Multi-label large margin hierarchical perceptron

Clay Woolam* and Latifur Khan

University of Texas at Dallas,
800 West Campbell Road, Richardson,
75080, Texas, 972-883-4137, USA
E-mail: clay@woolam.org
E-mail: lkhan@utdallas.edu
*Corresponding author

Abstract: This paper looks into classification of documents that have hierarchical labels and are not restricted to a single label. Previous work in hierarchical classification focuses on the hierarchical perceptron (Hieron) algorithm. Hieron only supports single label learning. We investigate applying several standard multi-label learning techniques to Hieron. We then propose an extension of the algorithm (MultiHieron) that significantly outperforms all previously mentioned techniques. MultiHieron has a new aggregate loss function for multiple labels. Improvement is shown on the Aviation Safety Reporting System (ASRS) flight anomaly database and OntoNews corpus using both at and hierarchical categorisation metrics.

Keywords: classification; semantic web; multi-label; multi-class; anomaly; aviation; safety; hierarchy; ontology; perceptron; large margin; data mining.

Reference to this paper should be made as follows: Woolam, C. and Khan, L. (2008) 'Multi-label large margin hierarchical perceptron', *Int. J. Data Mining, Modelling and Management*, Vol. 1, No. 1, pp.5–22.

Biographical notes: Clay Woolam is a PhD student at the University of Texas at Dallas. Currently, he works in the Semantic Web Lab creating extremely large distributed RDF storage systems. He has previously done research in data anonymisation and privacy tasks. He received his BS in Electrical Engineering from the University of Texas at Dallas in 2006.

Dr. Latifur Khan is currently an Associate Professor in the Computer Science Department at the University of Texas at Dallas where he has taught and conducted research since 2000. He received his PhD and MS in Computer Science from the University of Southern California in 2000 and 1996 respectively. He obtained his BSc in Computer Science and Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 1993. His current research areas include data and multimedia mining and semantic web.

1 Introduction

Classification is a method of assigning a label to some target given relevant information about the target. A classification system can generate models using information from the world to perform classification. Many classification systems have been constructed to suit

different machine learning tasks. This paper deals with two specific classification problems that are not normally addressed: multi-label and hierarchical label classification.

A hierarchical label is one that can be ordered in some sort of tree structure. In our case, we consider labels that explicitly state the tree structure. For example, lets make up two labels, say 'A: B: C' and 'A: B: D'. This would form a tree with the root node, 'A', a single child of the root, 'B' and two children for 'B', 'C' and 'D'. These labels can be treated used in systems that do not utilise their hierarchical structure but they contain additional information that has been shown to produce good results in specialised algorithms.

In multi-label systems, a document can have any number of labels attached to it. Most systems only accept one label per training document. For example, a news article can only be in one section of the newspaper. However, there are many real life sources that have multiple labels. If we attempt to train a document with multiple labels, the system may become confused and the performance will suffer as a result. For example, the NASA Aviation Safety Reporting System (ASRS) flight anomaly database is multi-label, with an average of 2.7 labels per anomaly report.

The ASRS flight anomaly database consists of free-form text documents. Each document can have multiple labels that reside to a structured hierarchy. This hierarchy is three levels deep containing 13 major classes and 55 minor classes. The following is an example anomaly report to illustrate the difficulty in evaluating these flight reports. This report contains three labels: 'aircraft equipment problem: less severe', 'other spatial deviation' and 'non-adherence: clearance'.

> "Cleared direct private very high frequency omni directional radio range after takeoff bos. Using right navigation flight management system and omega bos center advised we missed private very high frequency omni directional radio range by 20 miles. Upon checking found both flight management system and omega in gross error..."

The most recent and relevant work in hierarchical classification comes from Dekel et al. (2004). Dekel et al. (2004) propose a large margin hierarchical perceptron algorithm. It was shown to produce good results at entire document classification when document class labels were hierarchical in nature. Previous work with this algorithm focused only on single label learning. It has not been shown to work well if more than one class label exists. We extend this to do multiple label anomaly classification by performing simultaneous updates.

In our proposed algorithm, the system performs multi-label training. When the system is presented with a document, it returns the list of all possible labels attached to weights. Typically, we would simply take the highest weighted member of this list as our predicted label. To do multi-label training, we use the highest n weighted labels and compare this to the true set of labels attached to the document (Woolam and Khan, 2008). Large margin classifiers make use of loss function that reports the severity of error made by a given prediction. The Hieron loss function is hierarchical, it took the labels tree structure into account, but not multi-label. It could only do one-to-one loss reporting. We develop a many-to-many loss function that takes into account hierarchical information. Updates are performed simultaneously. We then study this algorithm operating in batch mode including other alterations for better performance.

The paper is organised in the following way. Section 2 discusses previous work in multi-label classification and large margin classifiers. Section 3 describes the traditional Hieron algorithm as has been explored in previous work. Section 4 describes a new, more general algorithm, capable of learning multiple concepts at a time for one document. Section 5 discusses batch operation of the algorithm with performance modifications. Section 6 discusses the database and provides evidence that learning with the new algorithm improves on learning in the single document case. Section 7 discusses conclusions and explores future possibilities for this research.

2 Related work

This paper builds on a major body of work from several different areas in classification, particularly a large margin hierarchical perceptron presented in Dekel et al. (2004). The study begins with looking at hierarchical label classification, and then we explore large margin classifiers and performing large margin hierarchical label classification and finally analysis of current known approaches to multi-label classification. Much work has been done in hierarchical classification, including (Koller and Sahami, 1997; McCallum et al., 1998; Weigend et al., 1999; Dumais and Chen, 2000). Many approaches to hierarchical classification have focused on rephrasing their problem so that it could be implemented using already existing algorithms. A classifier could be trained for each vertex in the hierarchy and then some additional mechanism could be implemented to account for linkages between points in the hierarchy.

Large margin classifiers had become popular and had received major attention (Vapnik, 1998). Early large margin work (Herbster, 2001; Crammer et al., 2003) was very successful and provided a framework for which to build algorithms. Dekel et al. (2004) applied these large margin design principals to a hierarchical label system to create a large margin hierarchical perceptron called Hieron. They decided that their algorithm needed to be lightweight and efficient. Due to hierarchical nature, errors can be treated differently depending on proximity to the true label. Predicting a sibling would not be as bad as predicting a node in a completely separate part of the tree. Online large margin classification systems have defined margin constraints by which the systems hypothesis must comply. Updates are made such that their effect is minor. Online large margin classifiers can be converted to batch classifiers (Cesa-Bianchi et al., 2004). Li and Bontcheva (2007) demonstrated promising results by applying this algorithm to ontology-based information extraction. Classifiers can be used to perform information extraction (Freitag, 1999). Large margin hierarchical perceptron was shown to outperform perceptron with uneven margins and uneven margins support vector machine, which produced good results (Li et al., 2005). Hierarchical classifiers have specific performance analysis metrics (Maynard et al., 2006). None of these considers training for multi-label data. On the other hand, our approach is a simultaneous multi-label update strategy for a large margin hierarchical perceptron.

A holistic overview of current multi-label classification techniques was outlined by Tsoumakas and Katakis (2007). In the case of non-binary classifiers, we can perform any one of a number of dataset transformations. It is explicitly stated that they do not deal with multi-label hierarchical classification problems. Two of their transformations were useful to us. One method only uses training instances that are attached to one level. This reduces the size of the dataset. We call this the single label (SL) method. In another

method, a multi-label document with n labels can be treated as if it were n separate single-label documents. We call this the serial method.

3 Hierarchical perceptron algorithm

The large margin hierarchical perceptron algorithm is named Hieron. In this section, we will briefly describe the algorithm as it was first presented as a fully developed online Hieron algorithm. This algorithm was designed for single-label classification. To describe the algorithm, we first establish some definitions. Labels are hierarchical and each has a corresponding weight that is a vector of real numbers, $\mathrm{w}^v \in \mathbb{R}^n$ where $v \in Y$. We superscript the weight vector with the label name to denote a label's weight. We call *Y* our set of all class labels. To refer to a set of all class labels from the top of the hierarchy to a given label we say *P*(v). We classify a vector of real numbers called x. Finally, predictions are made according to the rule in equation (1). For every possible label, we sum all of the weights from the top of the hierarchy to that label, then we multiply it by our instance. The largest summation is the prediction. We call the numerical value produced from this summation the corresponding prediction weight.

$$f(x) = \arg\max_{v \in Y} \sum_{u \in P(v)} \mathrm{w}^u \cdot \mathrm{x} \tag{1}$$

Next, we need to train the weights in the system. Initially they are set to zero. At each time step i, we query the system for a prediction, $\hat{y}$, of instance x. The error is the 'tree distance', or the shortest path from a predicted label to its corresponding true label. Error is denoted by $\gamma(y, \hat{y})$. Equation (2) defines margin conditions for the weight vectors. These margin conditions state that there exists a set of weights, $\{\omega^v\}_{v \in y}$, such that for all instance-label pairs, x and y, in the training set the difference between our prediction weight of our correct label, y and our predicted label, $\hat{y}$, will be greater than the square root tree error. We denote r to be any possible label.

$$\sum_{v \in P(y_i)} \omega^v \cdot \mathrm{x}_i - \sum_{u \in P(r)} \omega^u \cdot \mathrm{x}_i \geq \sqrt{\gamma(y_i, r)} \tag{2}$$

From this, we have a loss function that forms the basis for our update,

$$L(\{\mathrm{w}^v\}, \mathrm{x}_i, y_i) = \sum_{v \in P(y_i)} \mathrm{w}^v \cdot \mathrm{x}_i - \sum_{u \in P(r)} \mathrm{w}^u \cdot \mathrm{x}_i + \sqrt{\gamma(y_i, r)} \tag{3}$$

Algorithm 1 is the basic online Hieron algorithm that has been subject to formal analysis. In line 1, we iterate through our entire training set, which has m training instances. On each iteration, we have a new (w_i,y_i). A prediction is made in line 3. If it is correct, no update is made on lines 7 or 10, if false, then only relevant nodes are updated. Updates are made using an update rule derived by Dekel et al. (2004). This update rule uses a loss function from equation (3). Using Lagrange multipliers, we find the optimal update

policy, denoted above by alpha. This process is described in Section 4.1 using different margin conditions.

Algorithm 1 Online Hieron

Require: $\forall v \in Y : \mathrm{w}_0^v = 0, i = 0$

1 for $i = 1$ to m do

2 {training instance x_i and corresponding label y_i }

3 $\hat{y} \leftarrow \arg\max_{y \in Y} \sum_{v \in P(y)} \mathrm{w}_i^v \cdot \mathrm{x}_i$

4 $\alpha_i \leftarrow \dfrac{L(\{\mathrm{w}_i^{\mathrm{v}}\}, \mathrm{x}_i, y_i)}{\gamma(y_i, \hat{y}) \left\| \mathrm{x}_i \right\|^2}$

5 for v in Y do

6 if $v \in P(y) \setminus P(\hat{y})$ then

7 $\mathrm{w}_{i+1}^v \leftarrow \mathrm{w}_i^v + \alpha_i \mathrm{x}_i,$

8 end if

9 if $v \in P(\hat{y}) \setminus P(y)$ then

10 $\mathrm{w}_{i+1}^v \leftarrow \mathrm{w}_i^v + \alpha_i \mathrm{x}_i,$

11 end if

12 end for

13 end for

4 Multi-label hierarchical perceptron

The Hieron algorithm has shown promising results for the SL classification problem. There are many cases where we need to do multi-label classification. In the semantic web, for instance, entities are expected to have multiple relationships between each other. Hieron performs prediction by generating a prediction weight for all labels. Instead of using the single highest prediction weight, we simply use the top n.

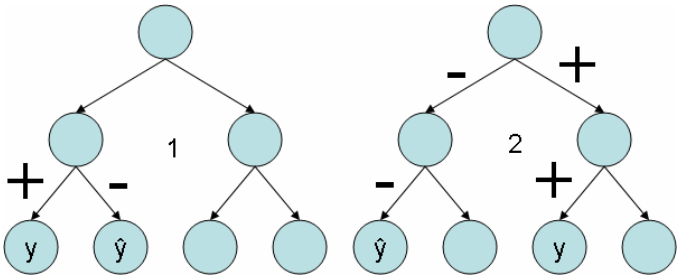
To illustrate training approaches, consider the following three methods of training a dataset that has the characteristics of being multi-label and hierarchical with Algorithm 1.

- Hieron-SL method will only train SL documents. This method will not work if there are no documents which are attached to a single label in the database.
- Hieron-serial method can be used to a train multi-label documents described by Algorithm 1. Intuition indicates problems with this method. The original algorithm was attempting to solve a problem framed in terms of one prediction label for one true label, not multiple labels, so our update rule may not work well. Also, we could

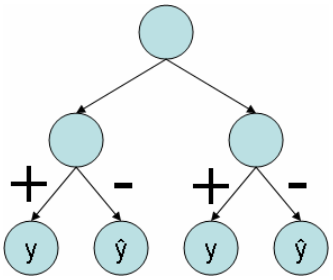
have a case where we erase important information when updates are made for different labels on the same document as the following example shows.

In Figure 1, we have a document that has two labels. Figure 1(a) shows how we may have an undesirable update case by repeating predictions in serial mode. The bottom right node represents the correct label (y) and its sibling represents the predicted node ($\hat{y}$). This is incorrect, so the weights on the true node will be rewarded (denoted by +) while the weights on the prediction will be punished (denoted by –). Because they share the same parent, no other nodes need be updated. When queried for another prediction, it makes what would be the correct label in Step 1, but is the incorrect label for Step 2. Here, the left-most leaf node and second right most leaf node are predicted and the true class, respectively [tree in Figure 1(a)]. Updating now cancels out half of the information we added to the system in Step 1. It is also expensive, six nodes have been updated in the process of training one document.

Figure 1 Example of learning weights for the same document containing two labels, (a) serial Hieron learning (b) MultiHieron learning (see online version for colours)



(a)



(b)

Figure 1(b) breaks from conventional models to provide a simultaneous update strategy. Only four updates are made, on each of leaf node in the tree. Both predictions are incorrect, but appear as siblings of the true values, therefore only those nodes need be adjusted and the rest of the tree remains untouched. In other words, left most leaf and second right most leaf nodes are rewarded; second most leaf and right most leaf nodes are punished. As the following sections will demonstrate, not only does this method intuitively feel better, but also it is advantageous in several ways. By updating everything in one shot, we are taking into account all information available for a more complete result. By making multiple predictions at once, we end up making fewer weight updates.

4.1 Proposed algorithm: MultiHieron

To develop a better algorithm, the problem must be redefined in multi-category terms. The new algorithm will assume that, for all instance-label pairs (x_i, y_i) in the training set and $\mathrm{r} \subset Y$ such that at least one $r_j \in Y \setminus \mathrm{y}_i$, the following margin requirements hold

$$\sum_{z \in \mathrm{y}_i} \sum_{v \in P(z)} \omega^v \cdot \mathrm{x}_i - \sum_{q \in \mathrm{r}} \sum_{u \in P(r)} \omega^u \cdot \mathrm{x}_i \geq \sqrt{\gamma(\mathrm{y}, \mathrm{r})} \tag{4}$$

From here and taking a similar approach to the method used by Dekel et al. (2004), we will craft MultiHieron, the multi-category hierarchical perceptron algorithm. First, notice that our tree distance function has changed characteristics slightly. We now denote $\gamma(\mathrm{V}, \mathrm{U})$ to be

$$\gamma(\mathrm{V}, \mathrm{U}) = \Big| \bigcup_{v \in \mathrm{V}} P(v) \Delta \bigcup_{u \in \mathrm{U}} P(u) \Big| \tag{5}$$

Note that Δ denotes the symmetric difference ($A \Delta B = (A \backslash B) \bigcup (B \backslash A)$) between sets. For example, Figure 1 shows three trees. The size of symmetric difference between the paths to the two points is 2 for the top left and 4 for both the top right and bottom trees. Following this, we define our multi-label loss function to be

$$L(\{\mathrm{w}_\mathrm{i}^\mathrm{v}\}, \mathrm{x}_i, \mathrm{y}_i) = \sum_{y \in \mathrm{y}_i} \sum_{v \in P(y)} \mathrm{w}_i^v \cdot \mathrm{x}_i - \sum_{r \in \mathrm{r}} \sum_{u \in P(r)} \mathrm{w}_i^u \cdot \mathrm{x}_i + \sqrt{\gamma(\mathrm{y}_i, \mathrm{r})} \tag{6}$$

To control learning by ensuring that the margin requirement is met but also keeping the updated weights close to the previous, we follow the method provided by Dekel et al. (2004) and say,

$$\min_{\{\mathrm{w}^v\}} \frac{1}{2} \sum_{v \in Y} \left\| \mathrm{w}^v - \mathrm{w}_i^v \right\|^2 s.t. \sum_{y \in \mathrm{y}_i} \sum_{v \in P(y)} \mathrm{w}^v \cdot \mathrm{x}_i - \sum_{\hat{y} \in \hat{\mathrm{y}}_i} \sum_{u \in P(\hat{y})} \mathrm{w}^u \cdot \mathrm{x}_i \geq \sqrt{\gamma(\mathrm{y}_i, \hat{\mathrm{y}}_i)} \tag{7}$$

We can solve this condition using Lagrange multipliers, α_i and perfrom some optimisation to get:

$$\alpha_i = \frac{L(\{\mathrm{w}_i^v\}, \mathrm{x}_i, \mathrm{y}_i)}{\gamma(\mathrm{y}_i, \hat{\mathrm{y}}_i) \left\| \mathrm{x}_i \right\|^2} \tag{8}$$

Algorithm 2 is similar to Algorithm 1. We iterate through every one of the m instances in the training set in line 1. During each iteration, line 3 gets $\left|\mathrm{y}_i\right|$ predictions and lines 7 and 10 perform weight updates if necessary. Line 3 generates multiple prediction labels, instead of a SL. Line 4 is significantly different where we use equation 6 instead of equation 3. Our update rule contains a different path symmetric difference formula, equation 5. Also, we update each relevant prediction and true class group in lines 7 and 10.

Algorithm 2 Online MultiHieron

Require: $\forall v \in Y : \mathrm{w}_0^v = 0, i = 0$

1 for $i = 1$ to m do

2 {training instance x_i and corresponding label y_i }

3 $\hat{\mathrm{y}} \leftarrow \arg\max_{y \in Y}(|\mathrm{y}_i|) \sum_{v \in P(y)} \mathrm{w}_i^v \cdot \mathrm{x}_i$

4 $\alpha_i \leftarrow \frac{L(\{\mathrm{w}_\mathrm{i}^\mathrm{v}\}, \mathrm{x}_i, \mathrm{y}_i)}{\gamma(\mathrm{y}_i, \hat{\mathrm{y}}) \|\mathrm{x}_i\|^2}$

5 for v in Y do

6 if $v \in \bigcup_{\mathrm{y} \in \mathrm{y}_i} P(\mathrm{y})$ then

7 $w_{i+1}^v \leftarrow w_i^v + \alpha_i \mathrm{x}_i,$

8 end if

9 if $v \in \bigcup_{\hat{\mathrm{y}} \in \hat{\mathrm{y}}_i} P(\hat{\mathrm{y}})$ then

10 $w_{i+1}^v \leftarrow w_i^v + \alpha_i \mathrm{x}_i,$

11 end if

12 end for

13 end for

4.2 Analysis of MultiHieron

MultiHieron is more general than the original. It can be trivially shown that MultiHieron will reduce to Hieron on any SL categorisation problem. If y_i and r have maximum size of 1 for all i, then $\gamma(\mathrm{y}_i, \mathrm{r}) = \gamma(y_i, r)$, the symmetric difference of the path to only two labels. All training, updates and predictions will remain the same.

Dekel et al. (2004), provide a theorem that implied that the cumulative loss suffered by online Hieron is bounded as long as the margin requirements are satisfied. In Theorem 4.1, we show that this also holds for MultiHieron.

Theorem 4.1: Let $\{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^m$ be a sequence of examples where $\mathrm{x}_i \in \mathbb{R}^n$ and $\mathrm{y}_i \in Y$. Assume there exists a set $\{\omega^u : \forall v \in y\}$ that satisfies the margin condition for all $1 \le i \le m$. Then, the following holds,

$$\sum_{i=1}^{m} L^2(\{\mathrm{w}_i^v\}, \mathrm{x}_i, \mathrm{y}_i) \le \sum_{v \in Y} \left\|\mathrm{w}^v\right\|^2 \gamma_{max} R^2 \tag{9}$$

where for all $i, \|\mathrm{x}_i\| \le R$ and $\gamma(\mathrm{y}_i, \hat{\mathrm{y}}_i) \le \gamma_{max}$.

Proof: Define $\bar{\omega}$ to be a concatenation of all vectors in $\{\omega^v\}$ and, likewise, $\bar{\mathrm{w}}_i$ in the same manner. The squared distance, δ_i is

$$\delta_i = \left\|\bar{\mathrm{w}}_i - \bar{\omega}\right\|^2 - \left\|\bar{\mathrm{w}}_{i+1} - \bar{\omega}\right\|^2$$

Now we can get upper bounds and lower bounds on δ_i over all i by,

$$\begin{aligned}\sum_{i=1}^{m}\delta_i &= \sum_{i=1}^{m}\left\|\bar{\mathrm{w}}_i - \bar{\omega}\right\|^2 - \left\|\bar{\mathrm{w}}_{i+1} - \bar{\omega}\right\|^2 \\ &= \left\|\bar{\mathrm{w}}_1 - \bar{\omega}\right\|^2 - \left\|\bar{\mathrm{w}}_m - \bar{\omega}\right\|^2 \\ &\le \left\|\bar{\mathrm{w}}_1 - \bar{\omega}\right\|^2 \\ &\le \left\|\bar{\omega}\right\|^2 = \sum_{v\in Y}\left\|\omega^v\right\|^2\end{aligned}$$

Thus, we have our upper bound on all $\sum_{i\in[1,m]}\delta_i$.

To get our lower bound, use the minimiser of equation (7) producing a result $\bar{\mathrm{w}}_{i+1}$. Using a theorem (Censor and Zenios, 1997)(Theorem 2.4.1), we have the following inequality,

$$\left\|\bar{\mathrm{w}}_i - \bar{\omega}\right\|^2 - \left\|\bar{\mathrm{w}}_{i+1} - \bar{\omega}\right\|^2 \ge \left\|\bar{\mathrm{w}}_i - \bar{\mathrm{w}}_{i+1}\right\|^2$$

So, $\delta_i \ge \left\|\bar{\mathrm{w}}_i - \bar{\mathrm{w}}_{i+1}\right\|^2$. After updates, only weights w_i^v are updated if $v \in \bigcup_{y\in \mathrm{y}_i} P(y) \triangle \bigcup_{\hat{y}\in\hat{\mathrm{y}}_i} P(\hat{y})$ which means,

$$\left\|\bar{\mathrm{w}}_i - \bar{\mathrm{w}}_{i+1}\right\|^2 = \sum_{y\in Y}\left\|\mathrm{w}_i - \mathrm{w}_{i+1}\right\|^2 = \sum_{v\in \bigcup_{y\in \mathrm{y}_i} P(y) \triangle \bigcup_{\hat{y}\in\hat{\mathrm{y}}_i} P(\hat{y})} \left\|\mathrm{w}_i - \mathrm{w}_{i+1}\right\|^2$$

Now we can use the update rule to get,

$$\sum_{v\in \bigcup_{y\in \mathrm{y}_i} P(y) \triangle \bigcup_{\hat{y}\in\hat{\mathrm{y}}_i} P(\hat{y})} \left\|\mathrm{w}_i - \mathrm{w}_{i+1}\right\|^2 = \sum_{v}\alpha_i^2\left\|\mathrm{x}_i\right\|^2 = \gamma(\mathrm{y}_i, \hat{\mathrm{y}}_i)\alpha_i^2\left\|\mathrm{x}_i\right\|^2$$

Plugging in equation (8),

$$\delta_i = \frac{L^2(\{\mathrm{w}_i^v\}, \mathrm{x}_i, y_i)}{\gamma(\mathrm{y}_i, \hat{\mathrm{y}}_i)\left\|\mathrm{x}_i\right\|^2} \ge \frac{L^2(\{\mathrm{w}_i^v\}, \mathrm{x}_i, y_i)}{\gamma_{max}R^2}$$

So,

$$\sum_{i=1}^{m}\frac{L^2(\{\mathrm{w}_i^v\}, \mathrm{x}_i, y_i)}{\gamma_{max}R^2} \le \sum_{i=1}^{m}\delta_i \le \sum_{v\in Y}\left\|\omega^v\right\|^2$$

Finally, we distribute variables around,

$$\sum_{i=1}^{m} L^2(\{\mathrm{w}_i^v\}, \mathrm{x}_i, \mathrm{y}_i) \leq \sum_{v \in Y} \left\| \omega^v \right\|^2 \gamma_{max} R^2$$

4.3 Batch MultiHieron with performance enhancements

We have presented and analysed an online large margin classifier. It has been shown that conversion from online to batch mode produces better results (Dekel et al., 2004). Operation in batch mode gives no guarantee for margin conditions that were the basis for our update rule (Cesa-Bianchi et al., 2004). First, we use time averaged weight vectors in prediction. Once training is complete for m documents, vector weights are represented as equation (10).

$$\mathrm{w}^u = \frac{1}{m+1} \sum_{i=1}^{m+1} \mathrm{w}_i^v \tag{10}$$

Previous batch Hieron implemented average weight system and changed the training goal. The batch Hieron trained the labels that maximised the loss function, equation (3), instead of updating the label that would be predicted. This increases the margin drastically but does not work well in any multi-label operation. In addition, doing this was very computationally expensive for multi-label considerations ($O(m^n)$ where m is total number of labels available and n is number of labels attached to the document). We changed the definition of error, equation (5), in Algorithm 2 to follow a pairwise cost maximisation system. We train using the labels that give us the maximum combined error. This can be found using an $O(n^2)$ dynamic programming algorithm. We can construct a matrix, A, where columns represent predicted nodes and rows represent true nodes. For each cell, we place the cost value from the corresponding predicted node and column node. Then we can create another matrix, B and copy row 1 from the first matrix. Now at each successive row, we chose a node that gives maximises cost that is in our not used yet list, L. $B[i][j] = B[i][j-1] + max_{k \in L[i]} \{A[i][k]\}$. At the end, we choose the sequence that gives the maximum value. Looking back at Figure 1(b), pairwise maximum distance will produce a value of 8. We call algorithm resulting from these modifications batch MultiHieron.

5 Experimentation

We used two datasets to test our algorithms, the ASRS flight anomaly database and the OntoNews corpus. Both datasets have the property of being hierarchical and multi-label. Table 1 shows the properties of both of these datasets. They are similar; ASRS has 2.71 average labels per instance while Ontonews shows 2.95 and about 20% of the instances in both sets only have one label. These datasets heavily contrast in that ASRS database documents, written by flight attendants, pilots, mechanics, etc., are very chaotic and free form, while news articles, written by professional journalists, are objective and structured. In describing both datasets, we use the same naming convention to refer to specific labels. The root of the hierarchy is taken for granted and not explicitly stated.

Our convention is 'level 1 node: level 2 node: ...: level n node' where n is the distance from the root to the label.

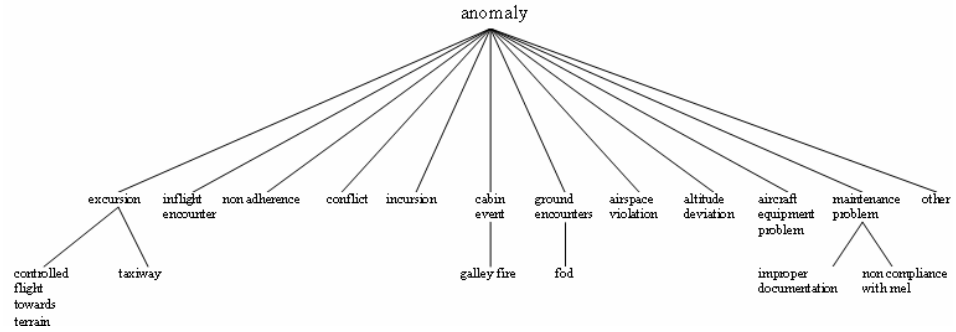
Table 1 ASRS database and OntoNews corpus

	ASRS	*OntoNews*
Training instances	20,000	2,400
Testing instances	10,000	1,076
Features per instance	3,814	13,085
Minimum labels per instance	1	1
Maximum labels per instance	10	12
Average labels per instance	2.71	2.95
Number of SL training instances	3,961	667
Maximum depth of hierarchy	3	9
Number of distinct document labels	55	288

5.1 ASRS anomaly reports dataset

The ASRS flight anomaly database is a repository of voluntary, confidential safety information provided by aviation personnel of all ranks, including pilots, controllers, mechanics, flight attendants and dispatchers. The database includes almost 150,000 incident reports submitted over more than 30 years. It has two major characteristics that distinguish it from other datasets: a document may have multiple anomalies (multi-labels) and each label belongs in a structured hierarchy. The difficulties associated with categorising the documents as highly unstructured free form texts was addressed previously, however, they lost a significant amount of highly relevant information by neglecting the underlying hierarchical structure of the categorisation set. The ASRS database has 13 major classes, which are fairly general observations such as 'inflight encounter' or 'conflict', followed by 55 subclasses, which are much more specific. Figure 2 shows all 13 major classes and six of the 55 subclasses. A report might be labeled as both 'inflight encounter: weather' and 'cabin event: passenger misconduct' if, say, it was a report about a passenger acting particularly angrily about the weather disturbing his flight.

Figure 2 Subset of ASRS database hierarchy



Note: Out of 55 third level nodes, six are shown.

In experimentation, a subset of the documents was chosen at random and feature vectors were generated using the tf-idf system. These anomaly reports were extremely noisy and full of complex acronyms, typos and abbreviations. Acronyms were expanded using PLADS (Barrientos et al., 2007) and case was ignored. Our feature space has been restricted to 3,814 distinct words which were chosen by information gain analysis.

5.2 OntoNews corpus

The OntoNews corpus is a collection of 290 annotated news reports. Annotations are made by marking the beginning and end of a target word or phrase. This corpus has been previously been the focus of study with Hieron (Maynard et al., 2006; Li and Bontcheva, 2007). These were annotated using the Proton ontology. The Proton ontology is designed to capture most high-level semantic concepts. Concepts begin with a root node 'entity' followed by second level 'happening', 'abstract' and 'object'. As levels get deeper, concepts get more and more narrow, eventually getting specific with instance labels like 'beach', 'quarter' and 'stock exchange'. Take this example sentence pulled from one of the reports: 'planes' flying low can pick out targets, including tanks and whatever the 'Taliban' has left in terms of military hardware. Planes and Taliban are labelled objects. Planes is labelled as 'entity: object: vehicle' and Taliban as 'entity: object: agent: group: organisation : religious organisation'.

Because there are only 290 documents and each document has a large number of labels, we instead use the paragraphs of each document as our instance. Each ASRS anomaly report is about a paragraph long so this creates a similar dataset. We generate tf-idf vectors for each paragraph and treat all annotations as labels. For example, in the sentence above, 'entity: object: vehicle' and 'entity: object: agent: group: organisation: religious organisation' would be labels for that sentence. As Table 1 shows, the dataset was divided into 2,400 training instances and 1,076 test instances. No restriction on the number of tokens was made. All 13,085 distinct words in the entire corpus were used.

5.3 Experimental set-up

There are two methods to test prediction accuracy. We can test with knowledge of the amount of predictions we need using a truncation method of result comparison. If a test document has two labels attached to it, we compare the top two predictions. Without knowledge of the number of classes, we need to predict, a threshold-based method generates the best list of prediction candidates. In both systems, we pass an instance x_i and get back a list of all labels and their corresponding weights generated for that instance (z). We then normalise these weights $(\hat{\mathrm{z}})$. Finally, we call our threshold, *T*, the point that immediately passes the sum of all true weights.

$$\sum_{p \in \mathrm{p}_t} p \leq T \tag{11}$$

Machine learning algorithm performance historically employs flat metrics. Precision is a classification metric that reports the amount of correct documents that were retrieved. Recall reports the correctness of the classifications. If we predict every possible document choice, then we would have 100% precision, but 0% recall. F-measure

combines precision and recall by using the weighted harmonic mean of the two values. Accuracy is the ratio of correct cases to all possible cases.

In addition to at permanence metrics, we use a hierarchical performance metric called BDM (Maynard et al., 2006). It is defined as follows: given key node *K* and response node R, BDM is computed as

$$BDM(K,R) = \frac{BR * CP / n0}{BR * CP / n0 + DPK / n2 + DPR / n3} \tag{12}$$

where

- BR (branching factor) is the average number of branches between the most specific common abstraction (MSCA) and the key node and response node normalized by average branching factor for the entire hierarchy.
- CP is the shortest path length from root node to MSCA.
- DPK(R) is the shortest path from MSCA to K(R).
- n0 is the average chain length of the whole hierarchy.
- n2 is the average length of all chains containing K from root.
- n3 is the average length of all chains containing R from root.

Augmented precision [AP, equation (13)] and recall [AR, equation (14)] gives a better overall picture of hierarchical algorithm performance. This accounts for proximity to the true value in the hierarchy and structure of the hierarchy. In flat classification, we assign a binary value to each prediction. That is to say, it is either correct, in which case we assign a value of 1, or it is not, assign a value of 0. In the hierarchical case, metrics like these allow us to better see what is going on inside the system as it learns. BDM will still give a value of 1 if correct, but also return a real number < 1 that takes into account the dynamics of the tree and the location of correct and predicted values on an error. For example, BDM calculation for graph 1 in Figure 1(a) is as follows. Our $BR = 1.43$, $CP = 2$, $DPK = 1$, $DPR = 1$, $n0 = 3$, $n2 = 1$ and $n3 = 1$. This gives a BDM of 0.32, according to equation (12). In graph 2, we have a different *CP* = 1, *DPK* = 2 and $DPR = 2$, therefore BDM is 0.11.

$$AP = \frac{BDM}{BDM + FP} \tag{13}$$

$$AR = \frac{BDM}{BDM + FN} \tag{14}$$

5.4 Results

In the following results, we present values generated using online Hieron applied to datasets with SL and serial transformation, online MultiHieron and batch MultiHieron. The batch Hieron algorithm performed worse than online Hieron in all cases, so results are not presented. The truncation method serves as an easy to understand performance base. Table 2 shows values for F-measure and accuracy over all datasets. For each label, precision and recall values were generated. If target label was predicted and it was

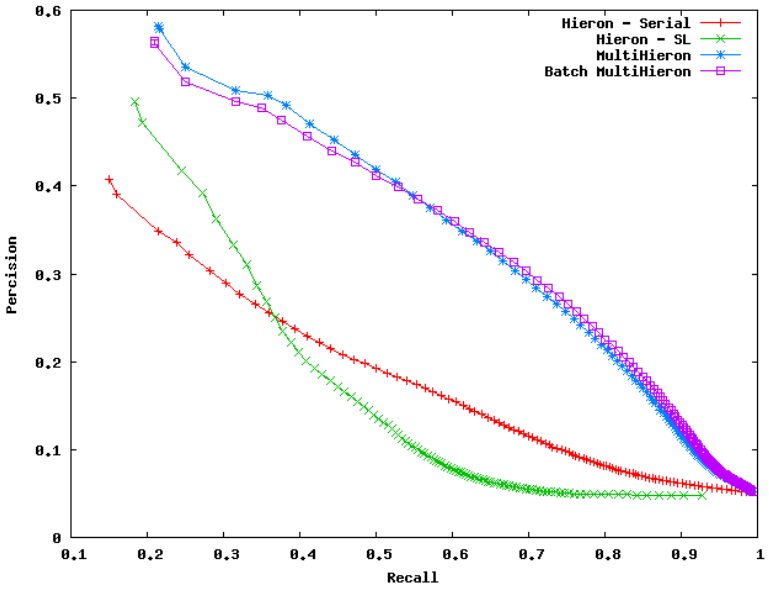
correct, that counts as a true positive for target label, else a false positive for the label. If it was not predicted but it was supposed to be, this is a false negative. If there are n labels in total and a document have three labels. There are three positives and $n - 3$ negatives.

Table 2 Performance comparison of Hieron (SL and serial), MultiHieron and batch MultiHieron

	Hieron-SL	*Hieron-serial*	*MultiHieron*	*Batch MultiHieron*
ASRS F-measure	0.346	0.344	0.495	0.501
ASRS accuracy	93.5	93.5	95.0	95.4
OntoNews F-measure	0.186	0.222	0.398	0.463
OntoNews accuracy	96.7	96.8	97.5	97.8

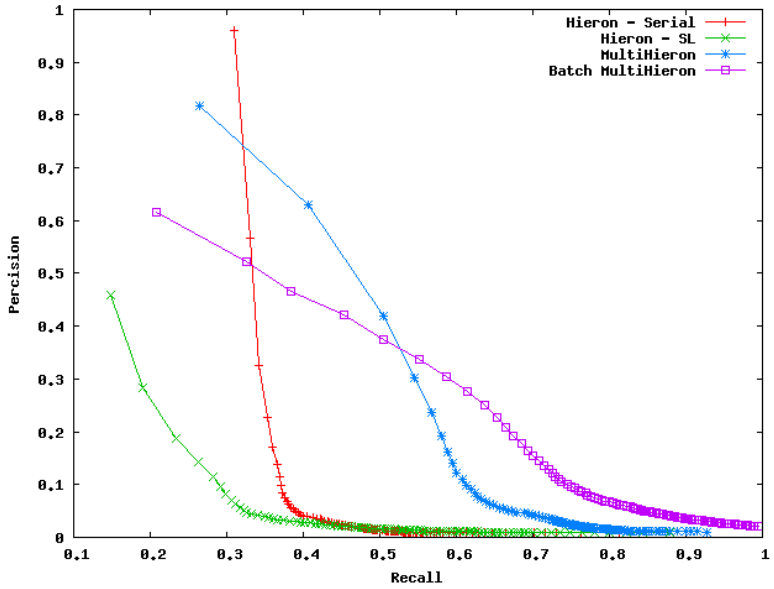
We vary the threshold, *T*, from 0 to 1 in 0.01 increments. *T* distinguishes positive predictions from negative predictions. We can use this to calculate a precision/recall curve. At each point in the graph, if we chose that corresponding *T* value, we get the given precision and recall. Figure 3(a) shows us such a curve for all algorithms on the ASRS dataset. Let us examine the graph by looking at the points where recall is 0.6. Precision is 0.35 for batch MultiHieron, 0.35 for MultiHieron, 0.15 for Hieron-serial and 0.07 for Hieron-SL. We can conclude that the MultiHieron does a significantly better job at classifying documents in this database. Compare with OntoNews, our base set, in Figure 3(b) where a similar behaviour is exhibited. At recall of 0.4, batch MultiHieron has a precision of 0.48, MultiHieron has precision of 0.7 while Hieron-single and Hieron-SL bottom out to 0.02.

Figure 3 Precision versus recall – at metric for performance, (a) ASRS database (b) OntoNews corpus (see online version for colours)



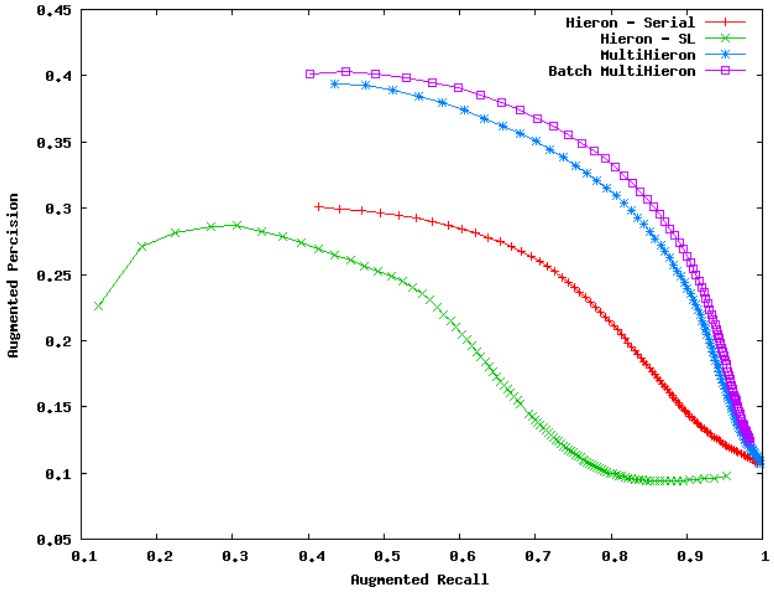
(a)

Figure 3 Precision versus recall – at metric for performance, (a) ASRS database (b) OntoNews corpus (continued) (see online version for colours)



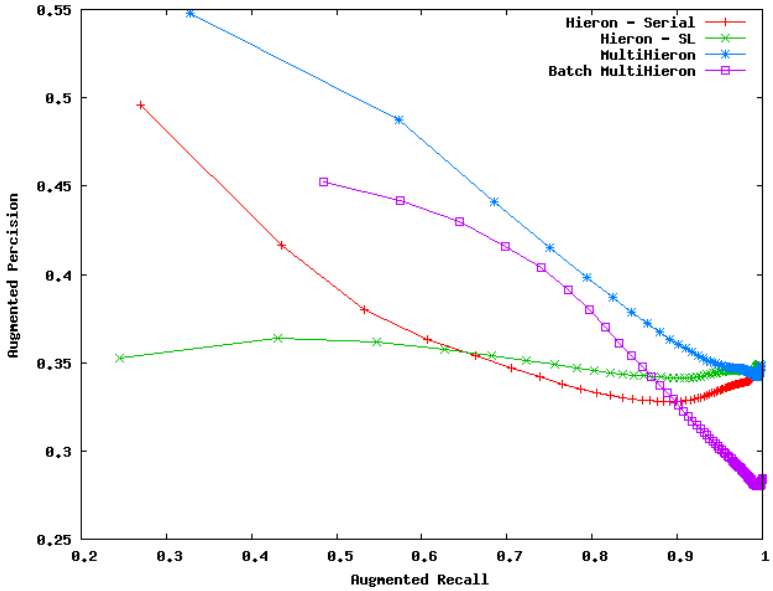
(b)

Figure 4 Augmented precision versus recall as a hierarchical measure for performance, (a) ASRS database (b) OntoNews corpus (see online version for colours)



(a)

Figure 4 Augmented precision versus recall as a hierarchical measure for performance, (a) ASRS database (b) OntoNews corpus (continued) (see online version for colours)



(b)

We now generate an augmented precision versus recall curves using hierarchical performance metric BDM. Figure 4(a) gives a dramatic less distance between the two curves. This is in part due to the flatness of the tree. This hierarchical measure gives error values in the range [0, 1] instead of the binary values given in ordinary classification metrics. At an augmented recall of 0.6 in the curve, batch MultiHieron gives augmented precision 0.38, MultiHeiron 0.36, Hieron-serial 0.3 and Hieron-SL 0.2. Batch MultiHieron is more suited to the task of multi-label hierarchical learning according on the ASRS database. In the OntoNews dataset shown in Figure 4(b), the margin is even wider. At augmented recall of 0.6, batch MultiHieron gives an augmented precision of 0.44 and MultiHieron has augmented precision of 0.48 while Hieron-serial and Hieron-SL comes in way lower with augmented precision.

MultiHieron showed a speedup over Hieron using serial training shown in Table 3. Serial training with multiple labels on the original Hieron algorithm was a multiple step process, with the majority of time spent calculating all of the possible prediction paths. With MultiHieron, only one prediction calculation need be made for each document, therefore the speedup is proportional to the average number of labels per document. The SL transformed dataset contains significantly fewer instances than the original dataset (about 20% of ASRS was SL), therefore its training time is drastically reduced. Testing uses the exact same algorithms and data structures because the underlying prediction model for the algorithm was not changed, therefore testing speed is the same. Our base for training time will be MultiHieron, 71.8s, as we do one single weight update per document. Hieron on the SL documents only uses 20% of our dataset for training, which corresponds to an approximate 20% speedup to 13.0 seconds. Hieron training serially with all documents takes significantly longer – 174.1 seconds. This difference

corresponds to the average 2.7 labels per document, seen in Table 1. Batch MultiHieron takes three seconds longer because of the cost of calculating maximum pairwise cost.

Table 3 Timing of Hieron (SL and Serial), MultiHieron and batch MultiHieron

	Hieron-SL	*Hieron-serial*	*MultiHieron*	*Batch MultiHieron*
ASRS training	13.0s	174.1s	71.8s	74.1s
ASRS testing	54.5s	53.6s	53.6s	53.7s
OntoNews training	19.1s	287.3s	105.1s	109.4s
OntoNews testing	91.2s	91.7s	92.3s	94.2s

6 Conclusions and future work

We have presented a more generalised large margin, multi-class, hierarchical perceptron algorithm. This new algorithm preserves all of the properties of original Hieron algorithm and uses the same principles to perform multi-label learning.

We used ASRS and OntoNews datasets for experimentation. The ASRS flight anomaly database posed two unique problems. First, the anomaly reports could have more than one label. Second, labels belong to a structured hierarchy. We addressed these problems with the MultiHieron algorithm. MultiHieron showed improvement in both performance and accuracy over Hieron in multi-label learning over two datasets.

In the future, we can further this research with more appropriate feature extraction techniques. Little work has been done in hierarchical importance of features. Semantic-based feature reduction can greatly improve performance of classifiers.

Acknowledgements

This research was funded (or funded in part) by NASA grant, NNX08AC35A. We would also like to thank Mohammed Salim Ahmed and the rest of the University of Texas at Dallas Database Laboratory for the constant, cheerful help they provide.

References

Barrientos, F., Castle, J., McIntosh, D. and Srivastava, A. (2007) *Preliminary Evaluation of an Aviation Safety Thesaurus Utility for Enhancing Automated Processing of Incident Reports*.

Censor, Y.A. and Zenios, S.A. (1997) *Parallel Optimization: Theory, Algorithms and Applications*, Oxford University Press, March.

Cesa-Bianchi, N., Conconi, A. and Gentile, C. (2004) 'On the generalization ability of on-line learning algorithms', *IEEE Transactions on Information Theory*.

Crammer, K., Dekel, O., Shalev-Shwartz, S. and Singer, Y. (2003) 'Online passive aggressive algorithms', *Advances in Neural Information Processing Systems 16*.

Dekel, O., Keshet, J. and Singer, Y. (2004) 'Large margin hierarchical classification', in *ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning*, New York, NY, USA, ACM. ISBN 1-58113-828-5, p.27, available at doi: http://doi.acm.org/10.1145/1015330.1015374.

Dumais, S. and Chen, H. (2000) 'Hierarchical classification of web content', in *SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, ACM, ISBN 1-58113-226-3, pp.256–263, available at doi: http://doi.acm.org/10.1145/345508.345593.

Freitag, D.B. (1999) 'Machine learning for information extraction in informal domains', PhD thesis, Pittsburgh, PA, USA, Chair-Tom Mitchell.

Herbster, M. (2001) 'Learning additive models online with fast evaluating kernels', in *COLT '01/EuroCOLT '01: Proceedings of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, Springer-Verlag, London, UK, ISBN 3-540-42343-5, pp.444–460.

Koller, D. and Sahami, M. (1997) 'Hierarchically classifying documents using very few words', in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1-55860-486-3, pp.170–178,.

Li, Y. and Bontcheva, K. (2007) 'Hierarchical, perceptron-like learning for ontology-based information extraction', in *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, New York, NY, USA, 2007, ACM, ISBN 978-1-59593-654-7, pp.777–786, available at doi: http://doi.acm.org/10.1145/1242572.1242677.

Li, Y., Bontcheva, K. and Cunningham, H. (2005) 'Using uneven margins svm and perceptron for information extraction', in *Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*.

Maynard, D., Peters, W. and Li, Y. (2006) 'Metrics for evaluation of ontology-based information extraction', *WWW 2006 Workshop on 'Evaluation of Ontologies for the Web' (EON)*.

McCallum, A., Rosenfeld, R., Mitchell, T.M. and Ng, A.Y. (1998) 'Improving text classification by shrinkage in a hierarchy of classes', in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1-55860-556-8, pp.359–367.

Tsoumakas, G. and Katakis, I. (2007) 'Multi-label classification: an overview', *International Journal of Data Warehousing and Mining*, Vol. 3, No. 3, pp.1–13.

Vapnik, V.N. (1998) *Statistical Learning Theory*, Wiley, New York, NY, USA.

Weigend, A.S., Wiener, E.D. and Pedersen, J.O. (1999) 'Exploiting hierarchy in text categorization', *Inf. Retr.*, Vol. 1, No. 3, pp.193–216, ISSN 1386-4564, available at doi: http://dx.doi.org/10.1023/A:1009983522080.

Woolam, C. and Khan, L. (2008) 'Multi-concept document classification using a perceptron-like algorithm', in *WI '08: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society.