list of NNs

This represents the average index of the first k unlike neighbours in the case's sorted nearest neighbour list. Intuitively, a larger average index is preferable, as this somewhat indicates that there wasn't much uncertainty, when the closer neighbours were voting for the predicted label.

**Similarity Ratio**

$$SimRatio(t,k) = \frac{\sum_{i=1}^{k} Sim(t, NLN_i(t)) + \epsilon}{\sum_{i=1}^{k} Sim(t, NUN_i(t)) + \epsilon}$$

where:

- $Sim(a,b)$ is the similarity between examples $a$ and $b$
- $\epsilon$ is a smoothing value to allow for situations where an example may have no NLNs or NUNs ($\epsilon$ = 0.0001 is used in all of her evaluations). Top and bottom have this value to give a ratio of 1 when there are no NLNs or NUNs.

This represents the ratio of the similarity values between a case's k-nearest like neighbours, and its k-nearest unlike neighbours. Intuitively, a higher ratio is better, as this would indicate that the classifier should be certain about its classification, since the cases with the same label as the predicted label are relatively a lot closer than the ones of a different label.

**Similarity Ratio Within K**

$$SimRatioK(t,k) = \frac{\sum_{i=1}^{k} Sim(t, NN_i(t))\delta_{t,NN_i(t)}}{\epsilon + \sum_{i=1}^{k} Sim(t, NN_i(t))(1 - \delta_{t,NN_i(t)})}$$

where:

- $Sim(a,b)$ is as above,
- $\delta_{a,b}$ is Kronecker's delta where $\delta_{a,b} = 1$ if the label of $a$ is the same as the label of $b$ and 0 otherwise.

[7] *Like* and *unlike* are compared to the case's predicted label by the KNN classifier, since obviously, the actual label of the case is unknown.

- $\epsilon$ is a smoothing value to allow for situations where an example may have no NUNs ($\epsilon = 0.0001$ is used)[8].

This measure is similar to the $SimRatio$ measure, except that instead of considering the k-nearest like neighbours, and the k-nearest unlike neighbours, it just considers the like and unlike neighbours within the first k-nearest neighbours (be they like, or unlike).

### 2.5.3 Combining the Confidence Measures

#### Overview

Firstly, in a machine learning fashion, the threshold values are determined for each measure for each label that indicate confidence for that label. These predicted values are then used in assembling a non-confident set, from which elements are selected in a least-confident-first style. This selection happens in 'batches', after which the algorithm is re-initialized for the next batch. This is repeated until the stopping condition is met.

#### The ACMS Algorithm

**Input**: An initial labelled training set $\mathcal{L}$, an unlabelled pool $\mathcal{U}$, a $k$-NN classifier $\mathcal{C}$ for classes $1 \dots c$, a stopping criterion $\mathcal{SC}$, a batch size $b$, a set of confidence measures $M_i$, $i = 1 \dots n$
**Output**: A labelled dataset
**while** $\mathcal{SC}$ *is not met* **do**
  **foreach** *confidence measure* $M_i, i = 1 \dots n$ **do**
    Identify the threshold: find $thres_{ij}$ and $k_{ij}$, for $j = 1 \dots c$;
  **end**
  $ConfSet = \emptyset$, $NonConfSet = \emptyset$, $Selected = \emptyset$;
  **foreach** *example* $e \in \mathcal{U}$ **do**
    Classify $e$ using the classifier $\mathcal{C}$;
    Calculate $m_i$ using $k_{ij}$ for $i = 1 \dots n$ and $j =$ predicted class of $e$;
    **if** $m_i > thres_{ij}$ *for any* $i = 1 \dots n$ *and* $j =$ *predicted class of* $e$ **then**
      $ConfSet = ConfSet + e$ ;
      Set the confidence score: $conf(e) = max(m_i)$;
    **else**
      $NonConfSet = NonConfSet + e$;
      Set the confidence score: $conf(e) = min(m_i)$;
    **end**
  **end**
  **foreach** $l, l = 1 \dots b$ **do**
    **if** $NonConfSet == \emptyset$ **then**
      $Selected = Selected \cup \{e_l\}$ where
      $conf(e_l) = min(conf(e)), e_l \in ConfSet$;
      $ConfSet = ConfSet / \{e_l\}$;
    **else**
      $Selected = Selected \cup \{e_l\}$ where
      $conf(e_l) = min(conf(e)), e_l \in NonConfSet$;
      $NonConfSet = NonConfSet / \{e_l\}$;
    **end**
  **end**
  Label each $e_l \in Selected$ ;
  $\mathcal{L} = \mathcal{L} \cup Selected$ , $\mathcal{U} = \mathcal{U} / Selected$ ;
**end**

Hu's Algorithm for the Aggregated Confidence Measures Selection Strategy

[8]It's unclear the reason for the inconsistency between the use of $\epsilon$ on the bottom only in $SimRatioK$, but on top and bottom in $SimRatio$.

#### Initializing the Case-base

Before the main algorithm which combines the confidence measures is initiated, the case-base is initially seeded with a deterministic variation on Furthest-First Initialization, as presented in Greene [11].

#### Confidence Measure Threshold Initialization

For **each** measure, the confidence threshold needs to be identified for **each** label.

For each measure, predictions with confidence values higher than the predicted label's threshold are considered confident for that measure, while those with values below are considered non-confident for that measure.

Details surrounding how exactly these thresholds are set can be found in Delany et al. [10], but roughly, values are chosen on a per-dataset basis through experimentation, with a threshold calculated for each measure for each label (i.e. cross product style) that maximizes confidence-accuracy while keeping False Positives to zero. This is done because of the boolean nature on which the measures apply, but in the context of classification where there might be more than two labels.

#### Operation

The algorithm essentially operates on repeatedly building two sets from the unlabelled data - the confident set, and the non-confident set.

The confident set is composed of cases for which any of their confidence measures exceeded the measure's confidence threshold. The *score* associated with each case is the maximum of all the confidence measures for that case.

The non-confident set is composed of cases which do not appear in the confident set, but the *score* associated with each case is the minimum of all the confidence measures for that case.

This min-max combination was chosen as it performed the best in Hu's preliminary experiments when she considered multiple variations[9]. Measures for each case "are normalized using statistical normalization after performing a log transformation to correct those with skewed distributions" [8].

After the confident and non-confident sets are built, selection begins. First elements in the non-confident set are selected, and once it is expended, elements from the confident set are selected. Selection occurs from each of the sets in a lowest-score-first order. A batch size, which is an input to the algorithm, indicates how many should be selected in this *batch*, before restarting the entire process, and beginning the selection of a new batch from the start again.

### 2.5.4 Experiment Results

Hu finds that using the ACMS strategy performs marginally better than using any one of the confidence measures alone. She also finds that the strategy, with Delany's measures, performs better than an uncertainty sampling based selection strategy.

[9] I question the validity of this approach. For discussion, see Section 2.5.5.

### 2.5.5 Comments

I am unconvinced about the validity of the min-max style of aggregating the confidence measures.

The paper and thesis section seem rather unspecific about how the measures are normalized. I'm guessing there are broadly two possibilities :

1. **Measure Threshold Normalization**

   In this possibility, measures would be normalized so that their thresholds would be the same.

   I don't think this was the method used. If the thresholds alone were normalized, the measure ranges' maximum possible values would be different. As a result the maximum measure scores would definitely not be directly comparable.

2. **Measure Range Normalization**

   In this possibility, the range of values over which the measure falls is normalized - giving the same minimum and maximum value possible for each measure. I don't think the measure threshold could then also be normalized, leading to different measures very likely having different thresholds.

   If any measure passes its respective threshold though, the maximum measure's value is chosen as the case's score, but this score may have nothing to do with the measure that actually caused the case to fail to be classed as confident. Scores are subsequently compared with other scores.

   For examle, for a given label,

   - $thres(M1) = 0.8$
   - $thres(M2) = 0.2$

   and for a given case $case_1$ for that label:

   - $m1 = 0.7$
   - $m2 = 0.3$

   Because $m2 > thres(M2)$, the case will be put in the confident set, and as such, the maximum of the measures will be taken: $m1 = 0.7$. Really though the confidence measure that passed the threshold was $m2$ with a value of $0.3$.

   Now, imagine a different case $case_2$, but for the same label as above[10]:

   - $m1 = 0.1$
   - $m2 = 0.6$

   Again, because $m2 > thres(M2)$, the case will be put in the confident set, but the maximum measure will be $m2 = 0.6$.

   $case_1$ (with a score of $0.7$) will be treated as more *confident* than $case_2$ (which has a score of $0.6$), even though $case_2$ was actually more confident than $case_1$ in the measure that actually caused it to be confident ($M2$).

Whichever was the actual methodology, both seem flawed. The underlying fact remains that the different confidence measures aren't directly comparable, and the relatively simplistic manner in which the presented aggregation strategy deals with them - simply taking the maximum of all of them when any

[10]The reason I'm specifying the same label is just to emphasize that the same thresholds apply. If it were a different label, different thresholds may apply - since thresholds are on a label-by-label basis - thus adding unnecessary complication to the example.

one is confident - does not appear to take this into account. Similarly with taking the minimum for the non-confident set.

It's possible that there was some advanced maths way, of which I am unaware, to do the normalization which might hope to account for both range and threshold normalization, but I can't see how it could change the fact that a score is being assigned based on possibly a different confidence measure than the one which actually achieved confidence.

As a result of the above, we decided not to proceed with using Hu's aggregation strategy.

## 2.6 Conclusions

From reviewing a sampling of current active learning literature, it appears that no one to date has used purely competence based selection strategies. It is this gap that we wish to tackle.

Even with the abundance of active learning research, there does not appear to be any single widely used platform for selection strategy research. In the next chapter, we present the research platform we developed for the purpose of our experimentation and investigation into competence based selection strategies.

# Chapter 3

# Platform Architecture

## 3.1 Overview

### 3.1.1 Purpose

The purpose of the system is to allow evaluation of and comparison between different selection strategies in a pool-based active learning context.

It achieves this by executing each of the selection strategies on several datasets from an initially empty case-base. At each selection in the selection strategy, it runs a KNN based classifier on the case-base, evaluating its capability to correctly classify the instances in a test set.

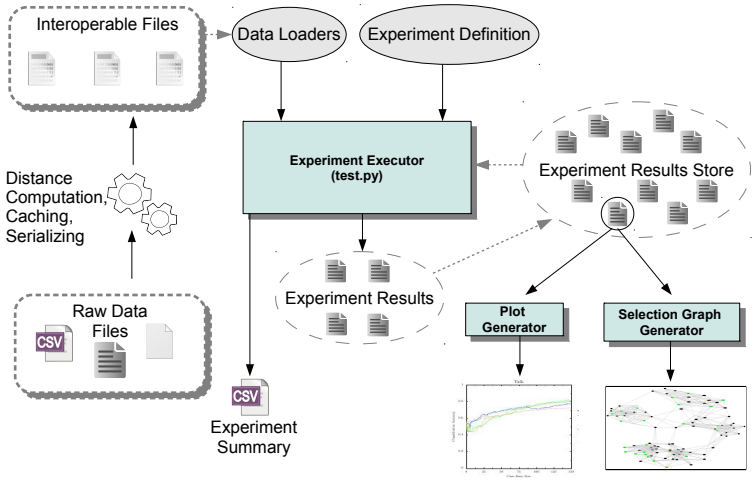
### 3.1.2 Technologies

Weka [12], a Java based machine learning framework, was initially briefly investigated, but due to my familiarity with Python, Orange [13] was used as the core machine learning component. It provided useful facilities to quickly get a basic evaluation platform up and running. As the platform progressed, most Orange dependencies were removed in favour of custom implementations due to the need for greater control, and a memory-management related bug which existed in Orange with regards its Python bindings for its C++ backend. Orange was subsequently just used for data loading, and distance computations.

For plotting, matplotlib [14] was initially used, but due to its very non-pythonic interface, we changed to using PyX [15], which also provided much better support for LaTeX style file formats.

Towards the end of the project, a basic python map-reduce framework 'mincemeat' [16] was integrated into the platform to allow cluster-style operation.

### 3.1.3 High Level Data Flow Diagram




## 3.2 Inputs

### 3.2.1 Data Files

Data files provide a table of instances, each instance consisting of attribute values and a label. Orange provides support for loading many common formats, and also is capable of performing distance computation between cases based on their attributes.

For ease, Orange was used to perform initial data loading, and distance computation between cases. The instances, and their pair-wise distances, were then serialized to a new data file to allow running on platforms (and runtimes) for which Orange wasn't present.

#### Orange Supported

Orange's primary data format is that of tabular, delimited text - such as CSV or TSV. Optionally, markup can be placed in the header rows of these files to help Orange know of the properties of each of the attributes - e.g. if the attribute represents the label for the case, if the attribute is discrete or continuous, etc., but Orange is surprisingly good at figuring most of it out on its own - occasionally just needing slight nudging in the right direction.

Orange also supports some of the formats commonly used in other Machine Learning platforms - e.g. ARFF (as found in Weka) and correctly formatted .names, .data file pairs as sometimes found in the UCI Machine Learning Repository [17][1].

#### Custom Pre-computed Distance Format

The $n \times n$ distance matrix for a dataset can really be stored in a symmetrical matrix[2]. Thus, when storing the distances for all pairs in the file, the space may essentially be halved by storing only one half of the $n \times n$ matrix[3].

[1] It should be noted that the majority of .data/.name pairs are not in the strict format necessary, thus some prior manipulation of the files present there may be needed to load them with Orange.

[2] This is assuming that the distance function is in fact symmetric, i.e. $d(a, b) \equiv d(b, a)$. In our distance function, this property does hold.

[3] Technically, the distances between cases and themselves need not be stored, because by definition the distance between a case and itself will be 0. Though this minor optimization is not performed in my code

| Case ID | Distances | | | | |
|---|---|---|---|---|---|
| 0 | $d(i_0, i_0)$ | | | | |
| 1 | $d(i_1, i_0)$ | $d(i_1, i_1)$ | | | |
| 2 | $d(i_2, i_0)$ | $d(i_2, i_1)$ | $d(i_2, i_2)$ | | |
| ... | ... | ... | ... | ... | |
| j | $d(i_j, i_0)$ | $d(i_j, i_1)$ | $d(i_j, i_2)$ | ... | $d(i_j, i_j)$ |

In addition to storing the distances, the label and a textual human-readable payload/case descriptor is also stored along with each case ID.

**CSV** The initial file type used for storing the precomputed distances along with the data was CSV (Comma Separated Values). The first line contains a header row. Subsequent lines contain the data rows. The first data row represents case ID 0, the second represents case ID 1, and so on. Each data row is organized as follows, where distances are stored in a half $n \times n$ matrix as described above:

| Textual Case Description ‖ Case Label ‖ Distances |

These files may also be gzipped - resulting in an extension .csv.gz, and this will be automatically uncompressed by the platform. The reason support for compression was added was due to the large file-sizes that resulted from the textual representation of the distances.

CSV was chosen due to it being quite trivial to produce irrespective of the programming language. It also allowed for easy debugging, and would potentially allow other machine learning systems to be used for distance computation.

**Google's Protocol Buffers [18]** Google's blurb for Protocol Buffers is "think XML, but smaller, faster, and simpler" [18]. It is a binary format for which a proxy can be automatically generated for Python (among many other languages). It is used as the preferred method of loading data with precomputed distances in the presented platform, due to its compact representation, and efficient loading.

These files store essentially the same information in a similar method as the CSV style presented above. The devised protocol specification is as follows:

```
message Case{
    optional string payload = 1;
    required string label = 2;
}

message PrecomputedDistanceData{
  optional string data_set_name = 1;
  optional string data_set_description = 2;

  message Entry {
    required Case case = 1;
    repeated double distances = 2 [packed=true];
  }

  repeated Entry entry = 3;
}
```

Most bits to the protocol are self explanatory. The number following each field is simply the field ID. In the Protocol Buffers specification, each field must have an integer ID, with each ID unique within its local namespace. The "[packed=true]" section is simply an optimization that allows the Protocol Buffers engine store the repeated numeric values in as compact a representation as it can.

### 3.2.2 Experiment Definitions

The definitions for the experiments to run need to be provided to the platform. Each experiment consists of data[4] and the selection strategies to run on the data.

### 3.2.3 Existing Results

Existing results for selection strategies executed on the Data may also optionally be provided with the experiment. When the experiment is being executed, if it can find the results for a selection strategy within the provided existing results, these will be used as opposed to re-running that selection strategy again.

This allows for new selection strategies to be added and run, without re-computing previous strategies, but still having all outputs, such as plots, include the previous strategies' results.

It also allows for the graphs, etc. to be generated separately from the main experiment running[5], and due to ease of re-combining existing results, allows the experiment to be split, and different selection strategies be evaluated on separate machines concurrently.

## 3.3 Experiment Execution

### 3.3.1 Selection Strategy Evaluation

#### Overview

In selection strategy evaluation, the purpose is to execute a selection strategy against unlabelled training data[6], to choose cases from the set and put them in the case-base with their labels until the stopping condition is satisfied.

After each selection, a KNN based classifier is given the case-base as training data, and run on the test set, with the resulting classification accuracy recorded.

The motivation is to judge how good each selection strategy is at selecting cases to maximize the ability of the case-base to classify unseen instances.

#### Selection Strategy

A selection strategy in the platform is responsible for selecting cases to consult the oracle for, given knowledge of what is already in the case-base, and a pool of unlabelled cases to select from.

[4] The data is provided as (training set, test set) pairs to allow for Cross-Validation, etc.

[5] This can be useful, as then all the plotting/graphing dependencies need not be installed on all machines/platforms the experiment is being executed on.

[6] It's not really unlabelled - it's just from the perspective of the selection strategy it's unlabelled.

### KNN Classifier

A k-nearest neighbours based classifier is used in the evaluation, with

- $k = 5$
- Distance as defined in the dataset loaded[7].
- Distance Weighted Voting. The distance function used is:

$$\frac{1}{distance^2 + 1}$$

where $distance$ is normalized between 0 and 1.

### Cross Validation

10-Fold Cross Validation is used for training/testing against the data. Initially Orange was used to provide stratified folds. Later, when we modified our decision and lessened our reliance on Orange, a simplistic non-stratified implementation was adapted from the SciKit Learn library [19].

Many walkthroughs of K-Fold Cross Validation exist online [20], but the basic approach is to split the data into k sets[8], followed by iteratively choosing one of the sets as the test set, and the union of the remaining sets as the training set. The results of the k tests are averaged to produce a single result.

### Basic Evaluation Algorithm

```
let result_sets = {}
for each (training_set, test_set) in the CV Sets:
  let unlabelled_set = training_set
  let case_base = {}, result_set = {}
  while the stopping criteria is not met:
    let selections = select cases from unlabelled_set using
                     the selection strategy.
    ask oracle for labels for all of selections
    remove selections from unlabelled_set
    add selections to case_base with their labels
    let result = Result from testing the classifier
                 built using case_base on test_set
    add result to result_set
  add result_set to result_sets
```

[7]For tabular data, Euclidean distance was used - this functionality provided entirely by Orange. For textual data, TF-IDF vectors were generated on the raw non-stemmed documents, and the cosine similarity between documents' TF-IDF vectors computed. This functionality was provided through a combination of SciKit-Learn, and NumPy. The similarities were subtracted from 1 to give the distances.

[8]This is an entirely different k than the one in the KNN classifier

### 3.3.2 Experiment Outcome Representation

#### Result

An individual *Result* consists of:

- Size of the case-base
- Classification accuracy of the case-base
- Selections from the selection strategy that were just added.

#### Result Set

A *Result Set* is a collection of Results. For convenience and completeness, a Result Set also has knowledge of the training set and the test set that was used in generating the contained Results.

#### Multi Result Set

A *Multi Result Set* is a collection of Result Sets, which provides averaging of Results grouped by case-base size. This is necessitated by the fact that we are performing cross validation. Instead of one Result Set being produced for an experiment, k are produced where k is the number of folds of cross validation being performed.

The Multi Result Set provides an encapsulation of these k cross validation results by allowing them to subsequently act as a standard Result Set based on the averaged results, except obviously, the Training Data, Test Data and Selections fields for this will be blank - since these entities can't really be averaged meaningfully.

### 3.3.3 Summarising The Results

#### Learning Curve

A commonly presented representation of selection strategy evaluation is the *learning curve* - a plot of classification accuracy vs case-base size.

#### Area Under the Learning Curve

A metric to summarise the progression, and allow for comparison among different selection strategies against the same dataset and same starting and finishing case-base size is *AULC (Area Under the Learning Curve*). As can be expected, this is simply the area under the learning curve. Result Set objects have the inbuilt capability of computing the AULC value for their contained Results.

### 3.3.4 Cluster-Based Execution

Due to the enormity of data processing involved, as a slight deviation towards the end, cluster based platform execution was implemented. Unused lab machines were used to perform approximately two thousand compute hours worth of experiments. At peak, the platform was executing slave processes concurrently on just over 200 machines - 400 processor cores.

### Map Reduce Framework: Mincemeat

Mincemeat [16] is a relatively simple implementation of a Map Reduce framework, written in pure Python. While not as fully featured as other more developed frameworks, it was chosen for its simplicity, which allowed easy understanding, modification, and portability, having no dependencies other than Python.

### Work Unit Definition

Independent work units are first computed by the platform before providing to the map reduce framework. They represent the smallest unit of work that the platform can meaningfully execute.

```
Work Unit:
  Dataset Name
  Variation Name
  Experiment Name
  Data Definitions Name
  Raw Results File
  Total Num Folds
  Fold Number
```

### Map / Reduce Operations

The map operation takes a work unit, runs the sub-experiment specified, and returns the raw results for the specified fold.

The reduce operation takes the results for all of the folds of a single experiment variation on a dataset, and merges them to a Multi Result Set, which is then written out to disk.

### Handling Failures

In the given environment, failures of nodes and jobs was an inevitability. Computers are continuously being turned off / rebooted, and sometimes other tasks can be running on a machine meaning that a mapping operation being run on that machine would be essentially non-terminating.

Thankfully, mincemeat provided sufficient handling for these circumstances. Any time spare capacity is available, it re-sends unfinished maps to use the capacity.

Failure of the 'master' was less well handled by the framework. Maps were carried out first until all were successfully completed, followed by the reduces. If the master died, or a bug occurred, all the completed maps would be lost. The framework was modified to allow early reduction of a reduce group, if the user could give a guarantee that no more values for the reduce group would be produced in any of the remaining map operations.

### Execution Environment

One problem with using the college lab computers was the very dated Ubuntu installation present on the machines. To get around this, virtualenv [21] was used to set up an isolated Python environment in a network-hosted home directory. The '.bashrc' file was then updated to correctly activate this environment upon login.

To allow for promptless operation, public/private key pairs were generated and installed to the executing user's SSH environment. On the master node, given that root access was available, the SSH configuration was changed to automatically accept without prompt previously unseen public keys.

SSH access was then used to initiate a slave process on each available machine, after the master server on a dedicated machine was started. A script was written to first check if a slave process was currently active on the machine, and to only initiate a new one if no other was executing. This allowed "topping up" of machines as more became available. The status of the script was then provided back to the master for the accounting purpose of tracking the quantity of active machines.

## 3.4 Outputs

### 3.4.1 Raw Selection Strategy Results

The primary and non-optional outputs of the platform are the raw selection strategy results for all the provided datasets. From these, the other outputs can be calculated. It is also these raw results which can be provided to another run of an experiment so that they need not be re-computed if they are needed by the experiment.

For space efficiency, and ease of manipulation, each set of selection strategy results for a dataset are stored in a compressed archive.

These Raw Results represent the serialized form of a Multi Result Set. When the Raw Results are provided to the program, they are deserialized to Multi Result Sets.

#### Individual Cross Validation Results

For each dataset and subsequently for each selection strategy - the evaluation on each selection strategy is carried out multiple times due to the k-fold cross validation methodology used. As such, there will be k different sets of results.

Within the compressed selection strategy results archive, these are labelled 0.csv, 1.csv, etc. with each corresponding to the fold number in the cross validation splits of the dataset.

Each csv contains essentially 3 items:

1. Results for various case-base sizes, with each Result instance denoted with:
    - The case-base size
    - The classification accuracy
    - The list of selections that were made by the selection strategy just prior to this test.
2. The training set used for this fold of cross validation
3. The test set used for this fold of cross validation

#### Summary Results

Summary Results are also saved[9], and are similar to the CSV style of Raw Cross Validation Results except that they do not contain lists of selections, or the test set or training set. They are saved to a file

[9]Technically, these Summary Results are unnecessary, as they can simply be recomputed using the raw individual Cross Validation Results. They are simply there for user ease.

named “summary.csv”.

### 3.4.2 Learning Curve Plots

PyX [15] is used to generate a PDF file of plots of the learning curves for all of the selection strategies[10]. One PDF is generated for each dataset.

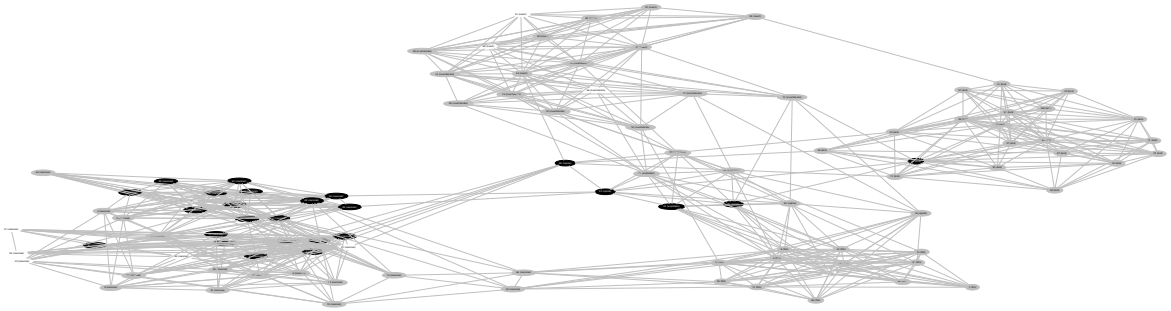
### 3.4.3 Selection Graphs

We also generate a visualisation of the case-base that each selection strategy produces. The distance matrix for the entire dataset is used to generate a spring-graph of all the cases, where each case (represented by a node in the graph) has an edge to a small proportion of its nearest neighbours, with a force proportional to the distance. Graphviz [22] was used to perform this graph generation, with PyGraphviz [23] as the interface from Python.

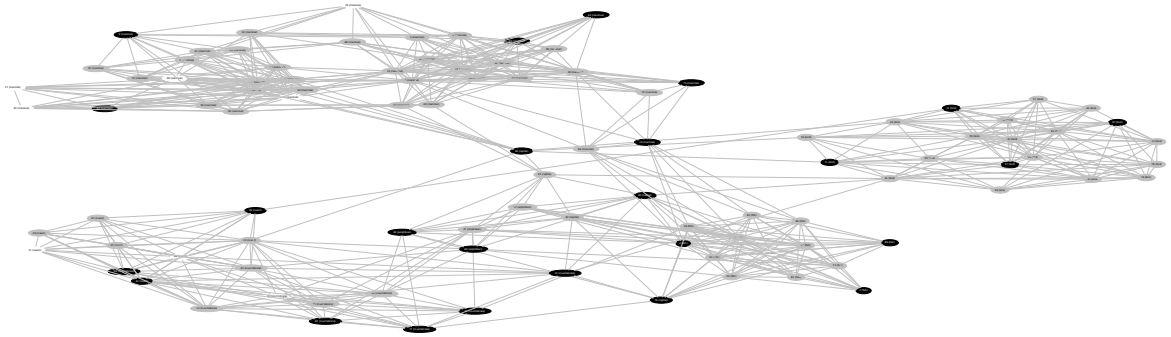
Cases which a given selection strategy chooses are highlighted at each selection iteration. These are then output to a PDF file, with each page representing an iteration in the selection strategy. Examples of the final state of these, after the stopping condition has been reached, can be seen in Figure **??**.

Because selections can't really be averaged across the different folds in cross validation, a PDF file is generated for each cross validation fold. White nodes represent test cases, grey nodes represent unlabelled cases, and black nodes represent cases selected by the selection strategy.

Due to the flattening of varying dimensionality datasets, the usefulness of these graphs vary widely depending on the datasets used, so it is not beneficial for all datasets.



(a) Selections graph for ‘Sparsity Minimization’ on ‘zoo’ dataset



(b) Selections graph for ‘Maximum Diversity Samplingn’ on ‘zoo’ dataset

Figure 3.1: Examples of Selection Graphs of final case-bases

[10]Initially - matplotlib [14] was used to generate the graphs - but its interface was somewhat messy and non-pythonic.

### 3.4.4 Experiments Summary

A summary CSV file is also output for the experiments executed during a run of the platform. It simply gives a tabular breakdown of dataset vs selection strategy AULC values. This summary also shows the ranking for each strategy on each dataset - i.e. what rank it came in on a given dataset compared to all the other selection strategies in the experiment.

## 3.5 Platform Usage

The general workflow of using the platform is relatively straightforward: provide inputs to the experimentation engine, allow the engine to run the requested experiments, and analyse the results output from the engine. This methodology provides a generic mechanism of experimenting with selection strategies.

Now, we wish to actually use this platform in researching competence based selection strategies. In the next chapter, we shall outline the core concepts involved in competence models, and some of the results which lead on from them.

# Chapter 4

# Competence Models

## 4.1 Overview

It is often useful, when thinking about an overall active learning system, to consider the competence of a case-base: the capabilities of the case-base when presented with new unseen problems. This view however is somewhat limiting when investigating the case-base in detail, as we get very little granularity. It is more useful to be able to consider the competence of individual cases, and their contribution to the overall case-base.

Smyth and Keane [24] considers the notion of individual case competence based on the 'positive' contribution a case has with regards the rest of the case-base.

Delany [25] expands on this profiling methodology to also consider the 'negative' contribution a case has. The resulting profiles better aid in representing the competence of individual cases by showing both their positive and negative contributions towards the overall case-base. She then goes on to use these profiles in considering and creating case-base maintenance algorithms.

We hope to expand on her work on competence profiles, and attempt to use them in an active learning selection strategy role as opposed to case-base maintenance.

## 4.2 Fundamentals

### 4.2.1 Nearest Neighbours / Reverse Nearest Neighbours

Before considering the definitions of the RCDL sets it is useful to consider the notion of nearest neighbours and reverse nearest neighbours, so that the RCDL sets can be framed in those terms.

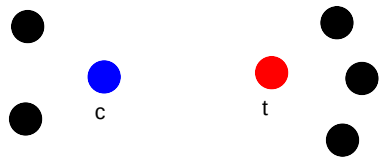
#### Nearest Neighbours (NNs)

For a case $c$, we define $NNs(c)$ as the k-nearest neighbours of $c$.

#### Reverse Nearest Neighbours (rNNs)

For a case $c$, $rNNs(c)$ is the set of cases that have $c$ in their NNs set.

**Example**




For $k = 3$:

- $t \in NNs(c)$
- $c \in rNNs(t)$

### 4.2.2 Classifies / Misclassifies

#### Case Contribution

A case $c_2$ *contributes* to another case $c_1$'s classification if $c_2$ is retrieved as a nearest neighbour of $c_1$ and $c_2$ aids 'positively' to $c_1$'s classification (i.e. is the same label as $c_1$ if $c_1$ is correctly classified, or is a different label to $c_1$ if $c_1$ is incorrectly classified).

#### Classifies

> $Classifies(t, c)$ means that case $c$ contributes to the correct classification of target case $t$. This means that target case $t$ is successfully classified and case $c$ is returned as a nearest neighbour of case $t$ and has the same classification as case $t$ [25].

Therefore, for $Classifies(t, c)$:

1. $t$ must be correctly classified by the case-base.
2. $c$ must be returned as a neighbour of $t$ in that correct classification.
3. $c$ must be of the same label as $t$'s actual label[1].

#### Misclassifies

> $Misclassifies(t, c)$ means that case $c$ contributes in some way to the incorrect classification of target case $t$. In effect this means that when target case $t$ is misclassified by the case-base, case $c$ is returned as a neighbour of $t$ but has a different classification to case $t$ [25].

Therefore, for $Misclassifies(t, c)$:

1. $t$ must be incorrectly classified by the case-base.
2. $c$ must be returned as a neighbour of $t$ in that classification.
3. $c$ must be of a different label than $t$'s actual label.

[1] By "actual label", we mean the label that the instance actually has, irrespective of how the case-base classifies it.

**Example**

In the example presented in Figure 4.1, the case $x$ is incorrectly classified as a circle by the case-base (with k=6). Here, the 3 circles (c2, c3 and c4) and the 1 square (c1) contribute to the incorrect classification, and thus:

- $Misclassifies(x, c)\ \forall c \in \{c1, c2, c3, c4\}$

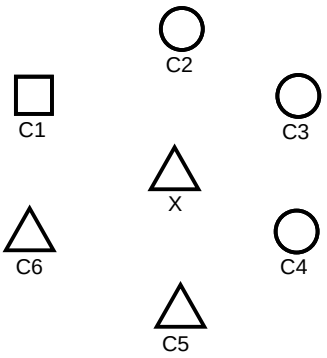



Figure 4.1: Example of case-base incorrectly classifying a case. x is classified as a Circle by the case-base, whereas its actual label is a Triangle.

### 4.2.3 RCDL Definitions

Reachability and coverage relate to cases being correctly classified by the case-base, whereas dissimilarity and liability relate to cases being incorrectly classified by the case-base.

Reachability and dissimilarity are concerned with the NNs of a case, whereas coverage and liability are concerned with the rNNs of a case.

**Reachability Set**

$$ReachabilitySet(c \in C) = \{t \in C : Classifies(c, t)\}$$

If $c$ is correctly classified by the case-base, these are all the cases in $NNs(c)$ that have the same label as $c$. Otherwise, this set is empty (and the Dissimilarity set is instead populated).

**Coverage Set**

$$CoverageSet(c \in C) = \{t \in C : Classifies(t, c)\}$$

These are all the cases $t \in rNNs(c)$ such that $t$ is correctly classified by the case-base and the label of $t$ is the same as the label of $c$.

**Dissimilarity Set**

$$DissimilaritySet(c \in C) = \{t \in C : Misclassifies(c, t)\}$$

If $c$ is incorrectly classified by the case-base, these are all the cases in $NNs(c)$ that have a different label than the actual label of $c$. Otherwise, this set is empty (and the reachability set is instead populated).

**Liability Set**

$$LiabilitySet(c \in C) = \{t \in C : Misclassifies(t, c)\}$$

These are all the cases $t \in rNNs(c)$ such that $t$ is incorrectly classified by the case-base, and the label of $c$ is different from the label of $t$.

## 4.3 Delaney's Analysis

### 4.3.1 Case Classifications

Through the definitions of the RCDL sets - Delaney notes the following [25][2]:

1. Firstly, it is possible to indicate whether the case is correctly or incorrectly classified by the case-base. This is identified by the case having either a non-empty reachability set (R) or a non-empty dissimilarity set (D).
2. Secondly, we can consider whether the case is useful, by the existence of a non-empty coverage set (C), and
3. Finally whether the case is harmful and causes damage, by the existence of a non-empty liability set (L).

Through enumeration of the possible RCDL Combinations, 8 possible profiles exist for a case, using the occurrence of a non-empty set as the *flag* on which to enumerate [25]:

1. *R*: A case which is correctly classified but is not used for classifying any other case in the case-base.
2. *D*: A case which is misclassified but is not used for classifying any other case in the case-base.
3. *RC*: A case which is correctly classified and is useful in that it has contributed to the correct classification of other cases in the case-base. This profile and the R profile are generally the majority case profile types.
4. *RL*: A case which is correctly classified but is harmful in the case-base causing damage by contributing to other cases being misclassified.
5. *DC*: A case which is misclassified but is useful in the case-base.
6. *DL*: A case which is misclassified and is harmful in the case-base.
7. *RCL*: A case which is correctly classified and is both useful and harmful in the case-base.
8. *DCL*: A case which is misclassified and is both useful and harmful in the case-base.

### 4.3.2 Experimental Conclusions

In her experiments, Delaney investigates the effect of removing cases with different RCDL profiles on case-base competence. She also compares her strategies with other noise-removal strategies, and frames them in RCDL terms.

While the specific results vary between datasets, Delaney notes that there was a consistent improvement with the removal of just DL & DCL cases from the datasets.

[2]The following profile analysis is taken verbatim from the cited paper. They gives a good break-down of the possibilities, and are simply useful to keep in mind when considering the rest of the RCDL approach - hence their inclusion.

The reader is recommended to consult Delany [25] for the thorough analysis. For the purpose of our work, the specific results are not of great importance, since we wish to try using the RCDL profiles for in a pool-based active learning scenario.

## 4.4 Continuing From Delaney's Work

The simplicity of the boolean set-non-emptiness characteristic of the case profiles Delaney defines has its merits. It is relatively easy to understand and reason about.

It does, however, throw away a lot of potentially useful information by not trying to do any reasoning about the contents of the sets.

In our work, we expand on the RCDL approach by using it in a non-boolean way for active learning selection strategies.

## 4.5 RCDL Consequences

### 4.5.1 Either-or Nature of reachability and dissimilarity sets

If a case has a non-empty reachability set then it has been classified correctly by the case-base and, as such, will have an empty dissimilarity set and vice versa.

Intuitively, these sets simply represent the set of cases that contribute to a case's classification (with the notion of *contributes* as outlined in section 4.2.2).

If the case is correctly classified, this set is referred to as the reachability set, whereas if incorrectly classified, it is referred to as the dissimilarity set.

### 4.5.2 Reachability-Coverage / Liability-Dissimilarity Duality

Simply put, if case $c$ occurs in case $t$'s reachability set, case $t$ occurs in $c$'s coverage set. Similarly with liability and dissimilarity. The reasoning is as follows:

- Take a single case pair, $c1$ and $c2$ such that $Classifies(c1, c2)$.
  - $c2$ contributes to the correct classification of $c1$.
- From $c2$'s perspective, $c1$ is in its coverage set.
  - From the formula above, from $c2$'s perspective, $c = c2$, and $t = c1$, and it is the case that $Classifies(t, c)$.
- From $c1$'s perspective, $c2$ is in its reachability set.
  - From the formula above, from $c1$'s perspective, $c = c1$ and $t = c2$, and it is the case that $Classifies(t, c)$.

Similar reasoning can be applied to the liability and dissimilarity sets.

This point is easier to understand just by intuition around the plain-English definitions of the RCDL profiles above.

### 4.5.3 Reachability is to Dissimilarity as Coverage is to Liability

This is obvious from looking at the formulas, but is worth pointing out for future notes.

### 4.5.4 Reachability/Dissimilarity concerned with Nearest Neighbours, Coverage/Liability with Reverse Nearest Neighbours

This can simply be derived from the definitions above, but of primary note is that:

- The reachability and dissimilarity sets of a case can be inferred by just looking at that case's nearest neighbours.
- The coverage and liability sets of a case can be inferred by just looking at that case's reverse nearest neighbours[3].

### 4.5.5 Max Size of Reachability and Dissimilarity sets bounded to K

Given that the reachability and dissimilarity sets of a case are always subsets of that case's NNs, the maximum size is K - the size of the NNs set[4].

## 4.6 An Incremental Strategy to RCDL

### 4.6.1 Justification

In Delaney's work, it was sufficient to calculate the case-base RCDL profiles in a brute-force manner, since it was the most efficient way in her case-base maintenance investigation on existing case-bases. For selection strategies this would not hold, since the profiles would need to be inferred many times for a changing case-base.

It would also be useful to be able to determine what RCDL changes would occur in the RCDL profiles if a new case was added with a given label. If this were to be done for each unlabelled case when selecting a case to add to the case-base, it would be very expensive to re-build the entire set of RCDL profiles for the case-base each time.

Thus, we decided to create an incremental strategy in which the changes can also be inferred in linear time before actually *adding* a case to the case-base.

### 4.6.2 Desired Algorithm Structure

#### Inputs

- A case-base
- k - the size of the neighbourhood
- RCDL, NN, and rNN profiles for each of the cases

[3]This is assuming the case-base's classification of each of the reverse nearest neighbours is known. In our later work, it will be, as it will be cached from previous classification.

[4]This is assuming that a *strict* KNN is used, where at max k neighbours are used even when ties exist.

For the remainder of this algorithm description, the term 'profile' is used to denote the RCDL profile with the NNs and rNNs:

```
CaseProfile:
  Reachability Set:  {...}
  Coverage Set:      {...}
  Dissimilarity Set: {...}
  Liability Set:     {...}
  NNs:               {...}
  rNNs:              {...}
```

**Operations**

$Suppose(unlabelled\_instance, label)$**:** This operation should determine what changes would occur to the profiles of the cases in the case-base if a given unlabelled instance was added to the case-base with a given label[5].

A Change to a given case may be denoted as the pair (Added, Removed), where Added and Removed in turn are simply partial profiles:

```
Change:
  Added:
    Reachability Set:  {...}
    Coverage Set:      {...}
    Dissimilarity Set: {...}
    Liability Set:     {...}
    NNs:               {...}
    rNNs:              {...}
  Removed:
    Reachability Set:  {...}
    Coverage Set:      {...}
    Dissimilarity Set: {...}
    Liability Set:     {...}
    NNs:               {...}
    rNNs:              {...}
```

The output for this algorithm will be a set of (Case, Change) pairs. For algorithmic ease, this also includes the input unlabelled case, since technically, adding it will cause cases to be added to its reachability set, etc. But understandably, for the input case's Change, the Removed data will always be empty.

$Add(labelled\_case)$**:** This operation simply adds a given case to the case-base, updating the case-base's profiles as appropriate. Operationally, this involves supposing that the case was added with its label using the previous suppose operation, and updating the case-base's profiles as per the returned set of (Case, Change) pairs.

[5]The label for the unlabelled instance will not be known prior to consulting the oracle, so we only want to perform "What-if" analysis - i.e. "What would happen happen if I added this case to the case-base, and it turned out to be this label".

### 4.6.3 Algorithm Outline

The rough approach taken is to first determine the changes which occur to NNs and rNNs. From these, along with knowledge of the existing RCDL profiles, the changes to reachability, coverage, dissimilarity and liability are inferred[6].

### 4.6.4 Figuring out NN / rNN changes

#### Overview

When a case is added to the case-base, figuring out the nearest neighbour and reverse nearest neighbours changes essentially involves two steps:

1. Determine the new case's nearest neighbours, and for each NN, add the new case to NN's reverse nearest neighbours set.
2. Determine the existing cases of the case-base that have the new case as a nearest neighbour, and deal with the change appropriately.

The second step is trickier than the first, since the new case may have displaced an existing case from the NNs set, therefore this needs to be taken into account when maintaining consistency of NNs to rNNs in all cases concerned.

#### Algorithm

```
1 // Sort out the new case's nearest neighbours
2 Determine new case's NNs
3 For each NN of the new case's NNs:
4   add the new case to the NN's rNNs
5
6 // Sort out the new case's reverse nearest neighbours.
7 For each existing case that has the new case as an NN:
8   Add the existing case to the new case's rNNs
9   Add the new case to the existing case's NNs
10  If the new case shunted an NN of the existing case:
11    Remove shunted case from the existing case's NNs
12    Remove existing case from the shunted case's rNNs
```

#### Algorithm Notes

**Determining if an existing case has the new case as an NN within k** Unfortunately, each case in the case-base needs to be examined to determine if it has the new case as an NN. But because we have the NNs stored for each case in the case-base, the examination of each case can be a relatively quick operation, versus the alternative of having to, for each case, go through the entire case-base to find its NNs if they weren't stored.

[6]For clarity, the outline refers to updating the actual case-base profiles collection, whereas really it's a Change object for each one that it modifies, and finally returns. While, at this stage, this is simply an implementation detail, it will become important when we start using the suppose operation (Section 5.2.1).

The maximum NN distance in the cached NN list need only be compared to the distance of the new case, with consideration for tie breaking as in the KNN classifier if they're equal.

**Duality of NN/rNN changes (both added and removed)** If a case has a nearest neighbour added, then that nearest neighbour will get a reversed nearest neighbour added. Thus, for any NN add, there is a corresponding rNN add. Similarly for remove.

This allows us just to iterate through the NN changes, and infer the presence of the corresponding rNN change as a result.

**NN Shunting** We define a case $A$ as *shunting* another case $N$ from the NNs set of $B$ if $N$ is removed as a nearest neighbour of $B$, and $A$ added.

An example can be found by looking at Figure 5.3. Here, for $k = 3$S, $Cn$ *shunts* $C1$ from $Cp$'s NNs set.

### 4.6.5 The Incremental RCDL Profile Building Algorithm

#### Overview

The algorithm uses the NN/rNN changes determined using the previous algorithm (Section 4.6.4), and goes through each NN change[7], handling individually. The core of the algorithm proceeds in roughly two parts:

1. Deal with the NN removals first
2. Deal with the NN additions, taking into account scrubbing and the effect of classification changes.

The primary issue that requires careful consideration is when the addition of the new case causes the classification of an existing case in the case-base to change - since the RCDL profiles relate heavily to whether a case has been correctly classified or not.

[7] Because of NN/rNN duality - changes to rNNs, and sets relating to rNNs (coverage and liability), can be inferred.

### Algorithm

```
Compute the NN/rNN changes using the suppose_nn algorithm.
foreach case in the NN/rNN changes:
  // Deal with NN removals, if present
  foreach removed_case in case's NN removals:
    if the removed_case was in case's reachability set:
      Remove removed_case from case's reachability set
      Remove case from removed_case's coverage set.
    else if removed_case was in case's dissimilarity set:
      Remove removed_case from case's dissimilarity set
      Remove case from removed_case's liability set.

  // Deal with NN additions, if present
  Calculate case's new classification
  if the case's correct-classification-status changed:
    // Need to scrub appropriate RCDL sets
    for each r in the case's old 'contributors' list:
      if the case's old classification was correct:
        Remove r from case's reachability set
        Remove case from r's coverage set
      else:
        Remove r from case's dissimilarity set
        Remove case from r's liability set

  for each c in the case's new 'contributors' list:
    if the case's new classification is correct:
      Add c to case's reachability set
      Add case to c's coverage set
    else:
      Add c to case's dissimilarity set
      Add case to c's liability set
```

### Algorithm Notes

**Ability to just look at NN changes** Due to the NN/rNN Duality property described above with regards NN changes, any NN add is guaranteed to have an rNN add, and similarly with remove. This allows us to consider only NN changes, and while dealing with an NN change, deal with the corresponding rNN change.

**Permutations of NN changes** Because of the previous note, we only really need to deal with the added and removed nearest neighbours. The following are the permutations of NN changes, following 'supposing' the addition of a single case:

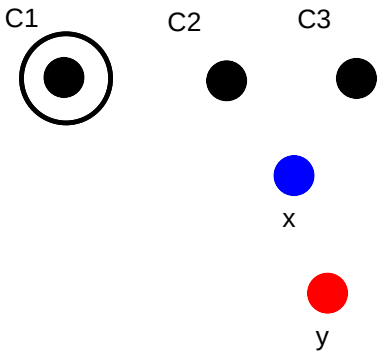
| NN Added | NN Removed | Notes |
|---|---|---|
| 1 | 1 | Will be the only type when $\|CaseBase\| > k$. |
| 1 | 0 | Happens only at the start, when $\|CaseBase\| \leq k$. |
| 0 | 1 | Can never actually happen - there would have to be another Case to replace the removed one since we're only dealing with adding a case to the case-base. |
| 0 | 0 | Uninteresting - its rNNs will be covered elsewhere with the corresponding NN changes of another case. |

**An NN Removal Guarantees an NN Add** This can be seen in the truth table above. Because we're only dealing with a new case being added, the effect of the new case can never cause an NN removal without causing a corresponding NN add of the new case. Because of having a shorter distance to the target case than the furthest-distance NN, the new case will have *shunted* that NN - causing an NN removal.

This allows us to not worry about, at the stage of dealing with NN removals, if the removal caused the label to change, because we'll be dealing with that scenario anyway in the corresponding add.

**An NN removal might affect R/D of case in question, C/L of other** For the following example,

- $k = 3$,
- $x$ is the case in question whose NNs we'll talk about
- $y$ is the new case added to the case-base
- $c1$, $c2$, $c3$ are the NNs of $x$ prior to $y$ being added
- $c1$ is the *shunted* case which is removed from NNs(x) after $y$ is added.




$c1$ was an NN of $x$, and $x$ was an rNN of $c1$. As described previously in 4.5.4, a case's NNs can affect its reachability / dissimilarity sets, whereas a case's rNNs can affect its coverage / liability sets. So the removal of $c1$ as an NN of $x$ (and the corresponding removal of $x$ as an rNN of $c1$) might affect $c1$'s reachability / dissimilarity set, and $x$'s coverage / liability set, because when $c1$ is removed as an NN, it can't contribute either positively or negatively any more to $x$'s classification.[8]

All this leads to a simple way to dealing with an NN removal [9]:

- If $c1$ was in $x$'s coverage / liability set, it should be removed.

[8] Note that it is not guaranteed that $c1$ was in $x$'s reachability / dissimilarity set, or that, equivalently, $x$ was in $c1$'s coverage / liability set. The sets will only be affected if $c1$ was a *contributor* of $x$'s classification by the case-base. Hence the usage of "might affect".

[9] The corresponding NN addition will be dealt with separately

- If $x$ was in $c1$'s reachability / dissimilarity set, it should be removed.

**Calculating a case's new classification** This can be done semi-intelligently since a case's old NNs are known. Instead of doing KNN classification with the entire case-base, we only need to look at a case's new NNs (which can be inferred from its old NNs and the NN changes) to determine the new classification.

**Correct-classification-status** We will furthermore commonly talk about a case's correct-classification-status as opposed to its classification alone. It simply means whether a case has been correctly classified by the case-base or not.

**A case's correct-classification-status determining reachability/dissimilarity** If a case is correctly classified by the case-base, its reachability set will be populated. If it is incorrectly classified, its dissimilarity set will be populated.

**Effect of a correct-classification-status changing** Firstly, we need only worry about if the correct-classification-status of a case changes with the addition of a new case, as opposed to if the classification alone changes.

For example, if a case's actual label was 'triangle', it was previously classified 'square', and with the addition of the new case, it's classified 'circle', it has still been incorrectly classified both before and after the addition of the new case, therefore we do not need to worry about scrubbing.

If however the correct-classification-status of a case $c$ does change, we need to perform *scrubbing* on some of the RCDL sets.

If it went from correctly classified to incorrectly classified:

- $c$'s reachability set needs to be scrubbed (all items removed).
- For each *scrubbed* case from the reachability set, the scrubbed case needs to have $c$ removed from its coverage set.
- $c$'s dissimilarity set needs to be populated with the elements in its new NNs that have a different label to its actual label.
- For each thing put in the dissimilarity set, their liability sets need to have $c$ added.

Similarly, if it went from incorrectly classified to correctly classified:

- $c$'s dissimilarity set needs to be scrubbed (all items removed).
- For each *scrubbed* case from the dissimilarity set, the scrubbed case needs to have $c$ removed from its liability set.
- $c$'s reachability set needs to be populated with the elements in its new NNs that have the same label as its actual label.
- For each thing put in the reachability set, their coverage sets need to have $c$ added.

**Not allowing supposing of an already present case** The algorithm presented does not allow for an already present case in the case-base to be *supposed* (e.g. if x is in the case-base with label 'square', what would happen if x was actually 'circle').

**Not allowing for supposing the removal of a case** The algorithm presented does not allow for supposing the removal of a case from the case-base (e.g. what would happen if I removed x from the case-base?), though it could be updated to support this scenario.

### 4.6.6 Testing and Validation

Testing was performed in Unit-Test style to ensure the RCDL profiles achieved using the incremental strategy presented correlated exactly with a simple brute-force strategy. This was done on each of the datasets used.

#### Brute Force Algorithm

```
// Compute NNs and rNNs
foreach case in case_base:
  case_rcdl = case's RCDL Profile
  case_nns = the k-nearest neighbours of case
  Add case_nns into case_rcdl.nearest_neighbours
  foreach nn in case_nns:
    nn_rcdl = nn's RCDL Profile
    case_to_profile_dict[nn].reverse_nearest_neighbours.add(case)

// From NNs & rNNs, compute the remainder of the RCDL Profiles
foreach case in case_base:
  case_rcdl = case's RCDL Profile
  actual_label = oracle(case)
  classified_label = Classify using case_rcdl.nearest_neighbours
  case_rcdl.classification = classified_label

  if actual_label == classified_label:
    helping_cases = {nn for nn in case_rcdl.nearest_neighbours
                     if oracle(nn) == actual_label}
    Add helping_cases to case_rcdl.reachability_set
    foreach hc in helping_cases:
      hc_rcdl = hc's RCDL Profile
      Add case to hc_rcdl.coverage_set
  else:
    hurting_cases = {nn for nn in case_rcdl.nearest_neighbours
                     if oracle(nn) != case_actual_label}
    Add hurting_cases to case_rcdl.dissimilarity_set
    foreach hc in hurting_cases:
      hc_rcdl = hc's RCDL Profile
      Add case to hc_rcdl.liability_set
```

### The Test

```
Shuffle the dataset
incremental_builder = A new instance of the Incremental RCDL Builder
case_base = {}

foreach case in dataset:
  Add case to case_base

  // Use the incremental builder to get the rcdl profiles
  profile_builder.put(case)
  incr_profiles = profile_builder.case_profiles

  // Use Brute Force to get the rcdl profiles
  bf_profiles = Build the profiles brute force from the case_base

  // Make sure that they're the same size so I don't 'miss' any
  assert that |br_profiles| == |incr_profiles|

  // Iterate through each profile, and make sure its
  // brute force equivalent is the same
  for (pb_case, pb_case_profile) in incr_profiles:
      bf_case_profile = Get pb_case's case profile from bf_profiles
      assert that pb_case_profile == bf_case_profile
```

### Timings

As can be seen in Figure 4.2, using the incremental algorithm to add a single case to the case-base is much faster than re-building the entire profile collection from scratch. There is however an overhead to using the incremental strategy, as can be seen by the cumulative times. For general application, it is thus better to use a brute force style if the profiles need only be computed once ever, but if the case-base is being continually modified, and the profiles needed, the incremental strategy has its obvious merits.

Figure 4.3 shows a plot of just the incremental timings, since it is difficult to make out in Figure 4.2.

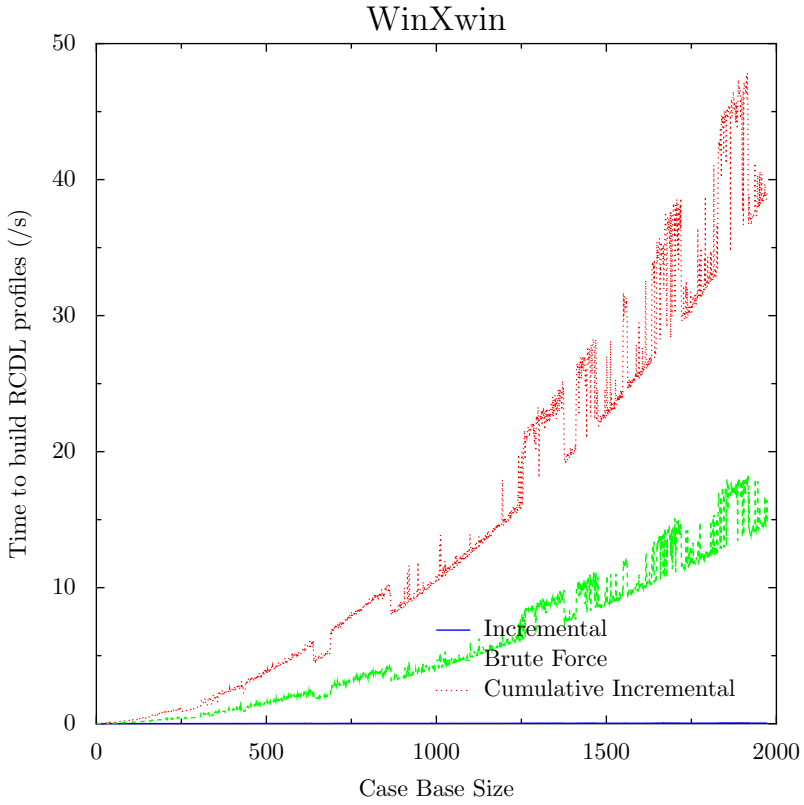



Figure 4.2: Plot of timings information for brute force algorithm, incremental algorithm, and the cumulative time of the incremental algorithm on dataset 'WinXwin'

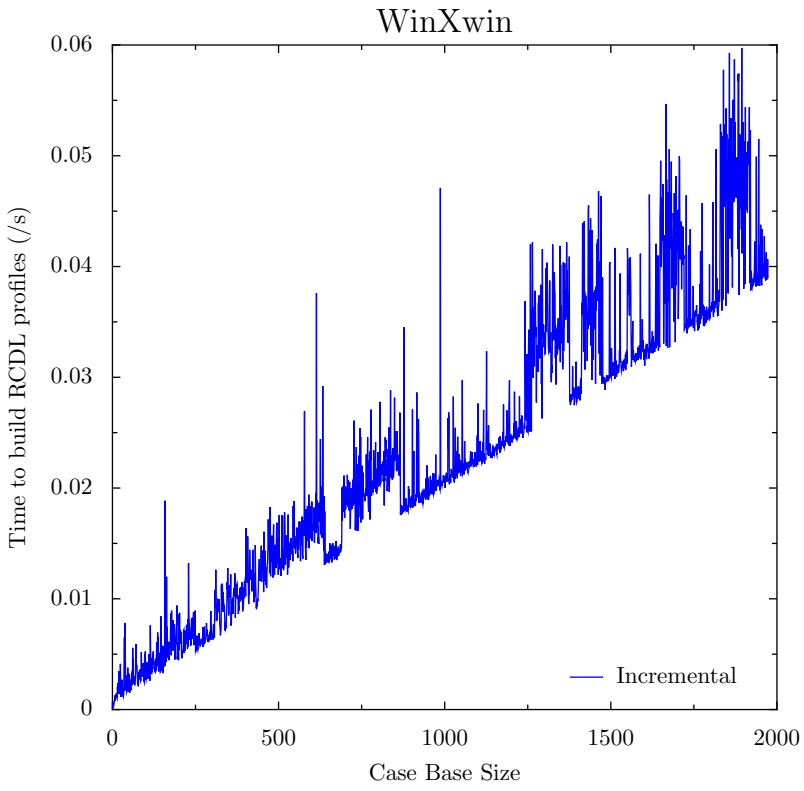



Figure 4.3: Plot of timings information for incremental algorithm on dataset 'WinXwin'

# Chapter 5

# Competence Based Selection Strategies

## 5.1 Types of RCDL Changes from adding a Single Case to the Case-Base

### 5.1.1 NNs of the New Case

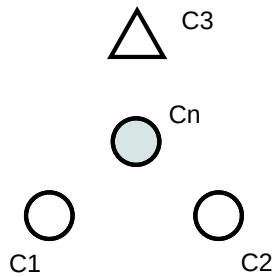



Figure 5.1: Nearest neighbours causing the correct classification of a case as a Circle.

In the example presented in Figure 5.1, the NNs of the added case Cn are {C1, C2, C3}. Cn is correctly classified by the case-base as a circle, thus we are dealing with the reachability and coverage sets.

- Cn will have a reachability set consisting of the cases which contribute to its correct classification - {C1, C2}.
- Similarly, because C1 and C2 both contribute to the correct classification of Cn, they will have Cn added to both their coverage sets.

If Cn had been incorrectly classified - it would instead have been the dissimilarity and liability sets affected - e.g.

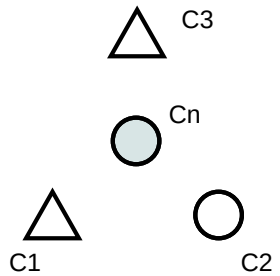


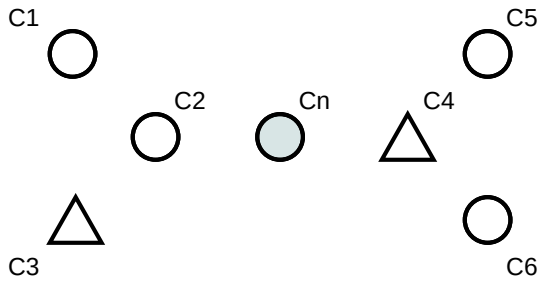

Figure 5.2: Nearest neighbours causing the incorrect classification of a case as a Triangle.

In the example presented in Figure 5.2, C3 and C1 contribute to the incorrect classification of Cn, so

- Cn will have a dissimilarity set consisting of {C3, C1}.
- Similarly, Cn will be added to C3 and C1's liability sets.

### 5.1.2 rNNs of the New Case

The rNN related changes are somewhat trickier, since they are not bounded to k, and have the potential to have knock-on effects to other cases not directly related to the new case. The running example throughout will be as follows:




k = 3, and distance weighted voting is used. C2 and C4 will be the primary cases of interest, with Cn the new case added (or supposed).

Before Cn's addition, C2 was incorrectly classified by the case-base as a triangle, with C3 and C4 contributing to it, and C4 incorrectly classified as a circle, with C2, C5 and C6 contributing to it.

After Cn's addition, C2 is correctly classified by the case-base, with C1 and Cn contributing to it and C4 is still incorrectly classified by the case-base, but now with Cn, C5 and C6 contributing to it.

#### Direct

When a new case is added to the case-base, it may contribute in some fashion to the classification of some of its rNNs.

In the example, C2 is correctly classified by the case-base as a circle, and Cn, the new case added, contributes to the correct classification. As a result,

- Cn's coverage set will have C2 added to it,
- C2's reachability set will have Cn added to it

C4, on the other hand, is incorrectly classified by the case-base, and the new case Cn is contributing to it.

- Cn's liability set will have C4 added to it,
- C4's Dissimilarity set will have Cn added to it.

#### Case NN Shunting

When a new case is added to the case-base, it is possible that it will become one of an existing case's nearest neighbours, and 'shunt' some other case for the position, i.e. the 'shunted' case was an NN of

some case, but after the new case is added, it is not. Here, we talk about actions relative to the case for which its NNs changed as a result of the new case being added.

In the example, C2, C5 and C6 were the contributing neighbours to C4's incorrect classification as a circle. When Cn was added, it was closer to C4 than C2, and so C2 was removed as an NN, and Cn added. Cn too contributes to C2's incorrect classification. As a result of the NN changes:

- C2 is removed from C4's dissimilarity set
- C4 is removed from C2's liability set
- Cn is added to C4's dissimilarity set (covered under Direct as in Section 5.1.2)
- C4 is added to Cn's liability set (covered under Direct as in Section 5.1.2)

| Case Correctly Classified | Shunted Contrib'd | Action |
|---|---|---|
| 1 | 1 | • *Shunted* removed from *case*'s reachability set<br>• *Case* removed from *Shunted*'s coverage set |
| 0 | 1 | • *Shunted* removed from *case*'s dissimilarity<br>• *Case* removed from *Shunted*'s liability |
| 0 | 0/1 | Uninteresting. A case was shunted, but it didn't contribute previously, so it will be unaffected by the shunting. |

**Correct-Classification-Status Flip**

The new case's rNNs might have changed their correct-classification-status due to the addition of the new case. This may result in other cases not directly related to the new case, but instead related to the rNN, to be affected.

In the example, C2 was incorrectly classified as a triangle before the addition of Cn, with C3 and C4 the contributors to this classification. After Cn is added, its classification changes to the correct classification - a circle. We say that its correct-classification-status has changed. As a result, C1 now contributes to the classification (in addition to the new case Cn), and C3 does not. C4 was shunted from its position and is covered instead under Case NN Shunting.

Changes as a result in particular of the correct-classification-status flip:

- C3 is removed from C2's dissimilarity set
- C2 is removed from C3's liability set
- C1 is added to C2's reachability set
- C2 is added to C1's coverage set

In addition, the effect of "NN Shunting" and "Direct" come into play with respect to C2:

- C4 is removed from C2's dissimilarity set (NN Shunting)
- C2 is removed from C4's liability set (NN Shunting)
- Cn is added to C2's reachability set (Direct)
- C2 is added to Cn's coverage set (Direct)

| Change | Action |
|---|---|
| Correct to Incorrect | • The case's reachability set needs all items removed from it.<br>• Each item removed needs to have the case removed from its coverage set.<br>• The case's dissimilarity set needs all NNs of a different label added to it.<br>• Each item added needs to have the case added to its liability set. |
| Incorrect to Correct | • The case's dissimilarity set needs all items removed from it.<br>• Each item removed needs to have the case removed from its liability set<br>• The case's reachability set needs all NNs of a different label added to it.<br>• Each item added needs to have the case added to its coverage set. |

### 5.1.3 RCDL Change Possibility Matrix from a Case Addition

| | | | Existing Cases | | |
|---|---|---|---|---|---|
| | | New Case | Direct | Shunt | Flip |
| Any RCDL Set | Add | X | X | | X |
| | Remove | | | X | X |

The above table summarises in brief the findings of the possible types of changes that can happen from adding a new case the case-base. For example:

- The mark corresponding to "New Case", "Add" means that, as a result of the addition of the new case, any of the new case's RCDL sets might have elements added to them.
- The mark corresponding to "Existing Cases Shunt", "Remove" means that, as a result of the addition of the new case, shunt type changes can involve removing cases from the RCDL sets of the existing cases.
- The absence of the mark corresponding to "Existing Cases Shunt", "Add" means that RCDL sets of existing cases will not be added to as a result of Shunt type changes as defined previously.

The table may simply be used as a reference when considering the formulae presented later when describing competence selection strategies.

## 5.2 Selection Strategy Pre-Notes

### 5.2.1 The Act of Supposing

When an unlabelled case is evaluated for inclusion in the case-base, the actual label for the case is unknown. It could have any of the possible labels. As a result, when considering a case for inclusion, it is desirable to evaluate its benefit for all possible labels, possibly considering the likelihood of the label given what we already know through the case-base.

### 5.2.2 Measure Deviation between the Possible labels

For some strategies, the deviation between the changes for the different possible labels is also of interest.

### 5.2.3 Categorizing Suppose Changes

It should first be noted that, due to the occurrence of Shunt and Flip type changes, a single case change (as described in Section 4.6.2) can concurrently contain multiple RCDL change types.

It would be useful for when developing new selection strategies to be able to determine whether a particular change were a 'Direct', 'Flip' or 'Shunt'.

#### Suppose Change Set SFD Categorization Algorithm

```
Algorithm CategorizeChanges(change_pairs, new_case):
  let (direct_cps, shunt_cps, flip_cps) = ({ }, { }, { })
  for (existing_case, change) in change_pairs:
    if existing_case == new_case:
      // All changes regarding the new case are regarded as Direct,
      // no extra work needed
      direct_cps[existing_case] = change)
      continue

    // Want to iterate over all changes (both added and removed)
    // of reachability and dissimilarity, and categorize each
    // case change individually
    for (rcdl_set, op) in cross product of
                          {'reachability', 'dissimilarity'}
                    and {'Added', 'Removed'}:
      for other_case in 'op' of 'rcdl_set' of change:
        if other_case == new_case:
          let change_type = 'direct'
        else if correct-classification-status did change
             and (other_case contributed to existing_case's
                      previous classification):
          let change_type = 'flip'
        else:
          let change_type = 'shunt'

        Add this change to existing_case pair in change_type's CPs set
            (i.e. shunt_cps, flip_cps or direct_cps)

  // All reachability and dissimilarity sets have been dealt with,now we
  // need to use the reachability and dissimilarity sets of each to infer
  // and make the changes for coverage and liability changes.
  for cps in (direct_cps, shunt_cps, flip_cps):
    PlaceDuals(cps)
```

**Flips taken to 'overpower' Shunts**

If a correct-classification-status flip occurs for a case, and the shunted case would have been removed by the flip anyway, even if it were not shunted, we choose to categorize this as a flip.

The justification for this is to not require similarity measures, or knowledge of k, at this stage of the algorithm, instead completely treating the computation of the profile changes as a black box, and operating only on the outputs of this black box.

**Example:** In Figure 5.3, k=3, and $Cn$ is the new case added. Previously, $C1$, $C2$ and $C3$ were the NNs of $Cp$, and $Cp$ was incorrectly classified as a Circle.

At that time,

- $ReachabilitySet(Cn) = \{\}$
- $DissimilaritySet(Cn) = \{C1, C2\}$

When $Cn$ was added, $C1$ was shunted from its membership in the NNs set, $Cn$ is correctly classified as a Triangle by the case-base, and the correct-classification-status of $Cn$ is flipped. Now:

- $ReachabilitySet(Cn) = \{C3, Cp\}$
- $DissimilaritySet(Cn) = \{\}$

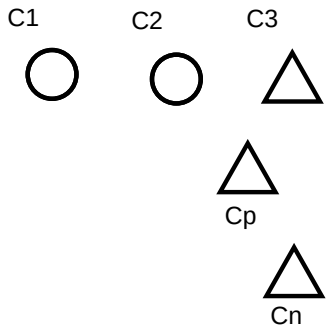



Figure 5.3: Example of a Flip/Shunt Scenario.

The resultant 'change' to the case-base for $Cp$:

- Added:
  - $ReachabilitySet(Cn) = \{C3, Cp\}$
- Removed:
  - $DissimilaritySet(Cn) = \{C1, C2\}$

From this information alone, we cannot infer which of $C1$ and $C2$ was shunted (or even that anything was shunted if we don't know $k$). To minimize the information requirements that this algorithm carries, we thus decide to consider $C1$ and $C2$ as flip-related changes.

**Inferring the changes of coverage / liability from reachability / dissimilarity**

The duality property as described in Section 4.5.2 allows us just to consider reachability and dissimilarity changes, and infer coverage and liability changes respectively. Reachability and dissimilarity are chosen due to easier reasoning about programatically, since they deal with NNs as opposed to rNNs.

The operation `PlaceDuals(CPs)` represents iterating through a given Change set, and for each reachability / dissimilarity related change, adding the corresponding duality set change (coverage / liability).

### 5.2.4 Counting Shunts

If a shunt related remove happens, it will happen for *every* possible label with no exception. This duplication holds no benefit for final weighting, since shunts would be counted for each of the number of labels there are. We therefore decided to only total shunts for a single label.

### 5.2.5 Counting Flips

If there are more than 2 labels and we totalled across different labels, both flip directions *might* lead to multiple-counting, depending on the exact cause of the flip. A flip may occur for some, or all of the supposed labels.

We do not attempt to count unique instances here, as the weighting attached due to the duplicates is somewhat useful. If a flip occurs for e.g. 3 out of 4 labels for a case, it is somewhat more relevant than if it occurs for 1 of the 4 labels.

## 5.3 A Grammar for Naming Competence Strategies

Due to the availability of compute resources with the incorporation of cluster-like functionality in the platform developed, it was possible to execute a large quantity of competence-based selection strategies. Naming individually every single strategy would be unwieldy at best, therefore a naming convention was adopted based on the various decision points in the strategies we developed.

### 5.3.1 Decision Points

#### Locality

*Locality*, in the context of our selection strategies, indicates which cases' RCDL sets we consider during the selection among the candidate cases.

*Local* refers to just the candidate case's RCDL sets being considered, while *Global* relates to the candidate case's effects on the RCDL sets of the cases already in the case-base.

While there certainly is overlap due to the duality property, it is also possible that other cases not directly related to the candidate case can be affected due to the candidate case's addition. Some of these side-effects, in particular what we term 'flips', can be quite interesting.

#### Cross Label Aggregation

RCDL profiles are highly dependent on knowing the labels of the cases in question. Given that we do not know what the label of a candidate case will be before we provide it to the oracle, it is necessary to consider "what-if" style analysis when considering the RCDL effects of a candidate case.

For the purpose of combining the possible labels' measures, we chose two basic aggregation approaches:

1. Total:

   Simply adding the overall measure values produced for each of the labels.

2. Deviation:

   Calculating the standard deviation between the values of the different possible labels.

#### Quantity Goal

The overall goal of the strategies we developed is based on *minimizing* or *maximizing* the scores of the individual cases.

#### Density Inclusion

Density information regarding the dataset can also be incorporated as a measure in the overall strategy. The density measure is defined as the average similarity to all other cases in the entire dataset, stretched to the range 0 to 1. The sparsity of a case we simply define as 1 minus the density of the case.

When either measure is included, it is combined with each RCDL measure in a non-weighted average fashion.

#### Set to Scores Mapper

The set to scores mapper defines the mechanism to map a single R, C, D or L set for a single case, to a sequence of scores. This may not necessarily be a one-to-one mapping for the cases contained in the set.

We define 4 simple maps:

1. All-Pairs Similarity (incl source):

   This maps a set to the similarity values for the all-pairs of the set plus the source case from which the set emanates.

2. All-Pairs Similarity (excl source):

   This is similar to the previous mapper, except it does not include the source case from which the set emanates.

3. Direct Similarity:

   This maps a set to the similarity values between the source case and each case in the set in question.

4. Counting:

   This simply maps a set to a sequence of ones of length equal to the size of the set.

#### Set Scores Reducer

The reducer here combines a set of scores down to a single score. Two simple reducers are defined; total, and average.

#### RCDL Sets to Use

Another decision point for a strategy is which of the RCDL sets to consider in the strategy. For our strategies, for a given strategy, the other decision points applied to one set are also applied separately to the others. The same decision points are used for each of the RCDL sets in the strategy.

It should be noted that this grammar design does not allow representation of a selection strategy with more than one RCDL set, but based on different measures, but is sufficient for us given that we only experimented with basic hybrid strategies.

#### Measures Reducer

The individual RCDL measures for a strategy need to be combined in some fashion. We chose simple summing, this being equivalent to non-weighed average given that the denominator is the same for every candidate, and given that no more combinations are done subsequent to this last step. We represent this simply as '+'. The other possibility considered was multiplication, but we decided against this due to the issue with how to deal with zero values, but in our naming convention, would be represented with '*'.

### 5.3.2 Decision Points Summary

| ID | Decision Point | Possible Values | # Values |
|---|---|---|---|
| 1 | Locality | Local, Global | 2 |
| 2 | RCDL Sets | {Reachability}, {Dissimilarity} . . . {Reachability, Coverage}, {Reachability, Dissimilarity} . . . | 17 |
| 3 | Measure Reducer | +, * | 2 |
| 4 | Set Scores Reducer | Average, Total | 2 |
| 5 | Set to Scores Mapper | All Pairs Similarity (incl source), All Pairs Similarity (excl source), Direct Similarity, Counting | 4 |
| 6 | Density Inclusion | With Density, With Sparsity, (None) | 3 |
| 7 | Cross Label Aggregator | Total, Deviation | 2 |
| 8 | Quantity Goal | Minimization, Maximization | 2 |

### 5.3.3 Combining Decision Points to a Name

Decision points are used to generate a string as follows, where '[ID]' represents the value of the decision point referenced by 'ID' in the matrix above:

> [1] ([2] with items separated by [3]) [4] [5] [6] Cross-Label [7] [8]

### 5.3.4 Example Strategy Name

For this example, we shall consider:

> Local Liability + Coverage Total Counting (With Sparsity) Cross-Label Total Minimization

The decision points are then extracted as follows:

| ID | Decision Point | Value |
|---|---|---|
| 1 | Locality | Local |
| 2 | RCDL Sets | Liability, Coverage |
| 3 | Measure Reducer | + |
| 4 | Set Scores Reducer | Total |
| 5 | Set to Scores Mapper | Counting |
| 6 | Density Inclusion | With Sparsity |
| 7 | Cross Label Aggregator | Total |
| 8 | Quantity Goal | Minimization |

Stepping through each in turn:

1. Locality: Local

   Just the RCDL sets of the candidate case are considered.

2. RCDL Sets: Liability, Coverage

   The measure subsequently defined shall be calculated separately for each of the liability and coverage sets of the candidate case.

3. Measure Reducer: +

   The value for the liability based measure and the value for the coverage based measure will be combined through simple summing.

4. Set Scores Reducer: Total

5. Set to Scores Mapper: Counting

   The sequence of values generated through the mapper (counting) will be reduced by totalling. In this strategy, this is simply the cardinality of the RCDL set in question. This will be used as the overall score for that set.

6. Density Inclusion: With Sparsity

   Density information shall be incorporated through the notion of sparsity, with sparsity combined with each RCDL based measure through non-weighted average.

7. Cross Label Aggregator: Total

   A score will be calculated for each possible label, with the scores combined to an overall score through simple summing

8. Quantity Goal: Minimization

   The overall goal of the strategy will be minimization. This means that for each candidate set, a value will be assigned by the strategy, and the candidate set with the lowest value will be selected (with random tie breaking used in the case of ties for lowest).

## 5.4 Selection Strategies

### 5.4.1 Simple Single-Set Counting Strategies

#### Local Reachability (Counting)

When Supposing a single case for addition to the case-base, irrespective of the label we choose to suppose, the case-base will classify it with only one label. For example, in Figure 5.1, irrespective of if we suppose Cn to be a triangle, or a circle, the case-base will still classify it a circle. Thus, the reachability set of Cn will only get populated in one instance - when we suppose Cn to be the same label as the case-base classifies it.

This holds true in Supposing any case for the case-base. A supposed case will only have its reachability set populated for one single label for a case-base state - the label that the case-base classifies it as. Assuming a non-empty case-base, for when the reachability set is populated - the cardinality of the set will always be between 1 and k.

With the absence of further knowledge, we could guess that when the reachability set for a new case is small compared to other candidate cases, the case-base is relatively uncertain about the new case. As a result, by choosing a new case from the unlabelled set which results in a minimally sized reachability set for that new case, we get a basic form of uncertainty sampling[1].

This strategy takes this approach exactly - assigning an overall score to each unlabelled case equal to the total of the reachability set supposes across all the different possible labels[2]. The unlabelled case with the minimum score is then selected. Ties are broken at random.

#### Local Coverage (Counting)

The intuition around this strategy is that by trying to find unlabelled cases which have a roughly equal positive effect for the different possible labels, we get cases which are close to classification boundaries.

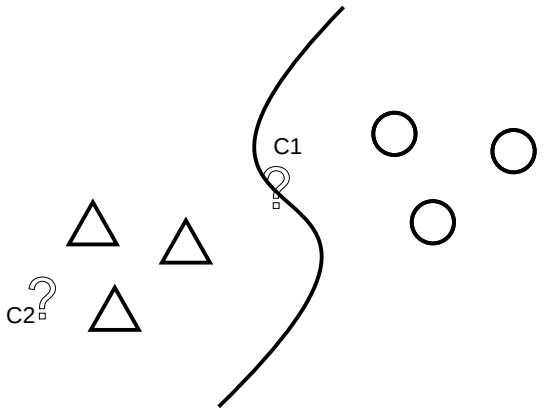



Figure 5.4: Example of classification boundary for 2 areas (Triangle and Circle), with two candidate cases for selection, C1 and C2.

[1] If we were using non-weighted voting in our KNN implementation, we would get exactly uncertainty sampling. With distance weighted voting, it's not quite the same, since our basic strategy simply takes only the cardinality of the reachability set into account, and not the distances of the cases within it.

[2] Totalling is sufficient because we are guaranteed that the reachability set will be empty for all but one of the supposes. So totalling is equivalent to just taking the value of the non-empty reachability set.

In the example presented in Figure 5.4, k=4, and we have two candidate unlabelled cases C1 and C2. With the case-base as it stands, the classification boundary we can guess is in the middle of the circle and triangle regions (denoted by the line).

Intuitively - if we knew the value of C1, it would better tell us where the classification boundary should be. If it turned out to be a circle, it would push the classification boundary left towards the triangles. Similarly, if it turned out to be a triangle, it would push the classification boundary right towards the circles.

C2 would be less useful to retrieve a label for, since compared to C1, we know more about the region in which it lies.

If we were to look at the size of the coverage sets of C1 and C2 for the different possible classifications:

| | Coverage if Triangle | Coverage if Circle | Standard Deviation |
|---|---|---|---|
| C1 | 3 | 3 | 0 |
| C2 | 3 | 0 | 2.12 |

This strategy chooses the case with the minimum Standard Deviation between the possible labels, with random tie-breaking used.

One thing that initially falls out of this strategy is that, during the early selections, it will likely choose relatively diverse cases. Unlabelled cases which have no rNNs in the case-base will have empty coverage sets irrespective of their supposed label, and thus, a standard deviation of 0.

After all of the diverse cases have been selected, it will then move onto balancing more similar to as described in the example above.

**Local Dissimilarity (Counting)**

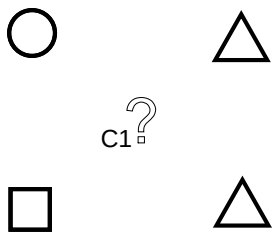



Figure 5.5: Example of unlabelled case relative to its k-nearest-neighbours within the case-base.

In the example presented in Figure 5.5, case $C1$ is an unlabelled example being supposed, with $k = 4$. Three labels are shown, Square, Circle and Triangle - but a fourth is also present in the dataset - an X. $C1$ will be classified as a Triangle by the case-base.

If we consider the dissimilarity set of the new case for the different supposes, we concentrate on the other cases which contribute to its incorrect classification. We get the following table:

| Supposed Label | Dissimilarity Set | Description |
|---|---|---|
| Triangle | {} | If it's a Triangle, then it is correctly classified by the case-base, and so won't have a Dissimilarity set. |
| Square | {Triangle, Triangle, Circle} | Incorrectly classified. The Square isn't aiding in the misclassification, so it's only the remaining NNs. |
| Circle | {Triangle, Triangle, Square} | Incorrectly classified similar to above. Circle isn't aiding, so just the remaining NNs. |
| X | {Triangle, Triangle, Circle, Square} | Incorrectly classified. All the NNs are contributors. |

If we use $C_P$ to denote the set of cases in the NNs of $C1$ where the case is of the same label as the case-base's classification of $C1$, and $C_N$ to denote the set of cases in the NNs with a different label. Let $L$ denote the set of possible labels. In the above example $L = \{Circle, Triangle, Square, X\}$.

Each element in $C_P$ will be appear in each suppose except for:

1. The suppose of the label the case $C1$ is classified as.

Somewhat similarly, each element in $C_N$ will appear in each suppose except for:

1. The suppose of the label the case $C1$ is classified as.
2. The suppose of the label equal to the $C_N$ element's label.

From this, we can derive the formula for the total (summed sizes) of the dissimilarity sets in terms of the contributions of $C_P$ and $C_N$:

$$Total = (|L| - 2) \times |C_N| + (|L| - 1) \times |C_P|$$

We can represent $|C_P|$ in terms of $C_N$:

$$|C_P| = k - |C_N|$$

where $k$ is the length of the NNs set (essentially a constant for the current state of the case-base). Incorporating this with the previous equation, we get:

$$\begin{aligned} Total &= (|L| - 2) \times |C_N| + (|L| - 1) \times (k - |C_N|) \\ &= |L| \times |C_N| - 2 \times |C_N| + |L| \times k - |L| \times |C_N| - k + |C_N| \\ &= (|L| - 1) \times k - |C_N| \end{aligned}$$

The $(|L| - 1) \times k$ part will be a constant for a single case-base state. $|C_N|$ represents the number of disagreeing neighbours. A high number of disagreeing neighbours would indicate that the case-base is relatively uncertain about the classification. Thus - a strategy to semi-emulate uncertainty sampling (though in a less fine-grained manner) is to maximize $|C_N|$. To do this, we should minimize $Total$.

**Local Liability (Counting)**

There are two types of cases which can appear in the new case's liability set:

1. A case which was already misclassified, and the new case simply joins in in the misclassification.
2. A case which has enough sway with an existing case to cause the case to go from correctly classified, to incorrectly classified.

In each, the new case is somehow aiding in the incorrect classification of an existing case in the case-base. If we forget about the possibility of mislabelled data, the intuition is that this is generally a bad thing - we are 'overpowering' a neighbourhood of the case-base with data which could cause mislabelling.

Totalling here, as in other strategies, is probably not the best approach - but as a simplistic initial investigatory strategy, we try to minimize the total of the new case's liability set summed over the different possible supposes.

### 5.4.2 Pruned Cross Product Strategies

Understandably, given the 6,528 strategies possible through the simple cross-product grammar based approach presented, it was necessary to prune the combinations to a more tractable level. As such - the following restrictions on decision point combinations were made:

- No Global Strategies:

  Global strategies require more careful thought than local strategies given the possibility of shunting and flipping, and dealing with many cases' sets as opposed to just the candidate case. Certain measures, such as all-pairs similarity, also would need tweaking. As a result, we decided not to include global strategies in our simple cross-product style investigation.

- Only Minimization Strategies:

  Strategies which were initially thought out were all minimization strategies to try to encourage some kind of diversity. To cut down largely the number of possibilities given the somewhat limited compute-time available, we only chose minimization cross-product based strategies.

- Disallow Averaged Counting

  Averaging makes sense for the similarity based set scores, but for counting, it would lead to more of a boolean 0 or 1 defining if the set was empty or not. While this might well lead to valid strategies in some cases, we wanted to reason more about the contents of the sets as opposed to simply if they contained elements or not.

- Use Sparsity as the only density based measure:

  Due to our decision to only do minimization strategies, we do not use density in our measures, since we want to pick from dense regions, thus minimize sparsity.

- Exclude Dissimilarity:

  From initial investigation, it was noted that our simple dissimilarity strategies performed identically or near-identically to the corresponding reachability strategies. This is relatively unsurprising given the definition of dissimilarity, and given that we perform the calculation for each possible label, with each getting equal weighting.

- Restrict hybrid strategies to well performing single-set strategies:

  Hybrid strategies are a large reason for the high number of cross-product based strategies. We thus limit our hybrid strategies (the strategies with more than one RCDL set considered) to a small selection of well-performing single-set strategies. The strategies we picked:

- Total Direct Similarity (With Sparsity) Cross-Label Total Minimization
- Total All-Pairs Similarity (incl source) (With Sparsity) Cross-Label Deviation Minimization
- Total Counting (With Sparsity) Cross-Label Total Minimization
- Total Counting Cross-Label Total Minimization

These restrictions lead to a somewhat more manageable total of 100 cross-product based strategies.

### 5.4.3 Global Counting-Based Strategies

#### Global Coverage (Counting)

**Direct** The direct rNN's coverage added is equivalent to the new case's reachability added. See Section 5.4.1 for justification on minimization.

**Shunted** When a case is shunted from an NNs set by the new case, its coverage set size might be reduced by one (i.e. a coverage set removal). Many such removals (at least in the situation of a relatively populated case-base) would indicate that the new case is making an area more dense.

**Flip** If the new case causes a 'flip' in the correct-classification-status of an rNN, that rNN's other NNs may have gained or lost (or neither) items in their coverage sets. The desire would be to minimize flips to incorrect classifications (Flip related coverage removals), but increase flips to correct classification (Flip related coverage additions).

Because the overall aim in the strategy is 'minimization', to give positive contribution to Flip related coverage additions, and negative contribution to Flip related coverage removals, we subtract Flip related coverage additions, and add Flip related coverage removals.

#### Overall Formula

$$T = Direct_{C+} + Shunted_{C-} - Flip_{C+} + Flip_{C-}$$

This value is computed for each candidate case, summed across the possible labels. The candidate case with the minimum value is selected.

#### Global Reachability (Counting)

**Direct** A high number of additions to Direct reachability sets of Other cases would indicate that the new case is in a relatively dense region. To attempt to increase the diversity of new cases, we would thus like to minimize this.

**Shunted** Shunting a 'helping' neighbour for an existing case could be considered a bad thing, since we are moving in a neighbour for which an existing neighbour was already covering that space. We would like to minimize these.

**Flip** Causing a flip to correctly classified is a positive thing (Flip related reachability additions), whereas causing a flip to incorrectly classified a bad (Flip related reachability removals).

**Overall Formula**

$$T = Direct_{R+} + Shunted_{R-} + Flip_{R-} - Flip_{R+}$$

We would like to minimize this value overall.

**Global Liability (Counting)**

**Direct** (No real point considering this with the whole double counting that cropped up in CompStrat 4.)

**Shunted** Causing a 'shunt' of a hurtful case could be considered a positive action. The added case that caused the shunt will hopefully add more information regarding that area of the case space.

**Flip** Causing a case to flip to incorrectly classified would lead to flip related liability additions, whereas causing a flip to correctly classified would lead to flip related liability removals. We would like to minimize flips to incorrectly classified, while maximizing flips to correctly classified.

**Overall Formula**

$$T = Shunted_{L-} + Flip_{L-} - Flip_{L+}$$

This value is computed for each candidate case, summed across the possible labels. The candidate case with the maximum value is selected.

**Global Dissimilarity (Counting)**

**Direct** A high number of additions to Direct dissimilarity sets of Other cases would indicate that the new case is in a relatively dense region. To attempt to increase the diversity of new cases, we would thus like to minimize this.

**Shunted** To also aid in identifying diverse cases, we minimize the shunted neighbours here.

**Flip** Causing a flip to correct classification (Flip related dissimilarity removals) could be construed as positive, whereas flips to incorrect classification (Flip related dissimilarity additions) as negative.

**Overall Formula**

$$T = Direct_{D+} + Shunted_{D-} + Flip_{D+} - Flip_{D-}$$

### 5.4.4 Global Similarity-Based Strategies

We also briefly experimented with global similarity-based strategies. We used direct similarity instead of counting, and applied the same formulas as defined for the global counting-based strategies, with the exceptions that for coverage and reachability, we subtract instead of add shunted.

# Chapter 6

# Experimental Analysis

## 6.1 Datasets Used

**Non-Textual**

| Dataset | Num Instances | Num Labels | Label Distribution |
|---|---|---|---|
| iris [26] | 150 | 3 | 0.33 / 0.33 / 0.33 |
| wine [27] | 178 | 3 | 0.40 / 0.33 / 0.27 |
| zoo [28] | 101 | 7 | 0.41 / 0.20 / 0.13 / 0.10 / 0.08 / 0.05 / 0.04 |
| dermatology [29] | 366 | 6 | 0.31 / 0.20 / 0.17 / 0.14 / 0.13 / 0.05 |
| hepatitis [30] | 155 | 2 | 0.79 / 0.21 |
| breathalyzer [31] | 127 | 2 | 0.61 / 0.39 |
| haberman [32] | 306 | 2 | 0.74 / 0.26 |
| glass [33] | 214 | 6 | 0.36 / 0.33 / 0.14 / 0.08 / 0.06 / 0.04 |
| tae [34] | 151 | 3 | 0.34 / 0.33 / 0.32 |
| heartdisease [35] | 303 | 2 | 0.54 / 0.46 |
| lungcancer [36] | 32 | 3 | 0.41 / 0.31 / 0.28 |

**Textual**

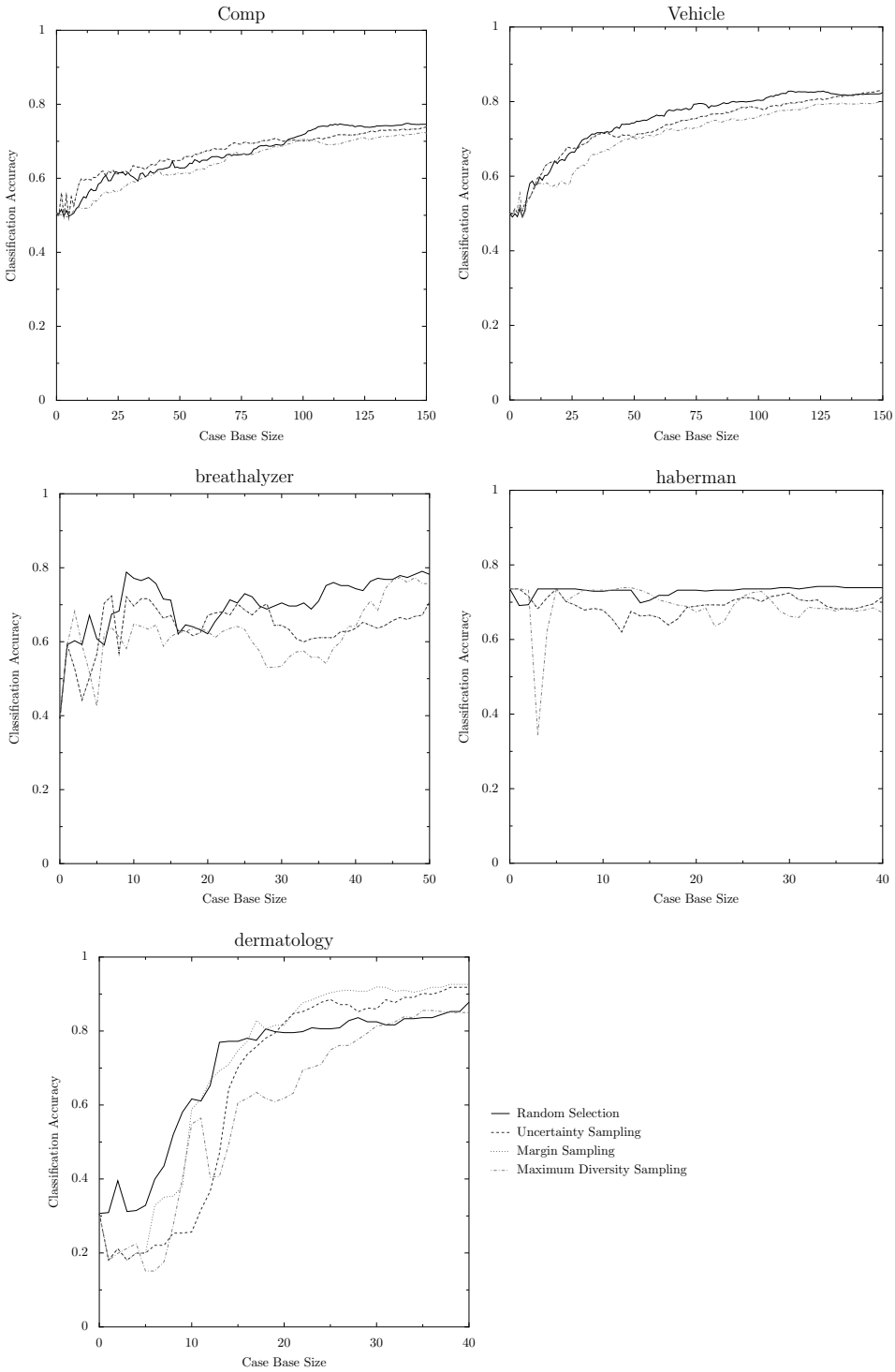
| Dataset | Num Instances | Num Labels | Label Distribution |
|---|---|---|---|
| WinXwin [37] | 1973 | 2 | 0.50 / 0.50 |
| Comp [38] | 1945 | 2 | 0.50 / 0.50 |
| Talk [39] | 1427 | 2 | 0.56 / 0.44 |
| Vehicle [40] | 1986 | 2 | 0.50 / 0.50 |

## 6.2 Results

Due to the large quantity of datasets experimented on, along with the sheer volume of selection strategies attempted, we do not present the results in entirety here. The reader can view these in the appendix, in Section 8.2.

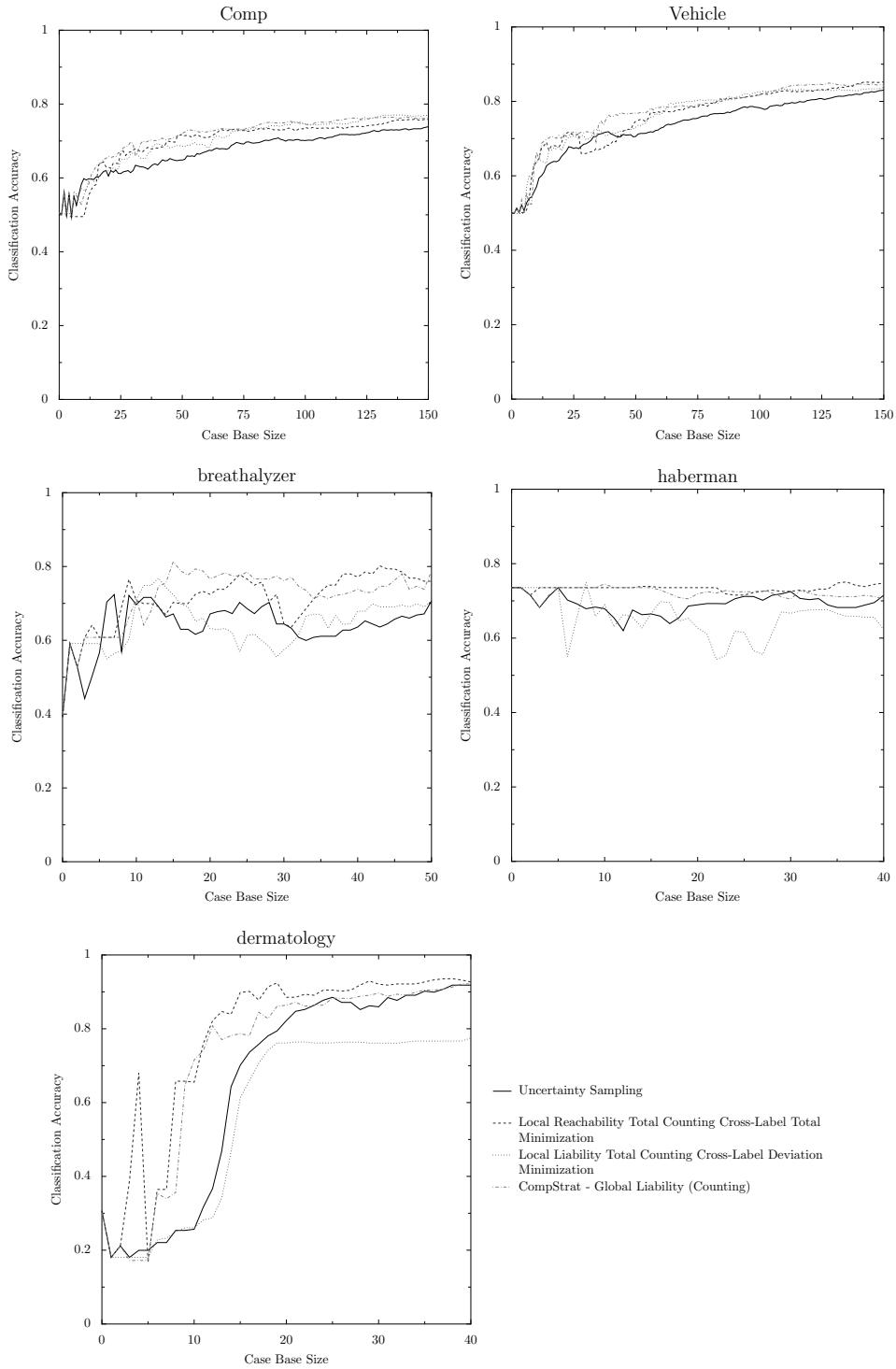
In each category, we choose a small subset of the noteworthy results, comparing against against some previously presented well-performing strategy to aid in comparison.

### 6.2.1 Baseline Comparison



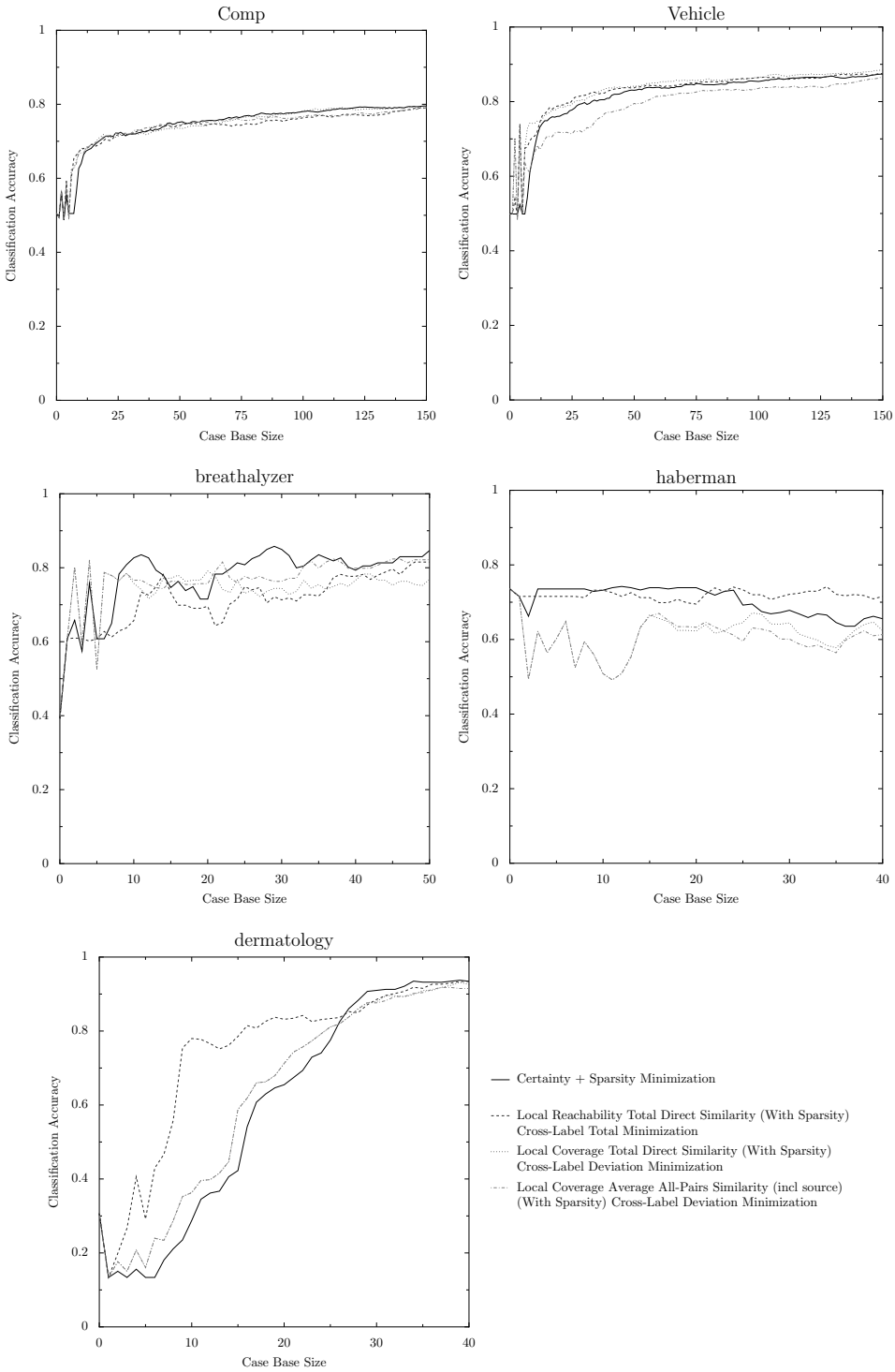

| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Random Selection | 99.852 (2) | 112.572 (1) | 35.225 (1) | 29.227 (1) | 27.996 (2) | 1.40 |
| Uncertainty Sampling | 100.797 (1) | 110.307 (2) | 32.071 (2) | 27.652 (2) | 25.909 (3) | 2.00 |
| Margin Sampling | 100.797 (1) | 110.307 (2) | 32.071 (2) | 27.652 (2) | 28.257 (1) | 1.60 |
| Maximum Diversity Sampling | 96.559 (3) | 106.469 (3) | 31.155 (3) | 27.606 (3) | 23.648 (4) | 3.20 |

### 6.2.2 Competence Based Counting



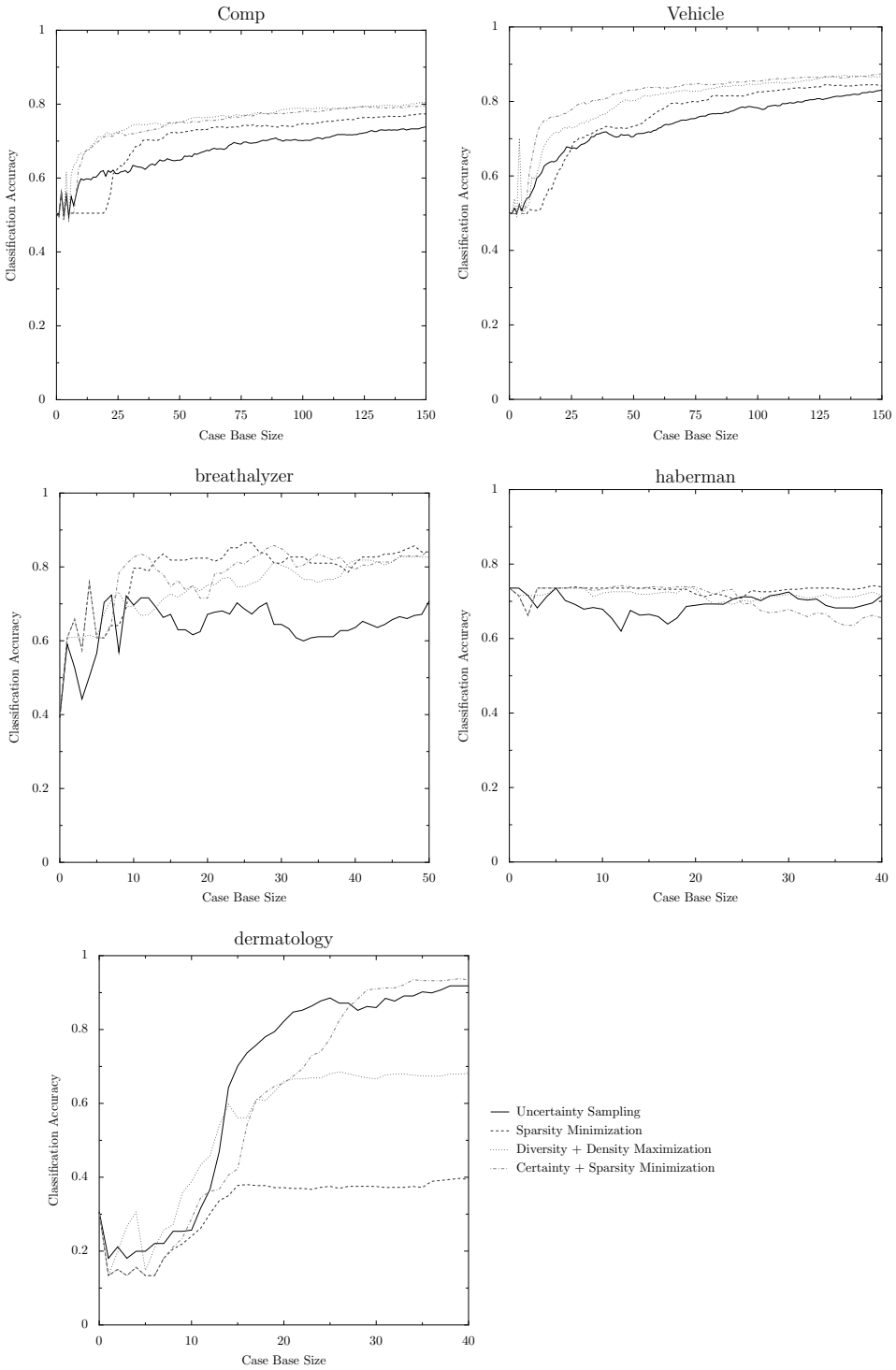

| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Uncertainty Sampling | 100.797 (4) | 110.307 (4) | 32.071 (4) | 27.652 (3) | 25.909 (3) | 3.60 |
| Local Reachability Total Counting Cross-Label Total Minimization | 104.740 (3) | 114.028 (3) | 35.627 (2) | 29.329 (1) | 31.015 (1) | 2.00 |
| Local Liability Total Counting Cross-Label Deviation Minimization | 105.242 (2) | 114.542 (2) | 32.264 (3) | 26.230 (4) | 22.811 (4) | 3.00 |
| CompStrat - Global Liability (Counting) | 107.075 (1) | 116.088 (1) | 36.040 (1) | 28.902 (2) | 28.909 (2) | 1.40 |

### 6.2.3 Competence with Similarity



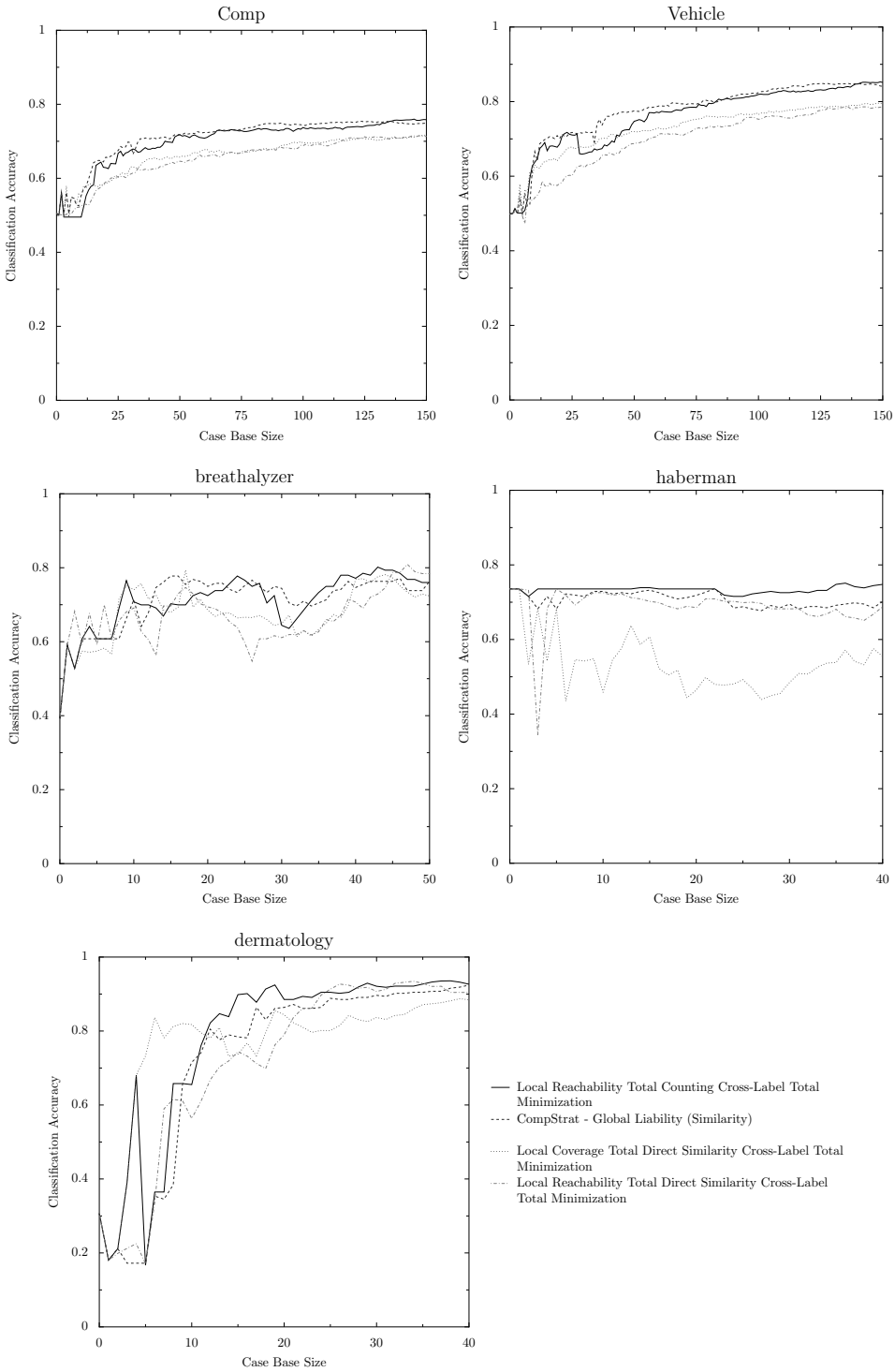

| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Certainty + Sparsity Minimization | 111.804 (1) | 122.310 (3) | 38.890 (1) | 28.218 (2) | 23.706 (4) | 2.20 |
| Local Reachability Total Direct Similarity (With Sparsity) Cross-Label Total Minimization | 110.739 (4) | 124.257 (2) | 35.671 (4) | 28.756 (1) | 29.583 (1) | 2.40 |
| Local Coverage Total Direct Similarity (With Sparsity) Cross-Label Deviation Minimization | 111.801 (2) | 125.126 (1) | 37.214 (3) | 24.467 (3) | 24.825 (2) | 2.20 |
| Local Coverage Average All-Pairs Similarity (incl source) (With Sparsity) Cross-Label Deviation Minimization | 111.347 (3) | 118.792 (4) | 38.492 (2) | 24.106 (4) | 24.778 (3) | 3.20 |

### 6.2.4 Basic Density Incorporation



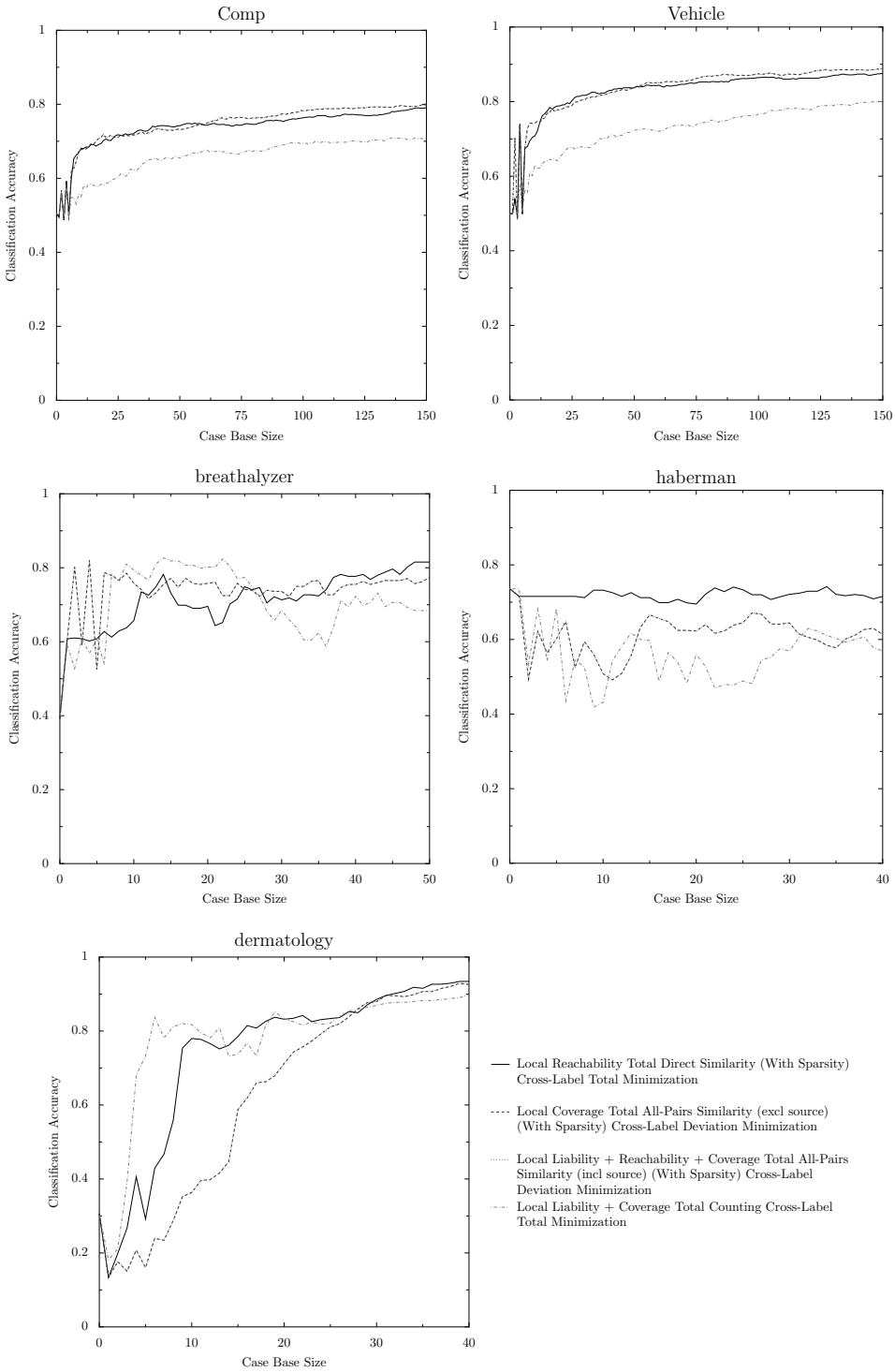

| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Uncertainty Sampling | 100.797 (4) | 110.307 (4) | 32.071 (4) | 27.652 (4) | 25.909 (1) | 3.40 |
| Sparsity Minimization | 105.268 (3) | 113.357 (3) | 39.388 (1) | 29.152 (1) | 12.706 (4) | 2.40 |
| Diversity + Density Maximization | 113.153 (1) | 119.469 (2) | 37.204 (3) | 28.699 (2) | 21.511 (3) | 2.20 |
| Certainty + Sparsity Minimization | 111.804 (2) | 122.310 (1) | 38.890 (2) | 28.218 (3) | 23.706 (2) | 2.00 |

### 6.2.5 Competence with Sparsity




| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Local Reachability Total Counting Cross-Label Total Minimization | 104.740 (2) | 114.028 (2) | 35.627 (2) | 29.329 (1) | 31.015 (1) | 1.60 |
| CompStrat - Global Liability (Similarity) | 106.603 (1) | 116.488 (1) | 35.632 (1) | 28.261 (2) | 29.027 (3) | 1.60 |
| Local Coverage Total Direct Similarity Cross-Label Total Minimization | 99.202 (3) | 109.113 (3) | 33.954 (3) | 21.317 (4) | 30.660 (2) | 3.00 |
| Local Reachability Total Direct Similarity Cross-Label Total Minimization | 97.882 (4) | 104.894 (4) | 33.459 (4) | 27.487 (3) | 28.697 (4) | 3.80 |

### 6.2.6 Competence Set Hybrids




| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Local Reachability Total Direct Similarity (With Sparsity) Cross-Label Total Minimization | 110.739 (2) | 124.257 (2) | 35.671 (2) | 28.756 (1) | 29.583 (2) | 1.80 |
| Local Coverage Total All-Pairs Similarity (excl source) (With Sparsity) Cross-Label Deviation Minimization | 111.969 (1) | 125.481 (1) | 37.016 (1) | 24.419 (2) | 24.811 (3) | 1.60 |
| Local Liability + Reachability + Coverage Total All-Pairs Similarity (incl source) (With Sparsity) Cross-Label Deviation Minimization | 111.969 (1) | 125.481 (1) | 37.016 (1) | 24.419 (2) | 24.811 (3) | 1.60 |
| Local Liability + Coverage Total Counting Cross-Label Total Minimization | 98.942 (3) | 108.981 (3) | 35.427 (3) | 22.374 (3) | 31.052 (1) | 2.60 |

## 6.3 Results Summary

| | Comp | Vehicle | breathalyzer | haberman | dermatology | Avg. Rank |
|---|---|---|---|---|---|---|
| Random Selection | 99.852 (13) | 112.572 (12) | 35.225 (12) | 29.227 (2) | 27.996 (9) | 9.60 |
| Uncertainty Sampling | 100.797 (12) | 110.307 (13) | 32.071 (16) | 27.652 (9) | 25.909 (10) | 12.00 |
| Margin Sampling | 100.797 (12) | 110.307 (13) | 32.071 (16) | 27.652 (9) | 28.257 (8) | 11.60 |
| Maximum Diversity Sampling | 96.559 (17) | 106.469 (16) | 31.155 (17) | 27.606 (10) | 23.648 (15) | 15.00 |
| Sparsity Minimization | 105.268 (9) | 113.357 (11) | 39.388 (1) | 29.152 (3) | 12.706 (18) | 8.40 |
| Diversity + Density Maximization | 113.153 (1) | 119.469 (5) | 37.204 (5) | 28.699 (6) | 21.511 (17) | 6.80 |
| Certainty + Sparsity Minimization | 111.804 (3) | 122.310 (4) | 38.890 (2) | 28.218 (8) | 23.706 (14) | 6.20 |
| Local Reachability Total Counting Cross-Label Total Minimization | 104.740 (11) | 114.028 (10) | 35.627 (10) | 29.329 (1) | 31.015 (2) | 6.80 |
| Local Liability Total Counting Cross-Label Deviation Minimization | 105.242 (10) | 114.542 (9) | 32.264 (15) | 26.230 (12) | 22.811 (16) | 12.40 |
| CompStrat - Global Liability (Counting) | 107.075 (7) | 116.088 (8) | 36.040 (7) | 28.902 (4) | 28.909 (6) | 6.40 |
| Local Reachability Total Direct Similarity (With Sparsity) Cross-Label Total Minimization | 110.739 (6) | 124.257 (3) | 35.671 (8) | 28.756 (5) | 29.583 (4) | 5.20 |
| Local Coverage Total All-Pairs Similarity (excl source) (With Sparsity) Cross-Label Deviation Minimization | 111.969 (2) | 125.481 (1) | 37.016 (6) | 24.419 (14) | 24.811 (12) | 7.00 |
| Local Liability + Reachability + Coverage Total All-Pairs Similarity (incl source) (With Sparsity) Cross-Label Deviation Minimization | 111.969 (2) | 125.481 (1) | 37.016 (6) | 24.419 (14) | 24.811 (12) | 7.00 |
| Local Liability + Coverage Total Counting Cross-Label Total Minimization | 98.942 (15) | 108.981 (15) | 35.427 (11) | 22.374 (16) | 31.052 (1) | 11.60 |
| Local Coverage Total Direct Similarity (With Sparsity) Cross-Label Deviation Minimization | 111.801 (4) | 125.126 (2) | 37.214 (4) | 24.467 (13) | 24.825 (11) | 6.80 |
| Local Coverage Average All-Pairs Similarity (incl source) (With Sparsity) Cross-Label Deviation Minimization | 111.347 (5) | 118.792 (6) | 38.492 (3) | 24.106 (15) | 24.778 (13) | 8.40 |
| CompStrat - Global Liability (Similarity) | 106.603 (8) | 116.488 (7) | 35.632 (9) | 28.261 (7) | 29.027 (5) | 7.20 |
| Local Coverage Total Direct Similarity Cross-Label Total Minimization | 99.202 (14) | 109.113 (14) | 33.954 (13) | 21.317 (17) | 30.660 (3) | 12.20 |
| Local Reachability Total Direct Similarity Cross-Label Total Minimization | 97.882 (16) | 104.894 (17) | 33.459 (14) | 27.487 (11) | 28.697 (7) | 13.00 |

## 6.4 Results Discussions

It should be noted that the following discussion also incorporates the full results, as opposed to the small subset presented above.

### 6.4.1 Random Selection Performance

It bears mention that, considering how utterly inexpensive a strategy it is, random selection performs quite well.

### 6.4.2 Local Reachability Counting

Considering their simplicity, strategies involving local reachability counting perform very well compared to uncertainty and random sampling.

One possible reason for this is due to the strategies' benefit of creating a candidate set of relatively uncertain cases from which random selection is used. This is slightly different to uncertainty sampling based on non-weighted voting, in that firstly, voting is performed in a distance weighted fashion to determine the probable label, then the cardinality of the set of cases in the k-nearest neighbours with the same label is minimized.

In this way, it doesn't fall prey to the traditional downfall of pure uncertainty sampling of always querying outliers, instead picking from the relatively uncertain cases at random.

### 6.4.3 Benefit of addition of Sparsity

The addition of density information (in the form of sparsity), has an undeniable benefit to the competence-based strategies, especially on textual data. The benefit is less seen on the non-textual data, probably due to the quicker convergence and smaller data sizes.

It should be noted that Diversity + Sparsity, and Uncertainty + Sparsity, also do quite well. We decided to perform basic tests on these to determine if the increase in competence strategies' performance was at all significant. The competence strategies on their own generally aim to pick somewhat diverse cases, and thus adding density information intends to augment this diversity picking with information regarding density of that region in the dataset.

Some of the competence strategies augmented with sparsity do perform significantly better than either Diversity + Sparsity, or Uncertainty + Sparsity, on the textual datasets. Though given the naive approach

used for these two comparative strategies, further work is needed to compare with established strategies which aim to incorporate uncertainty, density, and diversity.

### 6.4.4 Deviation almost equivalent results to total

Although deviation and totalling are technically different approaches, for the datasets tested and the manner in which we used these approaches in our selection strategies, they produced for the most part identical performance.

### 6.4.5 Poor hybrid results

Upon consultation of the full results, and as demonstrated in the subset of the results above, it can be seen that the hybrid strategies in our simple combinations appear to be dominated by one of the sets, being identical to that one set's non-hybrid strategy.

# Chapter 7

# Conclusions and Future Work

## 7.1 Project Results

The project has led to some promising competence-based selection strategies which perform quite well compared to traditional strategies. While the results vary between datasets, there being no single overall winner, they do prove somewhat that incorporating competence in active learning, as opposed to its traditional domain of case-base maintenance, might have some merit. Though certainly, further work would be needed to establish this more concretely.

## 7.2 Contributions Made

### 7.2.1 Experimental Platform

The Python-based platform developed for the project has some useful features for selection-strategy experimentation in a pool-based active learning context.

It allows pluggable definitions of selection strategies, without requiring code changes to the main platform. It also has very pluggable definitions for things like the oracle, stopping condition, classifier, etc. It could thus be used for evaluations far beyond the current scope of this project.

One small but useful additional capability of the platform is how it deals with data loading and distance computation. It provides two interoperable formats to store pre-computed distance information along with the data. This allows for very different platforms to be used in initially processing the data. Even in this project, it allowed textual data to be added by using the facilities of SciKit Learn, but required no modification to the platform code.

The experimental result re-use mechanism is another useful feature to the platform, which allows it to read in previous results, and use them if necessary when running the new experiment.

The capability of the platform to operate on a distributed cluster-based map-reduce fashion is another significant, albeit a relatively undocumented benefit given the late stage of the project that the feature was added.

### 7.2.2 Incremental Algorithm to RCDL

The incremental algorithm to RCDL profile generation presented in Section 4.6 is a new addition to the area. It was not necessary in Delany [25] due to how she was using it, but for active learning approaches,

having an incremental approach becomes a lot more relevant.

The 'suppose' operation is also somewhat novel, allowing selection strategies to determine in advance what the changes to the RCDL profiles would be if it picked a certain case, and that case turned out to have a certain label.

### 7.2.3 Competence Based Selection Strategies

We presented how we interpret the changes that might occur the competence profiles of cases following the addition of a new case to the case-base, and ran experimentation to test our derivations.

While the presented competence based strategies were by no means ground-breaking in their results, I believe their investigation still contributed useful thought-experiments in the field, and also set some of the ground work for potential future investigation into competence based methodologies.

### 7.2.4 SciKit-Learn Fix Contribution

Although relatively trivial in size, I identified a bug in the SciKit-Learn 20 newsgroups textual data loading mechanism. I fixed the bug, added unit tests to verify the presence of the bug, and the fix written, and subsequently pushed the changes back to the main SciKit-Learn repository. The community members in charge of the repository verified and accepted the change.

## 7.3 Further Research Opportunities

### 7.3.1 Statistical Analysis of Suppose Change Distribution

In our research, we treated the selection strategies and overall experimentation as simple black boxes. We would try implementing various strategies, run the experiments with the new strategies, and simply look at the outputted results, comparing them to those of previous strategies.

At no stage did we actually analyse the individual choices that the strategies made for the purposes of understanding the results or improving the strategies. While much scope exists for more in depth analysis, at a very minimum, one interesting area for future work would be to perform statistical analysis on supposes for multiple datasets to better understand distribution of types of case profile changes.

When we were selecting minimums, I am curious as to the level of zeroes present, whether for some of the strategies on certain datasets, they were almost equivalent in behaviour to random sampling. I also think it would be useful for when developing new strategies as to the frequency and size of the different types of RCDL changes from the selection of a case.

As a result of all this, I believe there would definitely be benefit in spending the time to perform statistical analysis on suppose operation based changes.

### 7.3.2 Try more sophisticated RCDL-augmented selection strategies

The work we did on building RCDL based selection-strategies was somewhat exploratory, focussing mainly on just the RCDL concepts. It would be interesting to try more sophisticated techniques in using RCDL profiles, by augmenting some existing selection strategies, both established and novel.

It might also be interesting to go in the other direction - to try to incorporate concepts from existing strategies into the simple RCDL based strategies we defined, for example taking into account case uncertainty and label probability when doing the suppose operations.

### 7.3.3 Use 'suppose' approach in developing case-base maintenance algorithms

Delany [25] investigated case-base maintenance algorithms using RCDL profiles in a compute-once fashion, generating the profiles at the outset of the algorithm and subsequently just using this snapshot for the remainder of the algorithm.

It would be interesting to try using an incremental "suppose" approach similar to what we developed in investigating case-base maintenance algorithms.

### 7.3.4 Improve Visualization Capabilities

The visualization capabilities described in Section 3.4.3 were a two day effort in removing some of the 'black-box' nature of our experimentation approach. This feature has scope for massive improvements:

- **Provide dynamic interactive view with overlay information**:

  Currently, only a static output is produced following the experiment. I believe it would be useful to have a more dynamic output option which includes the ability for user interaction, providing for example:

  - Overlay information about the relevant cases
  - Information regarding the state of the case-base
  - The decision process surrounding the current

- **Allow querying the engine**:

  As opposed to just viewing information regarding the current state of the experiment, it would be useful to be able to perform queries against the engine such as "What effects would selecting this case from the unlabelled set?". It could also be useful to actually have input into the experiment by being able to actually force the selection of a case to see the consequences as the experiment continues running.

I believe that by researching and implementing a generic visualization feature for the platform, the task of selection-strategy research and development would be eased by allowing greater visibility into the decisions of selection strategies.

The reason I would count this as a research opportunity as opposed to pure development is that investigation around how to effectively convey information, and provide interactivity to the user would need to be carried out.

### 7.3.5 Try alteration on definition of Delany's 'Contributes' notion

Delany's definition of 'contribution' is as one would expect for binary classification, but I feel is somewhat counter intuitive for her 'negative' sets (liability and dissimilarity) when performing multi-label classification.

If a case is mislabelled by the case-base, she defines a case contributing to that misclassification as any neighbour in the k-nearest neighbours of the original case that have a different label to the label of the original case.

In the example presented in Figure 4.1, even though $C1$ plays no role in the classification of $X$, by Delany's definition, it still contributes to the incorrect classification, and thus $Misclassifies(X,\ C1)$. This has implications in the liability and dissimilarity sets of $C1$ and $X$ respectively.

I believe however that it would be more intuitive if we defined contribution towards classification as the neighbours in a case's k-nearest neighbours that have the same label as the classification of the case.

### 7.3.6 Maintain information about 'helping' and 'hindering' sets

A case either has a reachability set, or dissimilarity set - never both. This either-or nature is convenient for Delany's binary-style analysis, but throws away all information about the *competing* set (be it reachability, or dissimilarity).

For example, in Figure 5.1:

- $reachability(Cn) = \{C1,\ C2\}$
- $dissimilarity(Cn) = \{\}$

Here, no knowledge of $C3$ is maintained. I believe it might be useful to also maintain additional sets 'helping' and 'hindering', along with their duals.

### 7.3.7 Reconsider Liability Strategies

The liability-based strategies perform quite poorly overall. Some further consideration and investigation is deserved around the causes.

### 7.3.8 Try more hybrid strategies

Our investigation into hybrid RCDL set strategies was very limited. Considering different combinations in a more meaningful and systematic way could prove interesting.

### 7.3.9 Incorporate certain aspects of global strategies

Certain global-type changes, in particular 'flips', are quite interesting, and much underused in our work. Selective incorporation of side-effect style information could allow some interesting variations on some of the strategies we defined, and also more traditional strategies.

### 7.3.10 Profile datasets, applying appropriate strategies

Analysis on which strategies perform well based on dataset characteristics could be useful. Certain types of strategies perform very well on some datasets, but only mediocre on others. For example, sparsity based strategies perform well on the textual data.

Trying to dynamically pick which strategy to apply based on the current case-base, and the dataset in general might be promising.

## 7.4 Platform Development Opportunities

### 7.4.1 Improve Test Coverage

Test coverage of the platform, and experimentation in general, was relatively low. For certain core operations, such as classification and incremental profile building, unit tests were written, but large portions of the overall code have not been methodically tested.

If the platform were to be used in a more intense fashion, increasing the test coverage should be a work item.

### 7.4.2 Document and package platform for general release

Given the one-person nature of the development of the platform, along with the focus on the research and results of selection strategies, the code of the platform is severely lacking in documentation.

To be more usable as a general purpose platform that others could easily also use, it would be good to properly document the code, and restructure the code to a more disciplined package.

### 7.4.3 Change to support other types of case-based reasoning experimentation

The current platform was developed for the sole purpose of investigating selection strategies in the context of pool-based active learning. A lot of the code however would also be applicable if one were performing other types of case-based reasoning research.

At a minimum, it would be nice if basic support was added for other core scenarios, such as case-base maintenance. This would allow researchers to build on top of this the elements that are unique to their investigations.

## 7.5 Closing Thoughts

### 7.5.1 Backing Source Code should follow publications

I believe that, at a minimum for any public funded research, a link to backing source code should be a requirement for any publications produced. I find it baffling that results can be presented without any capability for reproducing the experiment. It feels contra to the general purpose of public research.

It is possible that small bugs in people's code could invalidate entire experiments. Understandably, there would be fear when releasing source code as a result, but the work feels somewhat pointless otherwise. This is assuming good intention, but there is also the fact that researchers can carefully choose which results to publish, ignoring all the failed results.

Not only would the requirement of source code provision aid in preventing the above, it would also better encourage re-use and collaboration among researchers of similar areas.

### 7.5.2 Usefulness of active learning and case-based reasoning

The project has opened my eyes somewhat to the benefit of active learning. In particular, the fact that the concepts apply beyond traditional domains like text analysis, and into more modern ones like recommendation engines.

I don't think active learning is the 'next-big-thing', but the concepts in it are undeniably useful, and it's an area that I believe active research will continue to be beneficial. I will be especially interested in seeing in the coming years the new areas in which the concepts get successfully applied to.

### 7.5.3 Novelty of Competence Models

The notion of competence models appears to be a relatively unexplored area in case-based reasoning. The profiling of individual cases based on their competence-contributions to a case-base provides a different mindset from which to approach the domain.

Somewhat pessimistically, I don't believe competence oriented research alone will suddenly solve all the woes of active learning, or general case-based reasoning. I do however think it is a useful addition to the general toolbox of techniques within the case-based reasoning community, and I'm shamelessly glad to have contributed ever-so slightly to it.

# Chapter 8

# Appendix

## 8.1 Project Source Code

This Project's source code has been placed online [41]. Aside from the aim for this project to be as Open Source as possible, it is my strong belief that any research which actually aims to advance a given field should have all the things required to reproduce the results publicly available without having to make explicit requests to the authors.

It surprises me the massive body of research that doesn't follow this methodology - especially research which is publicly funded. It is important that external parties have full opportunities to validate experiment results. Small bugs in programs may cause entire projects to have invalid (though convincing) results, and while this would be disastrous for the researchers involved to have these discovered publicly, it is necessary for the actual progression of scientific knowledge.

By having all backing materials available, it also allows other researchers to continue with the original research, as opposed to starting from scratch and re-developing all the architecture necessary again and again.

## 8.2 Full Experiment Results Listing

Due to the incorporation of a map-reduce framework towards the end of the project, along with scripts to execute across several hundred college computers, throughput was available to run many additional experiments. Due to the limited quantity of time remaining at the time, a 'dumb' cross-product methodology was used to generate experiments based on the methods used previously.

Only a small proportion of these new generated experiments were some way promising, but for completeness, the full listing of experiments along with their results are presented here. Due to the cross-product style strategy name length, these names have been abbreviated, but due to their resulting unintelligibleness, the reader is recommended to instead consult the results online [41].

### 8.2.1 Non-Textual

| | iris | wine | zoo | dermatology | hepatitis | breathalyzer | haberman | glass | tae | heartdisease | lungcancer | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Random Selection | 12.157 (37) | 16.774 (45) | 13.909 (40) | 27.996 (20) | 7.517 (8) | 35.225 (26) | 29.227 (5) | 18.850 (3) | 11.429 (9) | 20.149 (33) | 2.897 (34) | 23.64 |
| Uncertainty Sampling | 14.810 (16) | 18.421 (37) | 19.035 (7) | 25.909 (26) | 5.942 (26) | 32.071 (53) | 27.652 (26) | 17.735 (7) | 11.274 (11) | 22.538 (7) | 4.220 (6) | 20.18 |
| Margin Sampling | 13.557 (20) | 18.219 (38) | 18.810 (9) | 28.257 (19) | 5.942 (26) | 32.071 (53) | 27.652 (26) | 17.858 (6) | 11.658 (6) | 22.538 (7) | 4.153 (8) | 19.82 |
| Maximum Diversity Sampling | 15.130 (14) | 15.818 (47) | 16.720 (32) | 23.648 (39) | 7.161 (20) | 31.155 (60) | 27.606 (27) | 16.745 (20) | 12.069 (1) | 20.835 (30) | 3.957 (14) | 27.64 |
| Sparsity Minimization | 6.667 (51) | 13.466 (54) | 10.078 (58) | 12.706 (64) | 7.670 (2) | 39.388 (3) | 29.152 (6) | 15.148 (38) | 9.432 (48) | 19.788 (34) | 3.017 (31) | 35.36 |
| Diversity + Density Maximization | 6.667 (51) | 15.768 (49) | 11.414 (47) | 21.511 (42) | 7.670 (2) | 37.204 (12) | 28.699 (11) | 16.085 (29) | 9.421 (49) | 21.284 (23) | 3.170 (28) | 31.18 |
| Certainty + Sparsity Minimization | 6.667 (51) | 17.353 (41) | 11.162 (50) | 23.706 (37) | 7.670 (2) | 38.890 (4) | 28.218 (17) | 16.990 (18) | 10.755 (20) | 21.227 (25) | 3.003 (32) | 27.00 |
| CGCC | 12.780 (26) | 19.453 (21) | 16.440 (34) | 25.245 (27) | 7.603 (5) | 31.496 (58) | 29.021 (7) | 16.099 (28) | 11.692 (5) | 22.092 (18) | 3.497 (23) | 22.91 |
| CGRC | 13.780 (19) | 20.237 (2) | 18.340 (19) | 29.541 (15) | 6.342 (25) | 31.580 (57) | 28.586 (12) | 17.057 (15) | 10.505 (26) | 20.848 (29) | 3.640 (21) | 21.82 |
| CGLC | 12.547 (35) | 18.533 (35) | 15.342 (39) | 28.909 (17) | 6.944 (21) | 36.040 (18) | 28.902 (8) | 17.055 (16) | 11.208 (12) | 22.632 (5) | 3.577 (22) | 20.73 |
| CGDC | 8.987 (49) | 17.551 (40) | 11.092 (51) | 20.876 (45) | 7.480 (9) | 30.559 (64) | 29.408 (2) | 15.725 (34) | 10.892 (17) | 15.987 (55) | 3.810 (18) | 34.91 |
| CGCS | 13.527 (21) | 17.933 (39) | 18.844 (8) | 26.880 (22) | 7.413 (12) | 31.841 (54) | 16.762 (62) | 15.977 (33) | 9.935 (40) | 14.457 (61) | 4.703 (1) | 32.09 |
| CGRS | 12.613 (32) | 20.402 (1) | 18.579 (10) | 30.847 (3) | 3.573 (29) | 30.240 (67) | 28.181 (18) | 17.101 (13) | 10.462 (28) | 17.347 (54) | 3.657 (20) | 25.00 |
| CGLS | 12.547 (35) | 18.950 (34) | 16.049 (36) | 29.027 (16) | 6.928 (23) | 35.632 (21) | 28.261 (15) | 16.876 (19) | 11.025 (15) | 22.655 (4) | 3.577 (22) | 21.82 |
| CGDS | 10.150 (43) | 10.297 (59) | 11.165 (49) | 17.877 (53) | 7.410 (13) | 30.509 (65) | 27.429 (30) | 14.752 (45) | 11.767 (4) | 14.508 (59) | 4.183 (7) | 38.82 |
| LLAAPSisCLDM | 9.420 (48) | 16.966 (44) | 12.797 (43) | 19.938 (49) | 6.936 (22) | 33.058 (42) | 25.426 (39) | 16.458 (25) | 10.493 (27) | 22.368 (13) | 3.767 (19) | 33.73 |
| LLAAPSisCLTM | 11.473 (38) | 7.790 (66) | 8.465 (61) | 18.907 (51) | 7.480 (9) | 30.756 (62) | 29.516 (1) | 14.712 (46) | 11.047 (14) | 14.456 (62) | 3.947 (15) | 38.64 |
| LLAAPSisWSCLDM | 6.667 (51) | 15.043 (52) | 10.001 (59) | 14.464 (61) | 7.653 (3) | 39.705 (1) | 23.179 (51) | 14.673 (48) | 9.541 (46) | 21.109 (26) | 2.540 (41) | 39.91 |
| LLAAPSisWSCLTM | 6.667 (51) | 8.349 (64) | 9.976 (60) | 14.319 (63) | 7.670 (2) | 32.817 (46) | 28.049 (20) | 16.396 (26) | 10.058 (36) | 18.822 (43) | 2.313 (45) | 41.45 |
| LLADSCLDM | 11.243 (39) | 12.223 (56) | 12.289 (45) | 20.412 (47) | 5.394 (28) | 35.468 (24) | 25.547 (38) | 15.470 (35) | 10.619 (23) | 22.207 (15) | 3.920 (16) | 33.27 |
| LLADSCLTM | 10.053 (44) | 7.808 (65) | 10.707 (54) | 17.631 (54) | 7.480 (9) | 30.933 (61) | 28.242 (16) | 15.127 (39) | 10.439 (29) | 14.466 (60) | 3.947 (15) | 40.55 |
| LLADSWSCLDM | 6.667 (51) | 15.067 (51) | 10.001 (59) | 14.325 (62) | 7.653 (3) | 39.416 (2) | 24.757 (42) | 14.989 (41) | 9.669 (45) | 20.951 (27) | 2.390 (42) | 38.64 |
| LLADSWSCLTM | 6.667 (51) | 8.370 (63) | 9.976 (60) | 14.738 (60) | 7.670 (2) | 33.091 (41) | 27.798 (24) | 17.012 (17) | 9.696 (44) | 18.473 (49) | 2.380 (43) | 41.27 |
| LLAAPSesCLDM | 9.420 (48) | 16.966 (44) | 12.797 (43) | 19.938 (49) | 6.936 (22) | 33.058 (42) | 25.426 (39) | 16.458 (25) | 10.493 (27) | 22.368 (13) | 3.767 (19) | 33.73 |
| LLAAPSesCLTM | 11.473 (38) | 7.790 (66) | 8.465 (61) | 18.907 (51) | 7.480 (9) | 30.756 (62) | 29.516 (1) | 14.712 (46) | 11.047 (14) | 14.456 (62) | 3.947 (15) | 38.64 |
| LLAAPSesWSCLDM | 6.667 (51) | 15.043 (52) | 10.001 (59) | 14.464 (61) | 7.653 (3) | 39.705 (1) | 23.179 (51) | 14.673 (48) | 9.541 (46) | 21.109 (26) | 2.540 (41) | 39.91 |
| LLAAPSesWSCLTM | 6.667 (51) | 8.349 (64) | 9.976 (60) | 14.319 (63) | 7.670 (2) | 32.817 (46) | 28.049 (20) | 16.396 (26) | 10.058 (36) | 18.822 (43) | 2.313 (45) | 41.45 |
| LLTAPSisCLDM | 10.613 (41) | 10.318 (58) | 10.596 (56) | 21.498 (43) | 7.329 (16) | 32.300 (50) | 24.180 (48) | 14.705 (47) | 10.673 (22) | 15.034 (57) | 4.107 (10) | 40.73 |
| LLTAPSisCLTM | 10.647 (40) | 10.060 (60) | 10.785 (53) | 19.795 (50) | 7.420 (11) | 31.695 (56) | 26.559 (35) | 14.932 (44) | 11.952 (3) | 14.538 (58) | 4.113 (9) | 38.09 |
| LLTAPSisWSCLDM | 6.667 (51) | 7.664 (68) | 9.976 (60) | 16.834 (56) | 7.680 (1) | 31.370 (59) | 25.136 (40) | 14.943 (43) | 10.204 (34) | 18.692 (47) | 2.680 (40) | 45.36 |
| LLTAPSisWSCLTM | 6.667 (51) | 10.417 (57) | 9.976 (60) | 15.852 (58) | 7.670 (2) | 32.640 (48) | 27.963 (22) | 16.055 (30) | 10.033 (37) | 18.405 (50) | 2.130 (47) | 42.00 |
| LLTCCLDM | 9.847 (45) | 18.456 (36) | 11.227 (48) | 22.811 (41) | 7.590 (7) | 32.264 (51) | 26.230 (37) | 15.458 (36) | 10.572 (24) | 22.516 (8) | 4.020 (12) | 31.36 |
| LLTCCLTM | 8.987 (49) | 15.785 (48) | 11.092 (51) | 21.037 (44) | 7.480 (9) | 30.337 (66) | 29.323 (4) | 16.031 (31) | 11.105 (13) | 15.977 (56) | 3.870 (17) | 35.27 |
| LLTCWSCLDM | 6.667 (51) | 12.573 (55) | 9.976 (60) | 17.285 (55) | 7.597 (6) | 37.648 (10) | 24.558 (43) | 14.482 (51) | 9.408 (50) | 20.897 (28) | 2.857 (35) | 40.36 |
| LLTCWSCLTM | 6.667 (51) | 9.767 (61) | 9.976 (60) | 17.285 (55) | 7.670 (2) | 36.000 (19) | 27.091 (32) | 14.474 (52) | 10.201 (35) | 19.288 (36) | 2.263 (46) | 40.82 |
| LLTDSCLDM | 9.700 (47) | 17.162 (43) | 6.815 (62) | 18.701 (52) | 5.473 (27) | 34.343 (30) | 24.927 (41) | 14.553 (49) | 10.674 (21) | 22.577 (6) | 4.013 (13) | 35.55 |
| LLTDSCLTM | 9.740 (46) | 8.936 (62) | 10.985 (52) | 20.058 (48) | 7.420 (11) | 31.829 (55) | 26.559 (35) | 15.036 (40) | 11.999 (2) | 14.538 (58) | 4.093 (11) | 38.18 |
| LLTDSWSCLDM | 6.667 (51) | 7.679 (67) | 9.976 (60) | 16.834 (56) | 7.597 (6) | 32.678 (47) | 21.809 (57) | 14.971 (42) | 10.323 (31) | 21.691 (21) | 3.153 (29) | 42.45 |
| LLTDSWSCLTM | 6.667 (51) | 10.417 (57) | 9.976 (60) | 15.852 (58) | 7.670 (2) | 32.594 (49) | 27.979 (21) | 16.011 (32) | 9.966 (39) | 18.405 (50) | 2.130 (47) | 42.36 |
| LLTAPSesCLDM | 10.613 (41) | 10.318 (58) | 10.596 (56) | 21.498 (43) | 7.329 (16) | 32.300 (50) | 24.180 (48) | 14.705 (47) | 10.673 (22) | 15.034 (57) | 4.107 (10) | 40.73 |
| LLTAPSesCLTM | 10.647 (40) | 10.060 (60) | 10.785 (53) | 19.795 (50) | 7.420 (11) | 31.695 (56) | 26.559 (35) | 14.932 (44) | 11.952 (3) | 14.538 (58) | 4.113 (9) | 38.09 |
| LLTAPSesWSCLDM | 6.667 (51) | 7.664 (68) | 9.976 (60) | 16.834 (56) | 7.680 (1) | 31.370 (59) | 25.136 (40) | 14.943 (43) | 10.204 (34) | 18.692 (47) | 2.680 (40) | 45.36 |
| LLTAPSesWSCLTM | 6.667 (51) | 10.417 (57) | 9.976 (60) | 15.852 (58) | 7.670 (2) | 32.640 (48) | 27.963 (22) | 16.055 (30) | 10.033 (37) | 18.405 (50) | 2.130 (47) | 42.00 |
| LRAAPSisCLDM | 15.310 (11) | 19.757 (12) | 17.978 (21) | 23.056 (40) | 7.433 (10) | 30.671 (63) | 26.452 (36) | 17.085 (14) | 10.867 (18) | 22.111 (17) | 4.610 (3) | 22.27 |

| | iris | wine | zoo | dermatology | hepatitis | breathalyzer | haberman | glass | tae | heartdisease | lungcancer | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LRAAPSisCLTM | 15.310 (11) | 19.757 (12) | 18.198 (20) | 23.056 (40) | 7.433 (10) | 30.671 (63) | 26.452 (36) | 17.085 (14) | 10.867 (18) | 22.111 (17) | 4.610 (3) | 22.18 |
| LRAAPSisWSCLDM | 6.667 (51) | 14.697 (53) | 10.092 (57) | 15.551 (59) | 7.670 (2) | 35.066 (27) | 28.845 (9) | 14.503 (50) | 8.565 (53) | 21.558 (22) | 2.850 (36) | 38.09 |
| LRAAPSisWSCLTM | 6.667 (51) | 16.159 (46) | 13.379 (41) | 20.663 (46) | 7.670 (2) | 34.445 (29) | 27.847 (23) | 15.157 (37) | 9.443 (47) | 21.901 (20) | 3.050 (30) | 33.82 |
| LRADSCLDM | 15.183 (13) | 19.396 (25) | 19.057 (5) | 27.126 (21) | 7.240 (19) | 33.387 (35) | 27.479 (29) | 17.418 (12) | 10.519 (25) | 22.879 (1) | 4.653 (2) | 17.00 |
| LRADSCLTM | 15.183 (13) | 19.396 (25) | 19.237 (1) | 27.126 (21) | 7.240 (19) | 33.387 (35) | 27.479 (29) | 17.418 (12) | 10.519 (25) | 22.879 (1) | 4.653 (2) | 16.64 |
| LRADSWSCLDM | 6.667 (51) | 15.571 (50) | 10.597 (55) | 16.651 (57) | 7.670 (2) | 35.554 (23) | 27.716 (25) | 14.432 (53) | 8.812 (52) | 22.378 (12) | 2.713 (39) | 38.09 |
| LRADSWSCLTM | 6.753 (50) | 19.460 (20) | 12.804 (42) | 24.184 (36) | 7.670 (2) | 34.811 (28) | 26.744 (33) | 16.350 (27) | 9.726 (43) | 22.482 (10) | 2.897 (34) | 29.55 |
| LRAAPSesCLDM | 15.310 (11) | 19.757 (12) | 17.978 (21) | 23.056 (40) | 7.433 (10) | 30.671 (63) | 26.452 (36) | 17.085 (14) | 10.867 (18) | 22.111 (17) | 4.610 (3) | 22.27 |
| LRAAPSesCLTM | 15.310 (11) | 19.757 (12) | 18.198 (20) | 23.056 (40) | 7.433 (10) | 30.671 (63) | 26.452 (36) | 17.085 (14) | 10.867 (18) | 22.111 (17) | 4.610 (3) | 22.18 |
| LRAAPSesWSCLDM | 6.667 (51) | 14.697 (53) | 10.092 (57) | 15.551 (59) | 7.670 (2) | 35.066 (27) | 28.845 (9) | 14.503 (50) | 8.565 (53) | 21.558 (22) | 2.850 (36) | 38.09 |
| LRAAPSesWSCLTM | 6.667 (51) | 16.159 (46) | 13.379 (41) | 20.663 (46) | 7.670 (2) | 34.445 (29) | 27.847 (23) | 15.157 (37) | 9.443 (47) | 21.901 (20) | 3.050 (30) | 33.82 |
| LRTAPSisCLDM | 14.910 (15) | 18.978 (33) | 19.055 (6) | 26.620 (23) | 7.407 (14) | 32.146 (52) | 26.649 (34) | 17.667 (8) | 11.294 (10) | 22.284 (14) | 4.370 (5) | 19.45 |
| LRTAPSisCLTM | 14.910 (15) | 18.978 (33) | 19.155 (2) | 26.620 (23) | 7.407 (14) | 32.146 (52) | 26.649 (34) | 17.667 (8) | 11.294 (10) | 22.284 (14) | 4.370 (5) | 19.09 |
| LRTAPSisWSCLDM | 13.037 (23) | 19.982 (7) | 15.452 (37) | 25.910 (25) | 7.670 (2) | 33.291 (37) | 27.228 (31) | 17.453 (11) | 9.931 (41) | 22.761 (2) | 2.813 (37) | 23.00 |
| LRTAPSisWSCLTM | 14.660 (17) | 19.617 (15) | 17.244 (22) | 29.649 (13) | 7.670 (2) | 32.891 (45) | 28.374 (13) | 18.595 (4) | 10.260 (33) | 22.153 (16) | 3.263 (26) | 18.73 |
| LRTCCLDM | 10.373 (42) | 19.508 (17) | 16.905 (25) | 31.015 (2) | 7.617 (4) | 35.627 (22) | 29.329 (3) | 16.502 (24) | 11.628 (7) | 22.750 (3) | 3.447 (25) | 15.82 |
| LRTCCLTM | 10.373 (42) | 19.508 (17) | 16.405 (35) | 31.015 (2) | 7.617 (4) | 35.627 (22) | 29.329 (3) | 16.502 (24) | 11.628 (7) | 22.750 (3) | 3.447 (25) | 16.73 |
| LRTCWSCLDM | 6.667 (51) | 17.250 (42) | 11.162 (50) | 23.684 (38) | 7.670 (2) | 38.000 (8) | 28.321 (14) | 16.622 (23) | 10.297 (32) | 21.263 (24) | 2.930 (33) | 28.82 |
| LRTCWSCLTM | 6.667 (51) | 17.250 (42) | 11.162 (50) | 23.648 (39) | 7.670 (2) | 38.000 (8) | 28.321 (14) | 16.655 (21) | 10.370 (30) | 21.263 (24) | 2.930 (33) | 28.55 |
| LRTDSCLDM | 15.233 (12) | 19.744 (13) | 19.107 (4) | 28.697 (18) | 7.387 (15) | 33.459 (34) | 27.487 (28) | 19.298 (2) | 11.534 (8) | 22.491 (9) | 4.567 (4) | 13.36 |
| LRTDSCLTM | 15.233 (12) | 19.744 (13) | 19.141 (3) | 28.697 (18) | 7.387 (15) | 33.459 (34) | 27.487 (28) | 19.298 (2) | 11.534 (8) | 22.491 (9) | 4.567 (4) | 13.27 |
| LRTDSWSCLDM | 13.373 (22) | 20.173 (4) | 15.397 (38) | 26.395 (24) | 7.670 (2) | 34.147 (31) | 28.113 (19) | 18.228 (5) | 9.771 (42) | 22.398 (11) | 2.780 (38) | 21.45 |
| LRTDSWSCLTM | 14.623 (18) | 19.960 (9) | 16.623 (33) | 29.583 (14) | 7.670 (2) | 35.671 (20) | 28.756 (10) | 20.023 (1) | 10.030 (38) | 21.978 (19) | 3.210 (27) | 17.36 |
| LRTAPSesCLDM | 14.910 (15) | 18.978 (33) | 19.055 (6) | 26.620 (23) | 7.407 (14) | 32.146 (52) | 26.649 (34) | 17.667 (8) | 11.294 (10) | 22.284 (14) | 4.370 (5) | 19.45 |
| LRTAPSesCLTM | 14.910 (15) | 18.978 (33) | 19.155 (2) | 26.620 (23) | 7.407 (14) | 32.146 (52) | 26.649 (34) | 17.667 (8) | 11.294 (10) | 22.284 (14) | 4.370 (5) | 19.09 |
| LRTAPSesWSCLDM | 13.037 (23) | 19.982 (7) | 15.452 (37) | 25.910 (25) | 7.670 (2) | 33.291 (37) | 27.228 (31) | 17.453 (11) | 9.931 (41) | 22.761 (2) | 2.813 (37) | 23.00 |
| LRTAPSesWSCLTM | 14.660 (17) | 19.617 (15) | 17.244 (22) | 29.649 (13) | 7.670 (2) | 32.891 (45) | 28.374 (13) | 18.595 (4) | 10.260 (33) | 22.153 (16) | 3.263 (26) | 18.73 |
| LCAAPSisCLDM | 15.323 (10) | 19.844 (10) | 18.370 (18) | 30.370 (9) | 6.342 (25) | 33.136 (39) | 21.912 (54) | 17.550 (10) | 10.791 (19) | 19.102 (39) | 3.473 (24) | 23.36 |
| LCAAPSisCLTM | 15.603 (8) | 19.325 (30) | 18.370 (18) | 30.368 (10) | 6.342 (25) | 32.925 (43) | 22.217 (53) | 17.550 (10) | 10.791 (19) | 18.977 (41) | 3.473 (24) | 25.55 |
| LCAAPSisWSCLDM | 12.513 (36) | 19.261 (31) | 12.350 (44) | 24.778 (32) | 7.249 (18) | 38.492 (6) | 24.106 (49) | 16.623 (22) | 9.097 (51) | 18.739 (44) | 2.317 (44) | 34.27 |
| LCAAPSisWSCLTM | 12.847 (24) | 19.495 (19) | 17.035 (23) | 24.805 (31) | 7.253 (17) | 37.139 (14) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 31.64 |
| LCADSCLDM | 15.390 (9) | 19.771 (11) | 18.455 (14) | 30.231 (11) | 6.342 (25) | 32.906 (44) | 21.881 (55) | 17.550 (10) | 10.791 (19) | 19.350 (35) | 3.473 (24) | 23.36 |
| LCADSCLTM | 15.727 (3) | 19.520 (16) | 18.440 (15) | 30.227 (12) | 6.342 (25) | 33.133 (40) | 21.864 (56) | 17.550 (10) | 10.791 (19) | 19.267 (37) | 3.473 (24) | 23.36 |
| LCADSWSCLDM | 12.627 (30) | 19.127 (32) | 12.130 (46) | 24.815 (29) | 7.249 (18) | 38.148 (7) | 24.226 (46) | 16.623 (22) | 9.097 (51) | 18.635 (48) | 2.317 (44) | 33.91 |
| LCADSWSCLTM | 12.830 (25) | 19.422 (22) | 16.955 (24) | 24.739 (35) | 7.253 (17) | 37.680 (9) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 32.00 |
| LCAAPSesCLDM | 15.323 (10) | 19.844 (10) | 18.370 (18) | 30.370 (9) | 6.342 (25) | 33.136 (39) | 21.912 (54) | 17.550 (10) | 10.791 (19) | 19.102 (39) | 3.473 (24) | 23.36 |
| LCAAPSesCLTM | 15.603 (8) | 19.325 (30) | 18.370 (18) | 30.368 (10) | 6.342 (25) | 32.925 (43) | 22.217 (53) | 17.550 (10) | 10.791 (19) | 18.977 (41) | 3.473 (24) | 25.55 |
| LCAAPSesWSCLDM | 12.513 (36) | 19.261 (31) | 12.350 (44) | 24.778 (32) | 7.249 (18) | 38.492 (6) | 24.106 (49) | 16.623 (22) | 9.097 (51) | 18.739 (44) | 2.317 (44) | 34.27 |
| LCAAPSesWSCLTM | 12.847 (24) | 19.495 (19) | 17.035 (23) | 24.805 (31) | 7.253 (17) | 37.139 (14) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 31.64 |
| LCTAPSisCLDM | 15.693 (4) | 19.973 (8) | 18.480 (13) | 30.723 (5) | 6.342 (25) | 32.891 (45) | 21.296 (60) | 17.550 (10) | 10.791 (19) | 19.170 (38) | 3.473 (24) | 22.82 |
| LCTAPSisCLTM | 15.670 (6) | 20.093 (6) | 18.495 (12) | 30.584 (8) | 6.342 (25) | 33.386 (36) | 21.317 (59) | 17.550 (10) | 10.791 (19) | 18.933 (42) | 3.473 (24) | 22.45 |
| LCTAPSisWSCLDM | 12.747 (28) | 19.408 (23) | 16.780 (31) | 24.811 (30) | 7.249 (18) | 37.016 (15) | 24.419 (45) | 16.623 (22) | 9.097 (51) | 18.259 (51) | 2.317 (44) | 32.55 |
| LCTAPSisWSCLTM | 12.600 (34) | 19.372 (26) | 16.800 (29) | 24.752 (33) | 7.253 (17) | 37.160 (13) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 33.82 |
| LCTCCLDM | 15.640 (7) | 19.496 (18) | 18.395 (16) | 30.802 (4) | 6.352 (24) | 34.110 (32) | 21.787 (58) | 17.552 (9) | 10.898 (16) | 20.248 (32) | 3.473 (24) | 21.82 |
| LCTCCLTM | 15.957 (1) | 20.218 (3) | 18.381 (17) | 31.052 (1) | 6.342 (25) | 35.427 (25) | 22.374 (52) | 17.552 (9) | 10.791 (19) | 20.665 (31) | 3.473 (24) | 18.82 |
| LCTCWSCLDM | 12.740 (29) | 19.344 (29) | 16.830 (28) | 24.811 (30) | 7.249 (18) | 38.502 (5) | 24.209 (47) | 16.623 (22) | 9.097 (51) | 18.721 (45) | 2.317 (44) | 31.64 |

| | iris | wine | zoo | dermatology | hepatitis | breathalyzer | haberman | glass | tae | heartdisease | lungcancer | Avg. Rank |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LCTCWSCLTM | 12.623 (31) | 19.357 (28) | 16.855 (27) | 24.742 (34) | 7.253 (17) | 36.234 (17) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.00 |
| LCTDSCLDM | 15.763 (2) | 19.707 (14) | 18.530 (11) | 30.679 (6) | 6.342 (25) | 33.159 (38) | 21.118 (61) | 17.550 (10) | 10.791 (19) | 18.720 (46) | 3.473 (24) | 23.27 |
| LCTDSCLTM | 15.673 (5) | 20.099 (5) | 18.495 (12) | 30.660 (7) | 6.342 (25) | 33.954 (33) | 21.317 (59) | 17.550 (10) | 10.791 (19) | 19.017 (40) | 3.473 (24) | 21.73 |
| LCTDSWSCLDM | 12.753 (27) | 19.399 (24) | 16.860 (26) | 24.825 (28) | 7.249 (18) | 37.214 (11) | 24.467 (44) | 16.623 (22) | 9.097 (51) | 18.251 (52) | 2.317 (44) | 31.55 |
| LCTDSWSCLTM | 12.607 (33) | 19.363 (27) | 16.785 (30) | 24.752 (33) | 7.253 (17) | 36.887 (16) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.18 |
| LCTAPSesCLDM | 15.693 (4) | 19.973 (8) | 18.480 (13) | 30.723 (5) | 6.342 (25) | 32.891 (45) | 21.296 (60) | 17.550 (10) | 10.791 (19) | 19.170 (38) | 3.473 (24) | 22.82 |
| LCTAPSesCLTM | 15.670 (6) | 20.093 (6) | 18.495 (12) | 30.584 (8) | 6.342 (25) | 33.386 (36) | 21.317 (59) | 17.550 (10) | 10.791 (19) | 18.933 (42) | 3.473 (24) | 22.45 |
| LCTAPSesWSCLDM | 12.747 (28) | 19.408 (23) | 16.780 (31) | 24.811 (30) | 7.249 (18) | 37.016 (15) | 24.419 (45) | 16.623 (22) | 9.097 (51) | 18.259 (51) | 2.317 (44) | 32.55 |
| LCTAPSesWSCLTM | 12.600 (34) | 19.372 (26) | 16.800 (29) | 24.752 (33) | 7.253 (17) | 37.160 (13) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 33.82 |
| LL+RTAPSisWSCLDM | 13.037 (23) | 19.982 (7) | 15.452 (37) | 25.910 (25) | 7.670 (2) | 33.291 (37) | 27.228 (31) | 17.453 (11) | 9.931 (41) | 22.761 (2) | 2.813 (37) | 23.00 |
| LL+RTCCLTM | 10.373 (42) | 19.508 (17) | 16.405 (35) | 31.015 (2) | 7.617 (4) | 35.627 (22) | 29.329 (3) | 16.502 (24) | 11.628 (7) | 22.750 (3) | 3.447 (25) | 16.73 |
| LL+RTCWSCLTM | 6.667 (51) | 17.250 (42) | 11.162 (50) | 23.648 (39) | 7.670 (2) | 38.000 (8) | 28.321 (14) | 16.655 (21) | 10.370 (30) | 21.263 (24) | 2.930 (33) | 28.55 |
| LL+RTDSWSCLTM | 14.623 (18) | 19.960 (9) | 16.623 (33) | 29.583 (14) | 7.670 (2) | 35.671 (20) | 28.756 (10) | 20.023 (1) | 10.030 (38) | 21.978 (19) | 3.210 (27) | 17.36 |
| LL+CTAPSisWSCLDM | 12.747 (28) | 19.408 (23) | 16.780 (31) | 24.811 (30) | 7.249 (18) | 37.016 (15) | 24.419 (45) | 16.623 (22) | 9.097 (51) | 18.259 (51) | 2.317 (44) | 32.55 |
| LL+CTCCLTM | 15.957 (1) | 20.218 (3) | 18.381 (17) | 31.052 (1) | 6.342 (25) | 35.427 (25) | 22.374 (52) | 17.552 (9) | 10.791 (19) | 20.665 (31) | 3.473 (24) | 18.82 |
| LL+CTCWSCLTM | 12.623 (31) | 19.357 (28) | 16.855 (27) | 24.742 (34) | 7.253 (17) | 36.234 (17) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.00 |
| LL+CTDSWSCLTM | 12.607 (33) | 19.363 (27) | 16.785 (30) | 24.752 (33) | 7.253 (17) | 36.887 (16) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.18 |
| LR+CTAPSisWSCLDM | 12.747 (28) | 19.408 (23) | 16.780 (31) | 24.811 (30) | 7.249 (18) | 37.016 (15) | 24.419 (45) | 16.623 (22) | 9.097 (51) | 18.259 (51) | 2.317 (44) | 32.55 |
| LR+CTCCLTM | 15.957 (1) | 20.218 (3) | 18.381 (17) | 31.052 (1) | 6.342 (25) | 35.427 (25) | 22.374 (52) | 17.552 (9) | 10.791 (19) | 20.665 (31) | 3.473 (24) | 18.82 |
| LR+CTCWSCLTM | 12.623 (31) | 19.357 (28) | 16.855 (27) | 24.742 (34) | 7.253 (17) | 36.234 (17) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.00 |
| LR+CTDSWSCLTM | 12.607 (33) | 19.363 (27) | 16.785 (30) | 24.752 (33) | 7.253 (17) | 36.887 (16) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.18 |
| LL+R+CTAPSisWSCLDM | 12.747 (28) | 19.408 (23) | 16.780 (31) | 24.811 (30) | 7.249 (18) | 37.016 (15) | 24.419 (45) | 16.623 (22) | 9.097 (51) | 18.259 (51) | 2.317 (44) | 32.55 |
| LL+R+CTCCLTM | 15.957 (1) | 20.218 (3) | 18.381 (17) | 31.052 (1) | 6.342 (25) | 35.427 (25) | 22.374 (52) | 17.552 (9) | 10.791 (19) | 20.665 (31) | 3.473 (24) | 18.82 |
| LL+R+CTCWSCLTM | 12.623 (31) | 19.357 (28) | 16.855 (27) | 24.742 (34) | 7.253 (17) | 36.234 (17) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.00 |
| LL+R+CTDSWSCLTM | 12.607 (33) | 19.363 (27) | 16.785 (30) | 24.752 (33) | 7.253 (17) | 36.887 (16) | 23.899 (50) | 16.623 (22) | 9.097 (51) | 17.940 (53) | 2.317 (44) | 34.18 |

### 8.2.2 Textual

| | WinXwin | Comp | Talk | Vehicle | Avg. Rank |
|---|---|---|---|---|---|
| Random Selection | 112.707 (22) | 99.852 (38) | 102.483 (45) | 112.572 (27) | 33.00 |
| Uncertainty Sampling | 109.993 (25) | 100.797 (35) | 103.137 (38) | 110.307 (33) | 32.75 |
| Margin Sampling | 109.993 (25) | 100.797 (35) | 103.137 (38) | 110.307 (33) | 32.75 |
| Maximum Diversity Sampling | 108.122 (27) | 96.559 (57) | 104.251 (30) | 106.469 (47) | 40.25 |
| Sparsity Minimization | 111.511 (24) | 105.268 (30) | 105.846 (28) | 113.357 (26) | 27.00 |
| Diversity + Density Maximization | 116.820 (10) | 113.153 (1) | 108.432 (9) | 119.469 (15) | 8.75 |
| Certainty + Sparsity Minimization | 117.368 (9) | 111.804 (7) | 107.491 (17) | 122.310 (11) | 11.00 |
| CGCC | 102.798 (57) | 96.125 (58) | 103.238 (37) | 104.644 (54) | 51.50 |
| CGRC | 103.854 (52) | 97.199 (54) | 100.939 (54) | 112.306 (28) | 47.00 |
| CGLC | 105.198 (41) | 107.075 (25) | 102.522 (43) | 116.088 (22) | 32.75 |
| CGDC | 103.280 (54) | 93.765 (59) | 101.960 (46) | 100.461 (62) | 55.25 |
| CGCS | 106.247 (34) | 88.112 (68) | 93.879 (66) | 103.758 (56) | 56.00 |
| CGRS | 103.953 (51) | 96.586 (56) | 100.177 (59) | 111.172 (30) | 49.00 |
| CGLS | 106.931 (30) | 106.603 (26) | 102.883 (40) | 116.488 (20) | 29.00 |
| CGDS | 97.012 (62) | 93.487 (60) | 102.840 (41) | 102.911 (57) | 55.00 |
| LLAAPSisCLDM | 103.299 (53) | 99.228 (40) | 105.638 (29) | 104.988 (51) | 43.25 |
| LLAAPSisCLTM | 93.768 (64) | 88.945 (67) | 93.553 (67) | 90.571 (67) | 66.25 |
| LLAAPSisWSCLDM | 107.204 (28) | 105.083 (32) | 103.980 (32) | 110.636 (32) | 31.00 |
| LLAAPSisWSCLTM | 105.876 (35) | 106.573 (27) | 100.850 (55) | 109.389 (38) | 38.75 |
| LLADSCLDM | 106.760 (31) | 99.973 (37) | 104.240 (31) | 102.731 (58) | 39.25 |
| LLADSCLTM | 93.794 (63) | 89.104 (66) | 97.403 (63) | 90.226 (68) | 65.00 |
| LLADSWSCLDM | 106.939 (29) | 105.831 (29) | 103.428 (36) | 110.785 (31) | 31.25 |
| LLADSWSCLTM | 105.778 (37) | 106.191 (28) | 101.114 (52) | 109.109 (41) | 39.50 |
| LLAAPSesCLDM | 103.299 (53) | 99.228 (40) | 105.638 (29) | 104.988 (51) | 43.25 |
| LLAAPSesCLTM | 93.768 (64) | 88.945 (67) | 93.553 (67) | 90.571 (67) | 66.25 |
| LLAAPSesWSCLDM | 107.204 (28) | 105.083 (32) | 103.980 (32) | 110.636 (32) | 31.00 |
| LLAAPSesWSCLTM | 105.876 (35) | 106.573 (27) | 100.850 (55) | 109.389 (38) | 38.75 |
| LLTAPSisCLDM | 105.368 (39) | 100.396 (36) | 107.338 (19) | 106.318 (49) | 35.75 |
| LLTAPSisCLTM | 92.308 (66) | 90.405 (64) | 102.514 (44) | 100.479 (61) | 58.75 |
| LLTAPSisWSCLDM | 102.916 (56) | 107.635 (23) | 101.466 (48) | 107.597 (46) | 43.25 |
| LLTAPSisWSCLTM | 100.675 (58) | 91.949 (62) | 96.687 (64) | 98.524 (65) | 62.25 |
| LLTCCLDM | 112.468 (23) | 105.242 (31) | 106.409 (26) | 114.542 (24) | 26.00 |
| LLTCCLTM | 98.275 (61) | 92.185 (61) | 101.367 (50) | 101.093 (60) | 58.00 |
| LLTCWSCLDM | 99.928 (60) | 108.388 (22) | 103.836 (33) | 101.280 (59) | 43.50 |
| LLTCWSCLTM | 99.937 (59) | 96.695 (55) | 99.426 (62) | 99.675 (63) | 59.75 |
| LLTDSCLDM | 106.334 (33) | 99.216 (41) | 103.740 (34) | 104.721 (53) | 40.25 |
| LLTDSCLTM | 93.498 (65) | 89.602 (65) | 99.830 (60) | 97.902 (66) | 64.00 |
| LLTDSWSCLDM | 103.015 (55) | 107.445 (24) | 103.658 (35) | 108.816 (44) | 39.50 |
| LLTDSWSCLTM | 100.675 (58) | 91.818 (63) | 96.685 (65) | 98.547 (64) | 62.50 |
| LLTAPSesCLDM | 105.368 (39) | 100.396 (36) | 107.338 (19) | 106.318 (49) | 35.75 |
| LLTAPSesCLTM | 92.308 (66) | 90.405 (64) | 102.514 (44) | 100.479 (61) | 58.75 |
| LLTAPSesWSCLDM | 102.916 (56) | 107.635 (23) | 101.466 (48) | 107.597 (46) | 43.25 |
| LLTAPSesWSCLTM | 100.675 (58) | 91.949 (62) | 96.687 (64) | 98.524 (65) | 62.25 |
| LRAAPSisCLDM | 104.867 (45) | 99.229 (39) | 108.227 (11) | 106.412 (48) | 35.75 |
| LRAAPSisCLTM | 104.867 (45) | 99.229 (39) | 108.227 (11) | 106.412 (48) | 35.75 |
| LRAAPSisWSCLDM | 113.502 (19) | 110.359 (19) | 106.441 (25) | 115.921 (23) | 21.50 |
| LRAAPSisWSCLTM | 113.829 (15) | 111.223 (14) | 106.567 (23) | 116.976 (19) | 17.75 |
| LRADSCLDM | 104.554 (48) | 97.312 (53) | 106.555 (24) | 103.961 (55) | 45.00 |
| LRADSCLTM | 104.554 (48) | 97.312 (53) | 106.555 (24) | 103.961 (55) | 45.00 |
| LRADSWSCLDM | 114.573 (14) | 111.503 (11) | 107.815 (14) | 116.352 (21) | 15.00 |
| LRADSWSCLTM | 115.722 (12) | 112.544 (3) | 108.439 (8) | 117.426 (18) | 10.25 |
| LRAAPSesCLDM | 104.867 (45) | 99.229 (39) | 108.227 (11) | 106.412 (48) | 35.75 |
| LRAAPSesCLTM | 104.867 (45) | 99.229 (39) | 108.227 (11) | 106.412 (48) | 35.75 |
| LRAAPSesWSCLDM | 113.502 (19) | 110.359 (19) | 106.441 (25) | 115.921 (23) | 21.50 |
| LRAAPSesWSCLTM | 113.829 (15) | 111.223 (14) | 106.567 (23) | 116.976 (19) | 17.75 |
| LRTAPSisCLDM | 104.732 (47) | 98.420 (50) | 107.362 (18) | 106.045 (50) | 41.25 |
| LRTAPSisCLTM | 104.732 (47) | 98.420 (50) | 107.362 (18) | 106.045 (50) | 41.25 |
| LRTAPSisWSCLDM | 120.250 (2) | 109.631 (20) | 108.834 (5) | 123.276 (10) | 9.25 |
| LRTAPSisWSCLTM | 114.655 (13) | 108.476 (21) | 108.867 (4) | 123.514 (9) | 11.75 |
| LRTCCLDM | 113.601 (17) | 104.740 (33) | 106.709 (22) | 114.028 (25) | 24.25 |

| | WinXwin | Comp | Talk | Vehicle | Avg. Rank |
|---|---|---|---|---|---|
| LRTCCLTM | 113.601 (17) | 104.740 (33) | 106.709 (22) | 114.028 (25) | 24.25 |
| LRTCWSCLDM | 116.611 (11) | 111.581 (10) | 107.508 (16) | 122.300 (12) | 12.25 |
| LRTCWSCLTM | 116.611 (11) | 111.581 (10) | 107.508 (16) | 122.300 (12) | 12.25 |
| LRTDSCLDM | 104.757 (46) | 97.882 (52) | 106.817 (21) | 104.894 (52) | 42.75 |
| LRTDSCLTM | 104.757 (46) | 97.882 (52) | 106.817 (21) | 104.894 (52) | 42.75 |
| LRTDSWSCLDM | 122.004 (1) | 112.972 (2) | 113.029 (2) | 123.629 (8) | 3.25 |
| LRTDSWSCLTM | 117.457 (8) | 110.739 (17) | 113.095 (1) | 124.257 (7) | 8.25 |
| LRTAPSesCLDM | 104.732 (47) | 98.420 (50) | 107.362 (18) | 106.045 (50) | 41.25 |
| LRTAPSesCLTM | 104.732 (47) | 98.420 (50) | 107.362 (18) | 106.045 (50) | 41.25 |
| LRTAPSesWSCLDM | 120.250 (2) | 109.631 (20) | 108.834 (5) | 123.276 (10) | 9.25 |
| LRTAPSesWSCLTM | 114.655 (13) | 108.476 (21) | 108.867 (4) | 123.514 (9) | 11.75 |
| LCAAPSisCLDM | 106.403 (32) | 99.178 (43) | 100.764 (57) | 108.800 (45) | 44.25 |
| LCAAPSisCLTM | 105.695 (38) | 98.990 (45) | 101.361 (51) | 110.273 (34) | 42.00 |
| LCAAPSisWSCLDM | 113.543 (18) | 111.347 (13) | 107.803 (15) | 118.792 (17) | 15.75 |
| LCAAPSisWSCLTM | 113.359 (20) | 110.373 (18) | 108.142 (12) | 119.591 (14) | 16.00 |
| LCADSCLDM | 105.801 (36) | 98.871 (48) | 103.023 (39) | 109.752 (35) | 39.50 |
| LCADSCLTM | 105.158 (43) | 98.982 (46) | 100.438 (58) | 109.517 (37) | 46.00 |
| LCADSWSCLDM | 113.726 (16) | 111.932 (6) | 107.867 (13) | 119.462 (16) | 12.75 |
| LCADSWSCLTM | 113.321 (21) | 111.798 (9) | 108.816 (6) | 120.334 (13) | 12.25 |
| LCAAPSesCLDM | 106.403 (32) | 99.178 (43) | 100.764 (57) | 108.800 (45) | 44.25 |
| LCAAPSesCLTM | 105.695 (38) | 98.990 (45) | 101.361 (51) | 110.273 (34) | 42.00 |
| LCAAPSesWSCLDM | 113.543 (18) | 111.347 (13) | 107.803 (15) | 118.792 (17) | 15.75 |
| LCAAPSesWSCLTM | 113.359 (20) | 110.373 (18) | 108.142 (12) | 119.591 (14) | 16.00 |
| LCTAPSisCLDM | 105.115 (44) | 98.557 (49) | 99.758 (61) | 108.954 (43) | 49.25 |
| LCTAPSisCLTM | 104.402 (49) | 99.100 (44) | 100.814 (56) | 109.171 (39) | 47.00 |
| LCTAPSisWSCLDM | 118.502 (4) | 111.969 (5) | 107.206 (20) | 125.481 (1) | 7.50 |
| LCTAPSisWSCLTM | 118.083 (7) | 110.955 (15) | 108.692 (7) | 124.798 (6) | 8.75 |
| LCTCCLDM | 109.482 (26) | 101.852 (34) | 102.566 (42) | 112.005 (29) | 32.75 |
| LCTCCLTM | 105.188 (42) | 98.942 (47) | 101.027 (53) | 108.981 (42) | 46.00 |
| LCTCWSCLDM | 119.022 (3) | 112.073 (4) | 106.205 (27) | 125.223 (3) | 9.25 |
| LCTCWSCLTM | 118.233 (6) | 111.477 (12) | 109.046 (3) | 125.258 (2) | 5.75 |
| LCTDSCLDM | 105.203 (40) | 98.243 (51) | 101.624 (47) | 109.668 (36) | 43.50 |
| LCTDSCLTM | 104.186 (50) | 99.202 (42) | 101.410 (49) | 109.113 (40) | 45.25 |
| LCTDSWSCLDM | 118.433 (5) | 111.801 (8) | 108.324 (10) | 125.126 (4) | 6.75 |
| LCTDSWSCLTM | 118.083 (7) | 110.876 (16) | 108.692 (7) | 124.826 (5) | 8.75 |
| LCTAPSesCLDM | 105.115 (44) | 98.557 (49) | 99.758 (61) | 108.954 (43) | 49.25 |
| LCTAPSesCLTM | 104.402 (49) | 99.100 (44) | 100.814 (56) | 109.171 (39) | 47.00 |
| LCTAPSesWSCLDM | 118.502 (4) | 111.969 (5) | 107.206 (20) | 125.481 (1) | 7.50 |
| LCTAPSesWSCLTM | 118.083 (7) | 110.955 (15) | 108.692 (7) | 124.798 (6) | 8.75 |
| LL+RTAPSisWSCLDM | 120.250 (2) | 109.631 (20) | 108.834 (5) | 123.276 (10) | 9.25 |
| LL+RTCCLTM | 113.601 (17) | 104.740 (33) | 106.709 (22) | 114.028 (25) | 24.25 |
| LL+RTCWSCLTM | 116.611 (11) | 111.581 (10) | 107.508 (16) | 122.300 (12) | 12.25 |
| LL+RTDSWSCLTM | 117.457 (8) | 110.739 (17) | 113.095 (1) | 124.257 (7) | 8.25 |
| LL+CTAPSisWSCLDM | 118.502 (4) | 111.969 (5) | 107.206 (20) | 125.481 (1) | 7.50 |
| LL+CTCCLTM | 105.188 (42) | 98.942 (47) | 101.027 (53) | 108.981 (42) | 46.00 |
| LL+CTCWSCLTM | 118.233 (6) | 111.477 (12) | 109.046 (3) | 125.258 (2) | 5.75 |
| LL+CTDSWSCLTM | 118.083 (7) | 110.876 (16) | 108.692 (7) | 124.826 (5) | 8.75 |
| LR+CTAPSisWSCLDM | 118.502 (4) | 111.969 (5) | 107.206 (20) | 125.481 (1) | 7.50 |
| LR+CTCCLTM | 105.188 (42) | 98.942 (47) | 101.027 (53) | 108.981 (42) | 46.00 |
| LR+CTCWSCLTM | 118.233 (6) | 111.477 (12) | 109.046 (3) | 125.258 (2) | 5.75 |
| LR+CTDSWSCLTM | 118.083 (7) | 110.876 (16) | 108.692 (7) | 124.826 (5) | 8.75 |
| LL+R+CTAPSisWSCLDM | 118.502 (4) | 111.969 (5) | 107.206 (20) | 125.481 (1) | 7.50 |
| LL+R+CTCCLTM | 105.188 (42) | 98.942 (47) | 101.027 (53) | 108.981 (42) | 46.00 |
| LL+R+CTCWSCLTM | 118.233 (6) | 111.477 (12) | 109.046 (3) | 125.258 (2) | 5.75 |
| LL+R+CTDSWSCLTM | 118.083 (7) | 110.876 (16) | 108.692 (7) | 124.826 (5) | 8.75 |

## 8.3 Open source Technologies Used

### 8.3.1 Development Tools

#### TortoiseGit

*TortoiseGit* [42] is an incredibly easy to use interface to Git on Windows, and was used as the primary means of interacting with Git when working in a Windows environment.

#### Eclipse

*Eclipse* [43] was the IDE used for all development. It was chosen due to its extreme extensibility and the author's prior familiarity with it.

#### Pydev

*Pydev* [44] is a Python plugin for Eclipse that allows Eclipse to be used as a full-fledged Python IDE. It features a debugger, syntax awareness, auto completion, and other features one would expect from an IDE.

#### Aptana

*Aptana* [45] is another Eclipse plugin that is primarily aimed at Web Development, though does add some small additional Python features, and includes Git integration.

#### RunSnakeRun

*RunSnakeRun* [46] is a Python profiling output viewer. It was very useful in helping identify areas of code which would likely provide the largest performance gains if refactored.

#### bpython

*bpython* [47] is "a fancy interface to the Python interpreter for Unix-like operating systems". It was used as the primary Python interactive interface throughout the project, due to its productivity-improving features such as auto-complete, and documentation string showing.

#### Graphviz

*Graphviz* [22] is a tool which provides many utilities for graph layout and visualization. It is highly developed and allows massive scope in specifying customizations. It was used in generating the selection graphs for the different selection strategies.

#### git

*Git* [48] was the source-control system used, with GitHub as the host. It was primarily chosen due to its powerful support for maintaining local repositories in addition to remote ones. This came in very useful during the times I was without network access.

### 8.3.2 Development Technologies

#### Python

*Python* [49] is a high-level programming language that allows for rapid development. It was chosen due to the author's familiarity with the language, its prettiness, the large availability of Open Source libraries for it, and the need for rapid development due to the relatively short timespan of the Final Year Project.

#### PyPy

*PyPy* [50] is an alternative implementation from the standard cPython interpreter, and can often give significant performance gains to Python programs. The downside is that it doesn't support SWIG (the layer which allows interface to C++ and other code), and thus rules out the use of many libraries, and thus is not appropriate for all Python programs.

PyPy was tried in the project after the Orange dependencies were removed, but it was found not to give a worthwhile performance improvement, so cPython was used instead. The probable reason for this is the frequent usage of operations such as list comprehensions that are already quite optimized in the cPython implementation.

#### Orange

*Orange* [13] is a Machine Learning framework with bindings for Python. It was used heavily initially, but as the project progressed, our usage patterns differed compared to what it was designed for, leading to significant memory related issues. We thus elected to just use it for data-loading, and distance computation, implementing the other bits we needed ourselves.

It was chosen as opposed to the other Machine Learning frameworks available for Python, as it seemed the most mature and diverse of group.

#### Matplotlib

*Matplotlib* [14] is a very popular plotting library for Python, meant to provide a feature-rich, Open Source alternative to Matlab, and Mathematica.

It was used at the outset of the project to perform initial plotting, but was later abandoned in favour of PyX due to its non-pythonic interface which was meant more for interactive use.

#### PyX

*PyX* [15] is a graphics package for Python. It was used in the project for producing plots, and was chosen for its very LaTeX-friendly output formats, good documentation, and well formed interface.

### SciKit Learn

*SciKit Learn* [19] "is a Python module integrating classic machine learning algorithms in the tightly-knit world of scientific Python packages". It was used for the Cross Validation splitting, and also for textual dataset loading and vectorizing.

During the progression of the project, a bug was identified in this package, and a fix and unit test submitted to its repository [51].

### NumPy

*NumPy* [52] is "the fundamental package needed for scientific computing with Python". It allows for very performant numerical computations in Python. It was used for generating the Cosine similarity between the tf-idf document vectors obtained from SciKit-Learn for the Textual Datasets.

### cProfile

*cProfile* [53] is the main recommended profiling engine for Python, achieving relatively low overhead compared to the traditional profile module. It came bundled with Python from v2.5 onwards and is based on the lsprof [54] library. It was used in the project for all Profiling, when performance improvements were needed.

### PyGraphviz

*PyGraphviz* [23] is one of a number of Python interfaces to Graphviz - both in aiding generating '.dot' files, and also in generating the graphs through the Graphviz binary. It was used in the project as the primary mechanism of interacting with Graphviz for the generation of selection graphs.

### pyPdf

*pyPdf* [55] is a pure-Python toolkit for PDF manipulation. It was used in the project to combine selection graphs into a single multi-page PDF.

### Protocol Buffers

*Protocol Buffers* [18] 'are a way of encoding structured data in an efficient yet extensible format". They are Google's primary data-interchange format, and were used in the project for storing Precomputed Distance style Datasets.

### Nose

*Nose* [56] is a very nice Python unit test runner, which supports concurrent usage of multiple types of Python unit tests. It was used in the project for executing the various doctests and unit tests written for the project code.

#### latexcodec

*latexcodec* [57] is a Python Package which allows easy encoding of text into a LaTeX-escaped format. It was used in generating the program outputs, so that they could be included directly in LaTeX.

#### Mincemeat

*Mincemeat* [16] is a pure Python Map-Reduce implementation, allowing straightforward distributed parallel execution of code. It was used in enabling cluster-based operation of the platform.

#### Virtualenv

*Virtualenv* [21] is a tool which allows setting up of a reproducible, isolated, Python environment. It was used in setting up an execution environment based purely on a network-mounted home folder to allow the platform to operate on any of departmental lab computers.

### 8.3.3 Documentation Related

#### LaTeX

LaTeX [58] is "a high-quality typesetting system". It was used to write the report that accompanied the project.

#### python.sty

*python.sty* [59] is a LaTeX package that provides the capability to embed Python code in a LaTeX file, and include the output of the code at that position in the file. It was incredibly useful in automatically including the platform outputs of experiments in the report by programatically iterating through the results, and outputting the appropriate LaTeX code.

#### TexStudio

*TexStudio* [60] is an Open Source, Cross-Platform LaTeX IDE, providing rich support and integration with LaTeX. It was used as the primary editor for writing the report.

#### extraplaceins

*extraplaceins* [61] is a LaTeX package which extends the standard 'placeins' package. It includes capabilities to prevent figures in LaTeX from floating outside sections. It was used in the report to force diagrams to stay in their desired places.

#### LyX

*LyX* [62] is an Open Source, Cross-Platform document processor built around LaTeX. I found it slightly kludgey for writing 'pure' LaTeX, but it was very useful for generating the source for some of the more complicated tables used in the report.

#### LibreOffice

*LibreOffice* [63] is an Open Source alternative to Microsoft Office, and was used for writing the Presentation for the project to present on the Project Open Day.

#### briss

*briss* [64] is a simple Cross-Platform GUI for the cropping PDFs. It was used for cropping extracted PDF pages to include just the algorithms from Hu's report, excluding the other text around the algorithms.

#### Dia

*Dia* [65] is an Open Source alternative to Microsoft Visio, and was used to draw the visual examples presented in this report.

#### PDF Rider

*PDF Rider* [66] is a "utility for doing some simple manipulations to PDF documents". It was used for extracting individual pages from Hu's report so as to include the algorithms presented verbatim.

### 8.3.4 Misc

#### Ubuntu

Due to some of the technologies involved currently requiring a *nix environment, *Ubuntu* [67] was chosen as the main OS in which to do the development and experiment-running related to the project.

#### WinSCP

*WinSCP* [68] is an SFTP, SCP and FTP client for Windows. It was used for downloading experiment results from the remote college server used for executing the larger experiment.

#### UnxUtils

*UnxUtils* [69] is a package that includes a group of common *nix utilities ported to native Windows binaries. It was used to get the python.sty LaTeX package executing correctly (the package relying on the 'cat' *nix command).

#### PuTTY

*PuTTY* [70] is a Telnet and SSH client. It was used for remotely executing commands on the college server used in running experiments, aswell as tunnelling traffic through a publicly accessible server to the remote server (which was on a private network on which the public one was also connected).

### VirtualBox

*VirtualBox* [71] is a Cross-Platform virtualization program - an Open Source alternative to the likes of VMWare and HyperV. It was used for running my Ubuntu environment, making it easily transferable across different systems. This allowed me to not have to worry about installing all the project dependencies on all the systems I was using for experiment running.

### PythonBrew

*PythonBrew* [72] is a small utility for setting up multiple concurrently usable versions of Python in one's home directory. It was used in setting up a bare-bones experiment running environment on the college server used, which was operating on an old version of Ubuntu (which only had Python 2.5.2 installed).

# Bibliography


[1] Burr Settles. Active Learning Literature Survey. *Sciences-New York*, 2010.

[2] Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, Andrew Sparkes, Kenneth E Whelan, and Amanda Clare. The automation of science. *Science (New York, N.Y.)*, 324(5923): 85–9, April 2009. ISSN 1095-9203. doi: 10.1126/science.1165620. URL http://www.ncbi.nlm.nih.gov/pubmed/19342587.

[3] Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating diversity and density in active learning for relevance feedback. *Advances in Information Retrieval*, pages 246–257, 2007. URL http://www.springerlink.com/index/J20663R2R1116141.pdf.

[4] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. *Proceedings of the 22nd International Conference on Computational Linguistics - COLING '08*, 1(August):1137–1144, 2008. doi: 10.3115/1599081.1599224. URL http://portal.acm.org/citation.cfm?doid=1599081.1599224.

[5] Barry Smyth and Elizabeth McKenna. Modelling the competence of case-bases. In Barry Smyth and Pádraig Cunningham, editors, *Advances in Case-Based Reasoning*, volume 1488 of *Lecture Notes in Computer Science*, pages 208–220. Springer Berlin / Heidelberg, 1998. ISBN 978-3-540-64990-8. URL http://dx.doi.org/10.1007/BFb0056334.

[6] L. Cummins. *Combining and Choosing Case Base Maintenance Algorithms (Upcoming)*. PhD thesis, Department of Computer Science, University College Cork, Ireland, 2011.

[7] D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994. URL http://dl.acm.org/citation.cfm?id=188490.188495.

[8] Rong Hu. *Active Learning for Text Classification*. PhD thesis, Dublin Institute of Technology, 2011. URL http://arrow.dit.ie/sciendoc/115/.

[9] Rong Hu, Brian Mac Namee, and Sarah Jane Delany. Off to a Good Start : Using Clustering to Select the Initial Training Set in Active Learning. *Artificial Intelligence*, (Flairs):26–31, 2010.

[10] S.J. Sarah Delany, Pádraig Cunningham, Dónal Doyle, and Anton Zamolotskikh. Generating estimates of classification confidence for a case-based spam filter. In Héctor Muñoz Ávila and Francesco Ricci, editors, *Case-Based Reasoning Research and Development*, volume 3620 of *Lecture Notes in Computer Science*, pages 177–190. Springer, 2005. ISBN 978-3-540-28174-0. URL http://dx.doi.org/10.1007/11536406_16.

[11] Derek Greene. *A State-of-the-Art Toolkit for Document Clustering*. PhD thesis, 2007.

[12] Weka, 1997. URL http://www.cs.waikato.ac.nz/ml/weka/.

[13] Orange, 2005. URL http://orange.biolab.si/.

[14] Matplotlib, 2008. URL http://matplotlib.sourceforge.net/.

[15] PyX, 2002. URL http://pyx.sourceforge.net/.

[16] Michael Fairley. Mincemeat. URL https://github.com/michaelfairley/mincemeatpy#readme.

[17] A Frank and A Asuncion. UCI Machine Learning Repository, 2010. URL http://archive.ics.uci.edu/ml.

[18] Protocol Buffers, . URL http://code.google.com/apis/protocolbuffers/.

[19] SciKit Learn, . URL http://scikit-learn.org/.

[20] K-Fold Cross Validation Demonstration, . URL http://animation.yihui.name/dmml:k-fold_cross-validation.

[21] Ian Bicking. virtualenv. URL http://www.virtualenv.org.

[22] Graphviz, 2001. URL http://www.graphviz.org/.

[23] PyGraphviz, . URL http://networkx.lanl.gov/pygraphviz/.

[24] Barry Smyth and Mark T Keane. Remembering To Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. In *Proceedings of the 14th International Joint Conference*, pages 377–382. Morgan Kaufmann, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.2767.

[25] Sarah Delany. The Good, the Bad and the Incorrectly Classified: Profiling Cases for Case-Base Editing. In Lorraine McGinty and David Wilson, editors, *Case-Based Reasoning Research and Development*, volume 5650 of *Lecture Notes in Computer Science*, pages 135–149. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-02997-4. URL http://dx.doi.org/10.1007/978-3-642-02998-1_11.

[26] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, pages 179–188, 1936. URL http://archive.ics.uci.edu/ml/machine-learning-databases/iris/.

[27] Lanteri S. Forina M, Leardi R, Armanino C. *PARVUS - An Extendible Package for Data Exploration, Classification and Correlation*. PhD thesis, Via Brigata Salerno, 16147 Genoa, Italy, 1988. URL http://archive.ics.uci.edu/ml/machine-learning-databases/wine/.

[28] Richard S. Forsyth. Zoo database, 1990. URL http://archive.ics.uci.edu/ml/machine-learning-databases/zoo/.

[29] Nilsen Ilter and H. Altay Guvenir. Dermatology Database, 1998. URL http://archive.ics.uci.edu/ml/machine-learning-databases/dermatology/.

[30] Gail Gong. Hepatitis Domain, 1988. URL http://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/.

[31] D. Doyle, P. Cunningham, D.G. Bridge, and Y. Rahman. Explanation Oriented Retrieval. In P. Funk and P. González-Calero, editors, *Procs. of the 7th European Conference on Case-Based Reasoning*, LNCS-3155, pages 157–168. Springer, 2004.

[32] S. J. Haberman. Generalized Residuals for Log-Linear Models. In *Proceedings of the 9th International Biometrics Conference, Boston*, pages 104–122, 1976. URL http://archive.ics.uci.edu/ml/machine-learning-databases/haberman/.

[33] B. German and Vina Spiehler. Glass Identification Database, 1987. URL http://archive.ics.uci.edu/ml/machine-learning-databases/glass/.

[34] Wei-Yin Loh and Y.-S. Shih. Split Selection Methods for Classification Trees. *Statistica Sinica*, 7:815–840, 1997. URL http://archive.ics.uci.edu/ml/machine-learning-databases/tae/.

[35] Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano. Heart Disease Databases, 1988. URL http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/.

[36] Z.Q. Hong and J.Y. Yang. Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane. *Pattern Recognition*, 24(4):317–324, 1992. URL http://archive.ics.uci.edu/ml/machine-learning-databases/lung-cancer/.

[37] Tom Mitchell. comp.windows.x vs comp.os.ms-windows.misc, 1999. URL http://kdd.ics.uci.edu/databases/20newsgroups/.

[38] Tom Mitchell. comp.sys.ibm.pc.hardware vs comp.sys.mac.hardware, 1999. URL http://kdd.ics.uci.edu/databases/20newsgroups/.

[39] Tom Mitchell. talk.religion.misc vs alt.atheism, 1999. URL http://kdd.ics.uci.edu/databases/20newsgroups/.

[40] Tom Mitchell. rec.autos vs rec.motorcycles, 1999. URL http://kdd.ics.uci.edu/databases/20newsgroups/.

[41] Project Source Code, . URL https://github.com/Epinoch/FYP.

[42] TortoiseGit, . URL http://code.google.com/p/tortoisegit/.

[43] Eclipse, 2004. URL http://www.eclipse.org/.

[44] PyDev, . URL http://pydev.org/.

[45] Aptana, . URL http://aptana.com/.

[46] RunSnakeRun, . URL http://www.vrplumber.com/programming/runsnakerun/.

[47] bpython, . URL http://bpython-interpreter.org/home/.

[48] Git, 2005. URL http://git-scm.com/.

[49] Python, . URL http://www.python.org.

[50] PyPy, . URL http://pypy.org/.

[51] SciKit Learn Open Source Bug Identification and Resolution Contribution., . URL https://github.com/scikit-learn/scikit-learn/pull/577.

[52] NumPy, 1995. URL http://numpy.scipy.org/.

[53] cProfile, . URL http://docs.python.org/library/profile.html#module-cProfile.

[54] lsprof, . URL https://code.launchpad.net/lsprof.

[55] pyPdf, . URL http://pybrary.net/pyPdf/.

[56] Nose, . URL https://github.com/nose-devs/nose.

[57] latexcodec, 2011. URL http://pypi.python.org/pypi/latexcodec.

[58] LaTeX, 1980. URL http://www.latex-project.org/.

[59] Python.sty, . URL http://weongyo.org/docs/SpiderMonkey/python.sty.

[60] TexStudio, . URL http://texstudio.sourceforge.net/.

[61] extraplaceins, . URL http://lexfridman.com/blogs/research/2011/03/06/prevent-figures-from-floating-outside-sections-in-latex/.

[62] LyX, 1995. URL http://www.lyx.org/.

[63] LibreOffice, 1999. URL http://www.libreoffice.org/.

[64] briss, 2010. URL http://sourceforge.net/projects/briss/.

[65] Dia, 2001. URL http://dia-installer.de/.

[66] PDF Rider, 2009. URL http://pdfrider.codeplex.com.

[67] Ubuntu, 2004. URL http://www.ubuntu.org.

[68] WinSCP, 2000. URL http://winscp.net/eng/index.php.

[69] GNU utilities for Win32, . URL http://unxutils.sourceforge.net/.

[70] PuTTY, . URL http://www.chiark.greenend.org.uk/~sgtatham/putty/.

[71] VirtualBox, 2007. URL https://www.virtualbox.org/.

[72] PythonBrew, . URL http://pypi.python.org/pypi/pythonbrew.