

# Iterative Semi-Supervised Learning: Helping the User to Find the Right Records

Chris Drummond

Institute for Information Technology,  
National Research Council Canada,  
Ottawa, Ontario, Canada, K1A 0R6  
Chris.Drummond@nrc-cnrc.gc.ca

**Abstract.** This paper proposes extending semi-supervised learning by allowing an ongoing interaction between a user and the system. The extension is intended to not only to speed up search for relevant aircraft engine maintenance records but also to help in improving the user's understanding of the problem domain. After the user has identified a small number of relevant records, the system produces a description which generalizes their common properties. If the user is satisfied with the description, the system retrieves more potentially relevant records. The user critiques the items returned, labeling them as relevant or not. The system updates the description using this new labeling information and retrieves more records. The process continues until the user is satisfied that most relevant records have been found. To validate the efficacy of the approach, a set of related maintenance records are collected using the system. These records are compared to those collected without system support.

## 1 Introduction

This paper proposes to extend semi-supervised learning to allow labeling to be an ongoing process. In the standard approach, information from a large number of unlabelled examples is used to try to overcome poor classification due to a relatively small number of labeled examples. Here, unlabelled examples are also used but instead of a fixed set of labeled examples, the user progressively labels more of them as search continues. Another important difference is that, unlike other semi-supervised learning algorithms [2,6,10], the system generates a description of the common properties of the relevant records. This is advantageous for a number of reasons. Extracting the essential properties of relevant records helps the user to understand the domain and to confirm that the right records have been found. The description is also useful as a basis for finding a different set of related records in the same domain. Further, in this application at least, the description forms the basis of an SQL query which actually retrieves the records from the relational database where they are stored.

Although existing semi-supervised learning algorithms might be easily extended to make them incremental, it would be much harder to modify them to generate the descriptions needed here. Instead, clustering, a traditional unsupervised learning method, is used to generate a tree that represents the degree of similarity between re-

cords. This tree, in conjunction with labeling information supplied by the user, is used to identify additional relevant records and to guide how rule generalization is applied to these records to generate the description.

The main motivation for this research is the construction of parametric models for fault prediction in turbofan engines on commercial aircraft [9]. Commercial aircraft are costly to maintain. Unscheduled maintenance can increase costs considerably. Failures are often detected while the aircraft is being readied for the next flight. The resultant delays produce customer dissatisfaction and a loss of profit, perhaps to the point where all profit for an individual flight is wiped out. If the airport does not carry the necessary part in stock, a new part may have to be flown in, increasing delays and costs further. Any advanced warning of component failure, which allows repair at a more appropriate juncture, would be of great benefit to an airline.

To support fault prediction, we have access to a large database of maintenance records. But before fault prediction models can be constructed, it is necessary to identify which maintenance records refer to which parts being replaced. Each record consists of a number of related forms which in turn describe the problem first being noticed, the action taken to correct the problem, the part removed and the part that replaced it. What makes the task difficult is that not all the forms are filled out completely on every occasion. Sometimes information is missing; sometimes it is entered into the wrong field. The problem is exacerbated as several different part numbers may be used: the manufacturer's, the airline's or possibly numbers from other sources. The numbers are not always entered completely; additional numbers are often included, indicating such things as revision levels. Surprisingly, however, there is no real ambiguity in the part that was actually replaced. There are free text fields in a couple of the forms and the mechanics are careful to detail the action taken.

Once the maintenance record has been located, a user who has sufficient familiarity with the domain will have little doubt in which part was replaced. The difficulty lies in writing a query that returns the appropriate records. Locating relevant records is invariably an iterative process; seldom does an initial query return exactly what the user requires. In the approach proposed here, the user constructs a simple description of at least some of the relevant records. The system converts this into an SQL query and retrieves matching records. The user critiques these records by labeling the relevant ones as positive and the irrelevant ones as negative. The system uses this information to generate a new description. This the user studies, possibly altering it if appropriate. It is then used to generate a new query and retrieve more records. This process repeats until the user is satisfied that all, or at least a significant fraction, of the relevant records have been returned.

This approach to finding relevant records should be useful to support fault prediction modeling in other domains where the down time of equipment is expensive. For instance, in mining and quarrying the heavy trucks often operate 24 hours a day and having one out of service is a serious and costly problem. This approach should also be useful in applications not part of a larger data mining process. There are certainly many situations in which humans collect related records. It might be important to find medical records where patients of similar age have similar symptoms, for example. Another example is configuration management. It might be useful to gather engineering change orders, either for hardware or software, based on the types of problem that lead to particular changes.

## 2 Supporting a User's Search

This section begins by describing in detail the user interface and how it might be used to search for a set of relevant maintenance records. It then discusses the background processes that support search and how they are initiated as a result of user actions.

### 2.1 The User Interface

The user interface, shown in Figure 1, is divided into three main parts. At the top is a series of buttons that in turn control the clustering ("Cluster"), the generation of the description ("Match"), the retrieval of the records ("Query") and the ability to back-track ("Restore"), at present only to the beginning of search. The upper small table shows the description. The large table occupying the lower three quarters of the interface displays a scrollable list of the individual records that meet the current description. At the beginning of search the description is empty and the list includes all of the maintenance records detailing problems with aircraft engines. For each record, the first three fields indicate the record number, the aircraft on which the problem occurred and the date when it occurred. The next two fields give the manufacturer's part number both for the component that was installed and for the component that was removed. The next two fields give the standard industry number (called the Item Id) for both the installed and removed component. Lastly, there are two free text fields, the first entered by the mechanic who noticed the problem and the second by the mechanic who fixed it.

The description table has equivalent fields for the four part numbers and two text entries. The user can add information directly to these fields and press the "Query" button to retrieve records that match the resultant description. Alternatively, the user can label individual records by pressing the left or right mouse button for positive or negative labels respectively (the middle button removes the label). When the "Match" button is pressed, a description is generated automatically from the labeling information. The user can modify it, if desired. Each row of the description table is a single disjunct and pressing the "Query" button returns the union of the records for each row. By clicking the mouse at the beginning of any row, the records that match only that disjunct are retrieved.

Typically, from the author's experience of searching maintenance records using the new system, a user starts with a very simple description, perhaps even a single term. The user studies the matching records to see if they are relevant. If it is clear from the records what common characteristics define relevancy, the user might add extra terms manually to the existing description. If it is not clear, the user would label the items that are clearly relevant or irrelevant and review the description generated by the system. Again, the user might directly modify the description or allow the system generated description to expand the search. We would anticipate that, particularly in an unfamiliar domain, initially the user would depend on the system to control the search. However, as familiarity is gained, the user should have more confidence in modifying the description directly.

Search ends when the user is satisfied that a sufficient proportion of the relevant records have been found. What constitutes sufficient is problem dependent. In this

application, it is important to have nearly all the relevant records. The records form the basis of training data for additional learning algorithms to extract fault prediction models. More examples tend to produce more accurate models. A more general concern, likely also true in other domains, is that if a significant number of relevant records are missed there is a danger they may concern problems of a single type and thus an important problem class may be overlooked.



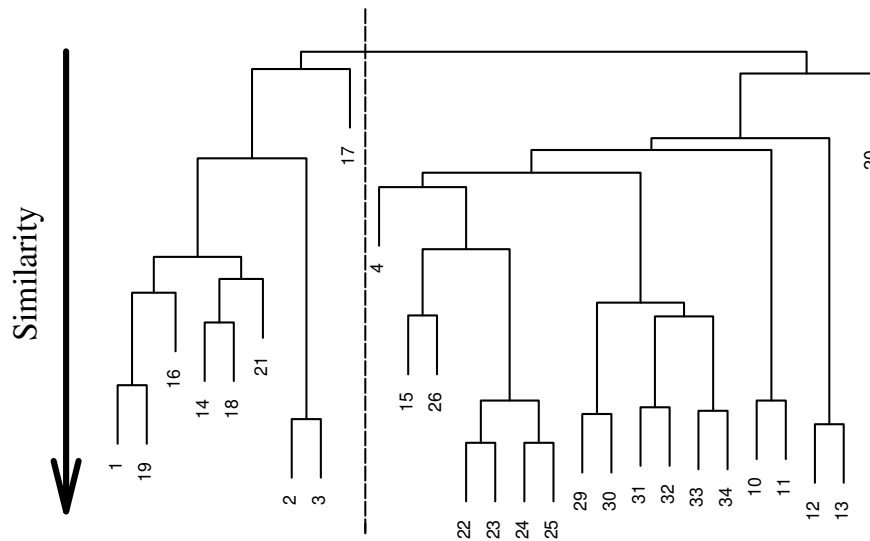
Figure 1: The User Interface.

## 2.2 Background Processes

There are three essential background processes: clustering the maintenance records; labeling new potentially relevant records by spreading activation; generating the description shown to the user through rule generalization.

**Clustering the Records.** A central data structure used by the system is a tree generated by clustering all the maintenance records. The simple agglomerative clustering algorithm “agnes” [7] is used. This takes the two most similar items and combines them into a cluster. The next closest pair, which may be two items or one item and the existing cluster, are combined to form a new cluster. This is repeated combining items and/or clusters until a single cluster remains. This results in a binary tree whose leaves are the individual items. Figure 2 shows the result of clustering a small sample of maintenance records. The numbers at each leaf are the particular record numbers. The position on the y-axis of the horizontal lines indicates the degree of similarity

when two clusters were merged. At the bottom, close to the leaves, the records and the resultant clusters are similar. As we move up the tree, similarity decreases. In this example there are two main clusters, on the left and right of the dashed line, with smaller internal sub-clusters.

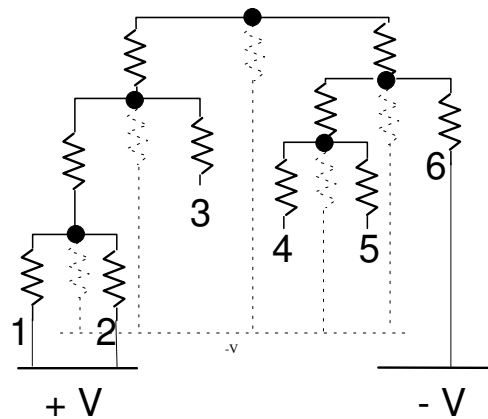


**Figure 2: The Binary Tree.**

The agglomerative clustering algorithm requires a similarity measure, which in this application is based on comparing records field-by-field, term-by-term. For each term in one record, we find the best matching term from the same field in the other record. Although an exact match gives the largest score, lower scores are obtained if terms share a common prefix. This is useful not only for part numbers, where leading digits are typically the most important, but also for free text, where words are often abbreviated by leaving off the endings. Text is further processed by removing “stop words” and any periods occurring in acronyms. Vowels are removed (e.g. VLV = VALVE) as they are often left out in the “short hand” the mechanics use. The total score is then just a count of the number of matching characters. The results are normalized to range from zero to one by using a sigmoid squashing function ( $2*(1-1/(1+1.1^x))$ ). The tree produced by the clustering algorithm is loaded into the system prior to any search. Each internal node, representing a merge point, includes the similarity measure; each leaf includes the corresponding record number.

**Labeling by Spreading Activation.** When the user adds positive and negative labels to records, by clicking the mouse buttons, these are also recorded at the appropriate leaves in the tree. Leaves corresponding to unlabelled records are left unassigned. When the user presses the “Match” button, spreading activation is used through the tree to label both interior nodes and unassigned leaves. The process is analogous to

current flow through a resistor network where the resistor values are determined by the similarity measure. A positive label is equivalent to connecting all positively labeled leaves to a positive voltage source and a negative label to connecting all negatively labeled leaves to a negative voltage source. Figure 3 gives a very simple example, where the user has labeled records 1 and 2 as positive and 6 as negative. The current flow through the resistors determines the voltage at any node and therefore the label for the leaves representing the unlabeled records. In this example, if the similarities are roughly equal we would expect record 3 to be labeled positive and records 4 and 5 negative. Interior nodes also have a weak negative bias, indicated by the dotted connections. This allows classification of records when there are only positive labels and controls the level of generalization.



**Figure 3: A Resistor Network.**

**Generating a Description.** Once all nodes in the tree have been labeled, each subtree containing only positive nodes is used to generate a disjunct in the description. Beginning at the leaves of each subtree, pairs of records are generalized by finding common terms. This is done in exactly the same way as the similarity measure was calculated, such as removing vowels and matching prefixes (e.g. RPL matches REPLACED, more examples are given in section 3). The generalized terms (e.g. RPL) now form a conjunctive rule. The rule is further generalized by combining it with rules from other generalized records, or with other new records directly. The process terminates when the root of the subtree is reached or there is no generalization of sufficient complexity possible (a rule must contain at least 10 characters to prevent trivial matches). In the latter case, rules from lower down the subtree are used. The resulting disjuncts are shown to the user on separate lines of the description table. Once the user is happy with the description and presses the “Query” button, the description is converted into an SQL query which retrieves matching records from the database. If the new description does not expand the search sufficiently, the “Generalization” slider at the top of the interface can be used to decrease the negative bias. This tends to increase the effect of positive labels and therefore generalization.

### 3 Experimental Validation

To experimentally validate the system and to show that some speed up is obtained, a search carried out previously by the author is repeated using the system. The airline had identified a number of components that were felt to be particularly costly in terms of maintenance. Relevant records had been found previously by querying the database directly using SQL queries constructed by hand. The system was built in part with the aim of making this process easier and faster. The rest of this section details the search for records about the replacement of the “low power turbine cooler valve”, typically represented by the acronym LPTC.

On opening the interface, the system displays the nearly 2000 records representing all maintenance records dealing with engine problems. It is somewhat laborious to study so many items and the list is easily reduced by entering a simple description. Two numbers supplied by the airline C25149000, the manufacturer’s part number, and 75-20-0115, the item id, are entered on separate lines in the description table.

C25149000			
	75-20-0115		

The two text fields are left blank. In this discussion, the tables show only numbers for the installed part. Although the numbers can differ for the removed part, they are often the same or missing altogether. So they are not shown for ease of exposition. After pressing the “Query” button, 19 records are retrieved, 3 matching on the manufacturer’s part number and 16 on the item id. So far there is little difference from the process carried out by hand, one minor advantage is that it is not necessary to write an SQL query. Marking all records as positive and pressing the “Match” button produces four disjuncts. The first two correspond largely to the original information entered by the user although they were generated by generalizing actual records.

C25149000	2600000005654		LPT_C
	75-20-0115-00		

One difference is that the first disjunct includes a new Item Id. (2600000005654) along with the manufacturer’s number. This number is actually the airline’s own number rather than the industry standard number we were given. The discovery of such commonly used additional numbers is an important part of understanding the domain. This information did appear in the records returned by hand crafted SQL queries, but the information was buried in the fields of some of the records. The acronym LPT\_C also appears, the underscore indicating a possible vowel. The reason for this is made clear if we look at the free text from two of the 19 records.

**Maintenance Action 1:** REPLACED RH LPTC VLV AS PER AMM FADEC CKC CARRIED OUT

**Maintenance Action 2:** LPTACC CHANGED

The component typically written as LPTC, as in the first example, is sometimes written as LPTACC, as in the second example. This is generalized by dropping the

extra C and allowing a possible vowel as the fourth letter. The second pair of disjuncts are more general than those entered by the user. The first generalizes the item id; the second uses only the free text fields. Matching on all four disjuncts returns 25 records, an additional six potentially relevant ones.

	75-20-011	75-20- ENG R_PL_C	V_LV
		VLV	CH_NG CH_N LPT_CC T_ST

Unfortunately, three of the records refer to the wrong component. This is due to the first disjunct being over-generalized. The Item Id. 75-20-011 is the prefix for both the “high power turbine cooler valve” and the “low power turbine cooler valve”. The text fields do not help. The second free text field identifies a valve (V\_LV), but both components are valves. The next disjunct is not over-generalized, it specifically mentions changing (CH\_NG) the LPTC. When the incorrect records are marked as negative, all others as positive, and the “Match” button pressed, a new set of six disjuncts is returned. Most notably the offending over-generalized disjunct has disappeared. (NB the first two disjuncts continue to appear in the user interface until the end of search but won’t be specifically listed from now on).

C24792000	26000000005654	V_LV	LPT
		75-20- LPTC	CH_CK LPT_C
		75-20- ENG LPT	CH_NG ENG LPTC V_LV
		VLV	CH_NG CH_N LPT_CC T_ST

Here the new Item Id. appears again but now accompanied by a different manufacturer’s part number (C24792000). Again the system has highlighted variations in the numbering system not mentioned by the airline. Pressing the “Query” button retrieves 30 records all referring to the correct component. The success of the other disjuncts is primarily due to including some variant of the acronym LPTC in the last field, although terms like changed (CH\_NG) are important to make sure the component was actually replaced. Marking all records as positive and pressing the “Match” produces 7 disjuncts. The most useful change is the generalization of the first of these disjuncts by relying only on the newly identified numbers and removing all text terms. The extra disjunct (5<sup>th</sup> row) does little to improve search.

C24792000	26000000005654		
		75-20- LPTC	CH_CK LPT_C
		75-20- ENG LPT	CH_NG ENG LPTC V_LV
		VLV	CH_NG CH_N LPT_CC T_ST
		CH_N R_P_RT V_LV	CH_N S_RV T_ST

By pressing “Query” again, 35 items are retrieved. The system did not generalize sufficiently to find more relevant records after marking these 35 as positive. Using the “Generalization” control tended to over-generalize, matching again on the “high power turbine cooler valve”. Generalizing a couple of the existing disjuncts, by manually removing terms from the first text field, retrieved a total of 46 records. Finding more records by removing terms from the problem description may indicate that it was initially wrongly diagnosed and only when a repair was carried out did the



component at fault become clear. The total number of records found was one more than had been identified in the original manual search. But two of the records were incorrect, although they both mentioned an LPTC, one that was changed earlier and one that would be changed later. But overall, the system helped the user quickly find relevant records and identified many of the essential terms that were important in the domain.

#### **4 Limitations and Future Work**

Having the user critique records returned by a query might be viewed as a form of relevance feedback [3,4] used extensively in information retrieval. Another similarity is that many such systems use spreading activation to find potentially interesting documents [8]. The large body of research into relevance feedback will undoubtedly prove to be a useful source of ideas to improve the retrieval of maintenance records.

Clustering has had somewhat mixed results in information retrieval [5]. So one question warranting further investigation is why it has worked in the experiments discussed here and the general information retrieval task. Firstly, the records consist of multiple fields of different types not only free text. Secondly, the text is much more constrained, being limited to a narrow domain, than the text encountered more generally in documents. Thirdly, this paper has strongly stressed the importance of generating a description not normally a concern in information retrieval.

As part the investigation of the effectiveness of clustering, it would be worth looking at how sensitive the system is to the choice of clustering algorithm. This would involve experiments with other commonly used algorithms [7] and perhaps with other more experimental ones [11]. At present, clustering is done only once but it still extracts useful structure that can be used to find relevant records. It might be advantageous, however, if the clustering took into account the labeling information [1]. However, repeating a standard clustering algorithm every time a user labels new records would be computationally very expensive. It would be worth investigating if fast, local modifications to the clusters in response to new labels would be sufficient.

The system found most of the relevant records without direct modification of the description by the user. To find the last 25% of the records, the user experimented with different generalizations of disjuncts already found by the system. Finer "Generalization" control, allowing the user to specify that only particular disjuncts should be generalized, should help this last exploratory stage of the search. So far records have been compared field by field. But quite often information was entered into the wrong field. In addition, particularly with text fields, similar terms might be quite correctly used in different fields. A simple extension would be to look for terms in other fields but reduce the matching score appropriately. This would require a more complex type of description, requiring disjunctions at the term level. But such disjunctions, and perhaps negations, would be useful anyway, producing more compact descriptions which should be more easily understood by the user.

The primary aim of this work is to speed up a user's search. The description is also meant to be an important aid to understanding the domain. The process has been so

far been evaluated by comparing it to prior experience when doing search by hand. More extensive experiments, ideally with many human subjects, would be the ultimate way of verify the goals have been met. The amount of effort needed though is not warranted at present but more experiments by the author with other parts from aircraft engines will be carried out in the near future. In the longer term, experience in working in a completely different domain would give greater confidence in the generality of the approach.

## 5 Conclusions

This paper has demonstrated a system that helps users find relevant records in a database containing a very large number of irrelevant ones. It also provides a description of what makes a record relevant. This helps the user to understand the domain and to be confident that the records found are indeed relevant.

## References

1. S. Al-Harbi and V. Rayward-Smith (2003) The Use of a Supervised k-Means Algorithm on Real-Valued Data with Applications in Health. *Proceedings of the 16<sup>th</sup> International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp 575-581
2. A. Blum and T. Mitchell. (1998) Combining Labeled and Unlabeled Data with Co-training. *Proceedings of the Workshop on Computational Learning Theory*, pp 92-100
3. D. Haines and W. Croft (1993) Relevance Feedback and Inference Networks. *Proceedings of the 16<sup>th</sup> International Conference on Research and Development in Information Retrieval*, pp 2-11
4. D. Harman (1992) Relevance Feedback Revisited. *Proceedings of the 15<sup>th</sup> International Conference on Research and Development in Information Retrieval*, pp 1-10
5. M. Hearst and J. Pedersen (1996). Reexamining the Cluster Hypothesis: Scatter/gather on Retrieval Results. *Proceedings of the 19<sup>th</sup> ACM SIGIR International Conference on Research and Development in Information Retrieval*, pp 76-84
6. T. Joachims (1999) Transductive Inference for Text Classification using Support Vector Machines. *Proceedings of the 16<sup>th</sup> International Conference on Machine Learning*, pp 200—209
7. L. Kauffman and P. Rousseeuw (1989) *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley
8. G. Salton and C. Buckley (1998) On the use of spreading activation methods in automatic information. *Proceedings of the 11<sup>th</sup> International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp 147-160
9. S. Létourneau, F. Famili and S. Matwin, (1999) Data mining for Prediction of Aircraft Component Replacement. *IEEE Intelligent Systems and their Applications* 14, 6, pp 59-66
10. K. Nigam, A. McCallum and S. Thrun and T. Mitchell (2000) Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39, 2/3, pp 103—134
11. Z. Su, L. Zhang and Y. Pan (2003) Document Clustering Based on Vector Quantization and Growing-Cell Structure. *Proceedings of the 16<sup>th</sup> International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp 326-336