Global Model for Hierarchical Multi-Label Text Classification

Yugo Murawaki
Graduate School of Informatics
Kyoto University
murawaki@i.kyoto-u.ac.jp

Abstract

The main challenge in hierarchical multi-label text classification is how to leverage hierarchically organized labels. In this paper, we propose to exploit dependencies among multiple labels to be output, which has been left unused in previous studies. To do this, we first formalize this task as a structured prediction problem and propose (1) a global model that jointly outputs multiple labels and (2) a decoding algorithm for it that finds an exact solution with dynamic programming. We then introduce features that capture inter-label dependencies. Experiments show that these features improve performance while reducing the model size.

1 Introduction

Hierarchical organization of a large collection of data has deep roots in human history (Berlin, 1992). The emergence of electronically-available text has enabled us to take computational approaches to real-world hierarchical text classification tasks. Such text collections include patents,[1] medical taxonomies[2] and Web directories such as Yahoo! and the Open Directory Project.[3] In this paper, we focus on *multi-label* classification, in which a document may be given more than one label.

Hierarchical multi-label text classification is a challenging task because it typically involves thousands of labels and an exponential number of output candidates. For efficiency, divide-and-conquer strategies have often been adopted. Typically, the label hierarchy is mapped to a set of local classifiers, which are invoked in a top-down fashion (Montejo-Ráez and Ureña-López, 2006; Wang et al., 2011; Sasaki and Weissenbacher, 2012). However, local search is difficult to harness because a chain of local decisions often leads to what is usually called error propagation (Bennett and Nguyen, 2009). To alleviate this problem, previous work has resorted to what we collectively call post-training adjustment.

One characteristic of the task that has not been explored in previous studies is that multiple labels to be output have dependencies among them. It is difficult even for human annotators to decide how many labels they choose. We conjecture that they consult the label hierarchy when adjusting the number of output labels. For example, if two label candidates are positioned proximally in the hierarchy, human annotators may drop one of them because they provide overlapping information.

In this paper, we propose to exploit inter-label dependencies. To do this, we first formulate hierarchical multi-label text classification as a structured prediction problem. We propose a global model that jointly predicts a set of labels. Under this framework, we replace local search with dynamic programming to find an exact solution. This allows us to extend the model with features for inter-label dependencies. Instead of locally training a set of classifiers, we also propose global training to find globally optimal parameters. Experiments show that these features improve performance while reducing the model size.

2 Task Definition

In hierarchical multi-label text classification, our goal is to assign to a document a set of labels $\mathbf{m} \subset \mathcal{L}$ that best represents the document. The pre-defined set of labels $\mathcal{L}$ is organized as a tree as illustrated in Figure 1.[4] In our task, only the

[1] http://www.wipo.int/classifications/en/

[2] http://www.nlm.nih.gov/mesh/

[3] http://www.dmoz.org/

[4] Some studies work on directed acyclic graphs (DAGs), in which each node can have more than one parent (Labrou and

International Joint Conference on Natural Language Processing, pages 46–54,
Nagoya, Japan, 14-18 October 2013.

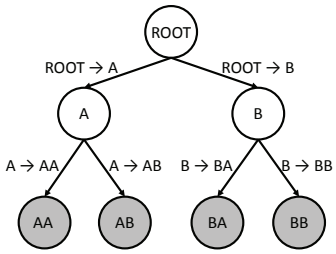


Figure 1: Example of label hierarchy. Leaf nodes, filled in gray, represent labels to be assigned to documents.

leaf nodes (AA, AB, BA and BB in this example) represent valid labels.

Let $\mathrm{leaves}(c)$ be a set of the descendants of c, inclusive of c, that are leaf nodes. For example, $\mathrm{leaves(A)} = \{\mathrm{AA}, \mathrm{AB}\}$. $p \rightarrow c$ denotes an edge from parent p to child c. Let $\mathrm{path}(c)$ be a set of edges that connect ROOT to c. For example, $\mathrm{path(AB)} = \{\mathrm{ROOT} \rightarrow \mathrm{A}, \mathrm{A} \rightarrow \mathrm{AB}\}$. Let $\mathrm{tree}(\mathbf{m}) = \bigcup_{l \in \mathbf{m}} \mathrm{path}(l)$. It corresponds to a subtree that covers $\mathbf{m}$. For example, $\mathrm{tree}(\{\mathrm{AA}, \mathrm{AB}\}) = \{\mathrm{ROOT} \rightarrow \mathrm{A}, \mathrm{A} \rightarrow \mathrm{AA}, \mathrm{A} \rightarrow \mathrm{AB}\}$.

We assume that each document x is transformed into a feature vector by $\phi(x)$. For example, we can use a bag-of-words representation of x.

We consider a supervised setting. The training data $\mathcal{T} = \{(x_i, \mathbf{m}_i)\}_{i=1}^{T}$ is used to train our models. Their performance is measured on test data.

3 Base Models

3.1 Flat Model

We begin with the flat model, one of the simplest models in multi-label text classification. It ignores the label hierarchy and relies on a set of binary classifiers, each of which decides whether label l is to be assigned to document x.

Various models have been used to implement binary classifiers, including Naïve Bayes, Logistic Regression and Support Vector Machines. We use the Perceptron family of algorithms, and it will be extended later to handle more complex structures.

The binary classifier for label l is associated with a weight vector $\mathbf{w}_l$. If $\mathbf{w}_l \cdot \phi(x) > 0$, then l is assigned to x. Note that at least one label is assigned to x. If no labels have positive scores, we choose one label with the highest score.

To optimize $\mathbf{w}_l$, we convert the original training data $\mathcal{T}$ into $\mathcal{T}_l$.

$$\mathcal{T}_l = \left\{ (x_i, y_i) \;\middle|\; \begin{array}{l} y_i = +1 \text{ if } l \in \mathbf{m}_i \\ y_i = -1 \text{ otherwise} \end{array} \right\}_{i=1}^{T}$$

Each document is treated as a positive example if it has label l; otherwise it is a negative example. Since local classifiers are independent of each other, we can trivially parallelize training.

We employ the Passive-Aggressive algorithm for training (Crammer et al., 2006). Specifically we use PA-I. The pseudo-code is given in Algorithm 1. We set the aggressiveness parameter C as 1.0.

Algorithm 1 Passive-Aggressive algorithm for training a binary classifier (PA-I).

Input: training data $\mathcal{T}_l = \{(x_i, y_i)\}_{i=1}^{T}$
Output: weight vector $\mathbf{w}_l$

$\mathbf{w}_l \leftarrow \mathbf{0}$
for $n = 1..N$ **do**
  shuffle $\mathcal{T}_l$
  for all $(x, y) \in \mathcal{T}_l$ **do**
    $l \leftarrow \max\{0, 1 - y(\mathbf{w}_l \cdot \phi(x))\}$
    if $l > 0$ **then**
      $\tau \leftarrow \min\{C, \frac{l}{\|\phi(x)\|^2}\}$
      $\mathbf{w}_l \leftarrow \mathbf{w}_l + \tau y \phi(x)$
    end if
  end for
end for

3.2 Tree Model

Unlike the flat model, the tree model exploits the label hierarchy. Each local classifier is now associated with an edge $p \rightarrow c$ of the label hierarchy and has a weight vector $\mathbf{w}_{p \rightarrow c}$. If $\mathbf{w}_{p \rightarrow c} \cdot \phi(x) > 0$, it means that x would belong to descendant(s) of c. Edge classifiers are independent of each other and can be trained in parallel.

We consider two ways of constructing training data $\mathcal{T}_{p \rightarrow c}$.

ALL — All training data are used as before.

$$\mathcal{T}_{p \rightarrow c} = \left\{ \begin{array}{l} (x_i, y_i) | \\ y_i = +1 \text{if } \exists l \in \mathbf{m}_i, l \in \mathrm{leaves}(c) \\ y_i = -1 \text{otherwise} \end{array} \right\}_{i=1}^{T}$$

Each document is treated as a positive example if it belongs to a leaf node of c, and the rest is negative examples (Punera and Ghosh, 2008).

SIB — Negative examples are restricted documents that belong to the leaves of c's siblings.

$$\mathcal{T}_{p \rightarrow c} = \left\{ \begin{array}{l} (x, y) | \\ \quad y = +1 \text{ if } \exists l \in \mathbf{m}, l \in \mathrm{leaves}(c) \\ \quad y = -1 \text{ if } \exists l \in \mathbf{m}, l \in \mathrm{leaves}(p) \\ \qquad \text{and } l \notin \mathrm{leaves}(c) \end{array} \right\}$$

Finin, 1999; LSHTC3, 2012). We leave it for future work.

Algorithm 2 Top-down local search.

Input: document x
Output: label set $\mathbf{m}$
1: $q \leftarrow [\mathrm{ROOT}]$, $\mathbf{m} \leftarrow \{\}$
2: **while** q is not empty **do**
3: $p \leftarrow$ pop out the first item of q, $\mathbf{t} \leftarrow \{\}$
4: **for all** c such that c is a child of p **do**
5: $\mathbf{t} \leftarrow \mathbf{t} \cup \{(c, \mathbf{w}_{p \to c} \cdot \phi(x))\}$
6: **end for**
7: $\mathbf{u} \leftarrow \{(c, s) \in \mathbf{t} | s > 0\}$
8: **if** $\mathbf{u}$ is empty **then**
9: $\mathbf{u} \leftarrow \{(c, s)\}$ such that c has the highest score s among p's children
10: **end if**
11: **for all** $(c, s) \in \mathbf{u}$ **do**
12: **if** c is a leaf node **then**
13: $\mathbf{m} \leftarrow \mathbf{m} \cup \{c\}$
14: **else**
15: append c to q
16: **end if**
17: **end for**
18: **end while**

This leads to a compact model because low-level edges, which are overwhelming in number, have much smaller training data than high-level edges. This is a preferred choice in previous studies (Liu et al., 2005; Wang et al., 2011; Sasaki and Weissenbacher, 2012).

3.3 Top-down Local Search

In previous studies, the tree model is usually accompanied with top-down local search for decoding (Montejo-Ráez and Ureña-López, 2006; Wang et al., 2011; Sasaki and Weissenbacher, 2012).[5] Algorithm 2 is a basic form of top-down local search. At each node, we select children to which edge classifiers return positive scores (Lines 4–7). However, if no children have positive scores, we select one child with the highest score (Lines 8–10). We repeat this until we reach leaves. The decoding of the flat model can be seen as a special case of this search.

Top-down local search is greedy, hierarchical pruning. If a higher-level classifier drops a child node, we no longer consider its descendants as output candidates. This drastically reduces the number of local classifications in comparison with the flat model. At the same time, however, this is a source of errors. In fact, a chain of local decisions accumulates errors, which is known as error propagation (Bennett and Nguyen, 2009). If the decision by a higher-level classifier was wrong, the model has no way of recovering from the error.

[5]For other methods, Punera and Ghosh (2008) post-process local classifier outputs by isotonic tree regression.

To alleviate this problem, various modifications have been proposed, which we collectively call post-training adjustment. Sasaki and Weissenbacher (2012) combined broader candidate generation with post-hoc pruning. They first generated a larger number of candidates by setting a negative threshold (e.g., -0.2) instead of 0 in Line 7. Then they filtered out unlikely labels by setting another threshold on the sum of (sigmoid-transformed) local scores of each candidate's path. S-cut (Montejo-Ráez and Ureña-López, 2006; Wang et al., 2011) adjusts the threshold for each classifier. R-cut selects top-r candidates either globally (Liu et al., 2005; Montejo-Ráez and Ureña-López, 2006) or at each parent node (Wang et al., 2011). Wang et al. (2011) developed a meta-classifier which classified a root-to-leaf path using sigmoid-transformed local scores and some additional features. All these methods assume that the models themselves are inherently imperfect and must be supplemented by additional parameters which are tuned manually or by using development data.

4 Proposed Method

4.1 Global Model

We see hierarchical multi-label text classification as a structured prediction problem. We propose a global model that jointly predicts $\mathbf{m}$, or $\mathrm{tree}(\mathbf{m})$.

$$\mathrm{score}(x, \mathbf{m}) = \mathbf{w} \cdot \Phi(x, \mathrm{tree}(\mathbf{m}))$$

$\mathbf{w}$ can be constructed simply by combining local edge classifiers.

$$\mathbf{w} = \mathbf{w}_{\mathrm{ROOT} \to \mathrm{A}} \oplus \mathbf{w}_{\mathrm{ROOT} \to \mathrm{B}}, \cdots, \oplus \mathbf{w}_{\mathrm{B} \to \mathrm{BB}}$$

Its corresponding feature function $\Phi(x, \mathrm{tree}(\mathbf{m}))$ returns copies of $\phi(x)$, each of which corresponds to an edge of the label hierarchy. Thus $\mathrm{score}(x, \mathbf{m})$ can be reformulated as follows.

$$\mathrm{score}(x, \mathbf{m}) = \sum_{p \to c \in \mathrm{tree}(\mathbf{m})} \mathbf{w}_{p \to c} \cdot \phi(x)$$

Now we want to find $\mathbf{m}$ that maximizes the global score, $\mathrm{argmax}_{\mathbf{m}}\ \mathrm{score}(x, \mathbf{m})$.

With the global model, we can confirm that local search is a major source of errors. In preliminary experiments, we trained local edge classifiers on **ALL** data and combined the resultant classifiers to create a global model. For 33% of documents in the same dataset, local search found sets of labels whose global scores were lower than the corresponding correct sets of labels.

Algorithm 3 MAXTREE(x, p)

Input: document x, tree node p
Output: label set $\mathbf{m}$, score s

1: $\mathbf{u} \leftarrow \{\}$
2: **for all** c in the children of p **do**
3: **if** c is a leaf **then**
4: $\mathbf{u} \leftarrow \mathbf{u} \cup \{(\{c\}, \mathbf{w}_{p \to c} \cdot \phi(x))\}$
5: **else**
6: $(\mathbf{m}', s') \leftarrow$ MAXTREE(x, c)
7: $\mathbf{u} \leftarrow \mathbf{u} \cup \{(\mathbf{m}', s' + \mathbf{w}_{p \to c} \cdot \phi(x))\}$
8: **end if**
9: **end for**
10: $\mathbf{r} \leftarrow \{(\mathbf{m}, s) \in \mathbf{u} | s > 0\}$
11: **if** $\mathbf{r}$ is empty **then**
12: $\mathbf{r} \leftarrow \{(\mathbf{m}, s)\}$ such that the item has the highest score s among $\mathbf{u}$
13: **end if**
14: $\mathbf{m} \leftarrow \bigcup_{(\mathbf{m},s) \in \mathbf{r}} \mathbf{m}$
15: $s \leftarrow \sum_{(\mathbf{m},s) \in \mathbf{r}} s$
16: **return** $(\mathbf{m}, s)$

4.2 Dynamic Programming

We show that an exact solution for the global model can be found by dynamic programming.[6] The pseudo-code is given in Algorithm 3. MAXTREE(x, p) recursively finds a subtree that maximizes the score rooted by p, and thus we invoke MAXTREE(x, ROOT). For p, each child c is associated with (1) a set of labels that maximizes the score of the subtree rooted by c and (2) its score (Lines 3–8). The score of c is the sum of c's tree score and the score of the edge $p \to c$. A leaf's tree score is zero.

To maximize p's tree score, we select all children that add positive scores to the parent (Line 10). If no children add positive scores, we select one child that gives the highest score (Lines 11–13). Again, the flat model can be seen as a special case of this algorithm. The selected children correspond to p's label set and score (Lines 14–15).

A possible extension to this algorithm is to output N-best label sets. Since our algorithm is much easier than bottom-up parsing (McDonald et al., 2005), it would not be so difficult (Collins and Koo, 2005).

Dynamic programming resolves the search problem. We no longer require post-training adjustment. It allows us to concentrate on improving the model itself.

[6] Bennett and Nguyen (2009) proposed a similar method, but neither global model nor global training was considered. In their method, the scores of lower-level classifiers were incorporated as meta-features of a higher-level classifier. All these classifiers were trained locally and required burdensome cross-validation techniques.

Algorithm 4 Modification to incorporate branching features. Replace Lines 10–15 of Algorithm 3.

10: $r \leftarrow \mathbf{u}$ sorted by s in descending order
11: $\mathbf{r}' \leftarrow \{\}, \quad s' \leftarrow 0, \quad \mathbf{m}' \leftarrow \{\}$
12: **for** $k = 1..$size of r **do**
13: $(\mathbf{m}, s) \leftarrow r[k]$
14: $s' \leftarrow s' + s, \quad \mathbf{m}' \leftarrow \mathbf{m}' \cup \mathbf{m}$
15: $\mathbf{r}' \leftarrow \mathbf{r}' \cup \{(\mathbf{m}', s' + \mathbf{w}_{\mathrm{BF}} \cdot \phi_{\mathrm{BF}}(p, k))\}$
16: **end for**
17: $(\mathbf{m}, s) \leftarrow$ item in $\mathbf{r}'$ that has the highest s

4.3 Inter-label Dependencies

Now we are ready to exploit inter-label dependencies. We introduce branching features, a simple but powerful extension to the global model. They influence how many children a node selects. The corresponding function is $\phi_{\mathrm{BF}}(p, k)$, where p is a non-leaf node and k is the number of children to be selected for p. To avoid sparsity, we choose one of $R + 1$ features ($1, \cdots, R$ or $>R$) for some pre-defined R. To be precise, we fire two features per non-leaf node: one is node-specific and the other is shared among non-leaf nodes. As a result, we append at most $(I + 1)(R + 1)$ features to the global weight vector, where I is the number of non-leaf nodes.

All we have to do to incorporate branching features is to replace Lines 10–15 of Algorithm 3 with Algorithm 4. For given k, we first need to select k children that maximize the sum of the scores. This can by done by sorting children by score and select the first k children. We then add a score of branching features $\mathbf{w}_{\mathrm{BF}} \cdot \phi_{\mathrm{BF}}(p, k)$ (Line 15). Finally we chose a candidate with the highest score (Line 17).

4.4 Global Training

Up to this point, the global model is constructed by combining locally trained classifiers. Of course, we can directly train the global model. In fact we cannot incorporate branching features without global training.

Algorithm 5 shows a Passive-Aggressive algorithm for the structured output (Crammer et al., 2006). We can find an exact solution under the current weight vector by dynamic programming (Line 5).[7] The cost ρ reflects the degree to which the model's prediction was wrong. It is based on the

[7] If we want for some reason to stick to local search, we need to address the problem of "non-violation." With inexact search, the model prediction $\hat{\mathbf{m}}$ may have a lower score than correct $\mathbf{m}$, making the update invalid. Several methods have been proposed to solve this problem (Collins and Roark, 2004; Huang et al., 2012).

Algorithm 5 Passive-Aggressive algorithm for global training (PA-I, prediction-based updates).

Input: training data $\mathcal{T} = \{(x_i, \mathbf{m}_i)\}_{i=1}^{T}$
Output: weight vector $\mathbf{w}$

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1..N$ **do**
  shuffle $\mathcal{T}$
  for all $(x, \mathbf{m}) \in \mathcal{T}$ **do**
    predict $\hat{\mathbf{m}} \leftarrow \mathrm{argmax}_{\mathbf{m}}\ \mathrm{score}(x, \mathbf{m})$
    $\rho \leftarrow 1 - 2|\mathbf{m} \cap \hat{\mathbf{m}}|/(|\mathbf{m}| + |\hat{\mathbf{m}}|)$
    if $\rho > 0$ **then**
      $l \leftarrow \mathrm{score}(x, \hat{\mathbf{m}}) - \mathrm{score}(x, \mathbf{m}) + \sqrt{\rho}$
      $\tau \leftarrow \min\{C, \frac{l}{\|\Phi(x, \mathrm{tree}(\mathbf{m})) - \Phi(x, \mathrm{tree}(\hat{\mathbf{m}}))\|^2}\}$
      $\mathbf{w} \leftarrow \mathbf{w} + \tau(\Phi(x, \mathrm{tree}(\mathbf{m})) - \Phi(x, \mathrm{tree}(\hat{\mathbf{m}})))$
    end if
  end for
end for

example-based F measure, which will be reviewed in Section 5.3.

Note that what are called "global" in some previous studies are in fact *path*-based methods (Qiu et al., 2009; Qiu et al., 2011; Wang et al., 2011; Sasaki and Weissenbacher, 2012). In contrast, we present *tree-wide* optimization.

4.5 Parallelization of Global Training

One problem with global training is speed. We can no longer train local classifiers in parallel because global training makes the model monolithic. Even worse, label set prediction is orders of magnitude slower than a binary classification. For these reasons, global training is extremely slow.

We resort to iterative parameter mixing (McDonald et al., 2010). The basic idea is to split training data into small "shards" instead of subdividing the model. Algorithm 6 gives a pseudocode, where S is the number of shards. We perform training on each shard in parallel. At the end of each iteration, we average the models and use the resultant model as the initial value for the next iteration.

Iterative parameter mixing was originally proposed for Perceptron training. However, as McDonald et al. (2010) noted, it is possible to provide theoretical guarantees for distributed online Passive-Aggressive learning.

5 Experiments

5.1 Dataset

We used JSTPlus, a bibliographic database on science, technology and medicine built by Japan Science and Technology Agency (JST).[8] Each document consisted of a title, an abstract, a list of authors, a journal name, a set of categories and many other fields. For experiments, we selected a set of documents that (1) were dated 2010 and (2) contained both Japanese title and abstract. As a result, we obtained 455,311 documents, which were split into 409,892 documents for training and 45,419 documents for evaluation.

Algorithm 6 Iterative parameter mixing for global training.

Input: training data $\mathcal{T} = \{(x_i, \mathbf{m}_i)\}_{i=1}^{T}$
Output: weight vector $\mathbf{w}$

split $\mathcal{T}$ into $\mathcal{S}_1, \cdots \mathcal{S}_S$
$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1..N$ **do**
  for $s = 1..S$ **do**
    $\mathbf{w}_s \leftarrow$ asynchronously call Algorithm 5 with some modifications: $\mathcal{T}$ is replaced with $\mathcal{S}_s$, $\mathbf{w}$ is initialized with $\mathbf{w}$ instead of $\mathbf{0}$, and N is set as 1.
  end for
  join
  $\mathbf{w} \leftarrow \frac{1}{S}\sum_{s=1}^{S} \mathbf{w}_s$
end for

The number of labels was 3,209, which amounts to 4,030 edges. All the leave nodes are located at the fifth level (the root not counted). Some edges skip intermediate levels (e.g., children of a second-level node are located at the fourth level). On average 1.85 categories were assigned to a document, with a variance of 0.85. The maximum number of categories per document was 9.

For the feature representation of a document $\phi(x)$, we employed two types of features.

1. Journal name (binary). One feature was fired per document.

2. Content words in the title and abstract (frequency-valued). Frequencies of the words in the title were multiplied by two.

To extract content words, we first applied the morphological analyzer JUMAN[9] to each sentence to segment it into a word sequence. From each word sequence, we selected content words using the dependency parser KNP,[10] which tagged content words at a pre-processing step. Each document contained 380 characters on average, which corresponded to 120 content words according to JUMAN and KNP.

[8] http://www.jst.go.jp/EN/menu3/01.html

[9] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

[10] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP

5.2 Models

In addition to the flat model (**FLAT**), the tree model with various configurations was compared. We performed local training of edge classifiers on **ALL** data and **SIB** data as explained in Section 3.2. We applied top-down local search (**LS**) and dynamic programming (**DP**) for decoding. We also performed global training (**GT**) with and without branching features (**BF**).

We performed 10 iterations for training local classifiers. For iterative parameter mixing described in Section 4.5, we evenly split the training data into 10 shards and ran 10 iterations. For branching features introduced in Section 4.3, we set $R = 3$.

5.3 Evaluation Measures

Various evaluation measures have been proposed to handle multiple labels. The first group of evaluation measures we adopted is document-oriented measures often referred to as *example-based* measures (Godbole and Sarawagi, 2004; Tsoumakas et al., 2010). The example-based precision (**EBP**), recall (**EBR**) and F measure (**EBF**) are defined as follows.

$$\mathrm{EBP} = \frac{1}{T}\sum_{i=1}^{T}\frac{|\mathbf{m}_i \cap \hat{\mathbf{m}}_i|}{|\hat{\mathbf{m}}_i|}$$

$$\mathrm{EBR} = \frac{1}{T}\sum_{i=1}^{T}\frac{|\mathbf{m}_i \cap \hat{\mathbf{m}}_i|}{|\mathbf{m}_i|}$$

$$\mathrm{EBF} = \frac{1}{T}\sum_{i=1}^{T}\frac{2|\mathbf{m}_i \cap \hat{\mathbf{m}}_i|}{|\hat{\mathbf{m}}_i| + |\mathbf{m}_i|}$$

where T is the number of documents in the test data, $\mathbf{m}_i$ is a set of correct labels of the i-th document and $\hat{\mathbf{m}}_i$ is a set of labels predicted by the model.

Another group of measures are called label-based (**LB**) and are based on the precision, recall and F measure of each label (Tsoumakas et al., 2010). Multiple label scores are combined by performing macro-averaging (**Ma**) or micro-averaging (**Mi**), resulting in six measures.

Lastly we used hierarchical evaluation measures to give some scores to "partially correct" labels (Kiritchenko, 2005). If we assume a tree instead of a more general directed acyclic graph, we can formulate the (micro-average) hierarchical precision (**hP**) and hierarchical recall (**hR**) as follows.

$$\mathrm{hP} = \frac{\sum_{i=1}^{T}|\mathrm{tree}(\mathbf{m}_i) \cap \mathrm{tree}(\hat{\mathbf{m}}_i)|}{\sum_{i=1}^{T}|\mathrm{tree}(\hat{\mathbf{m}}_i)|}$$

$$\mathrm{hR} = \frac{\sum_{i=1}^{T}|\mathrm{tree}(\mathbf{m}_i) \cap \mathrm{tree}(\hat{\mathbf{m}}_i)|}{\sum_{i=1}^{T}|\mathrm{tree}(\mathbf{m}_i)|}$$

The hierarchical F measure (hF) is the harmonic mean of hP and hR.

5.4 Results

Table 1 shows the performance comparison of various models. DP-GT-BF performed best in 5 measures. Compared with FLAT, DP-GT-BF drastically improved LBMiP and hP. Branching features consistently improved F measures. The tree model with local search was generally outperformed by the flat model. Compared with FLAT, DP-ALL and DP-GT, DP-GT-BF yielded statistically significant improvements with $p < 0.01$.

DP-ALL outperformed LS-ALL for all but one measures. DP-SIB performed extremely poorly while DP-ALL was competitive with DP-GT-BF. This is in sharp contrast to the pair of LS-ALL and LS-SIB, which performed similarly. Dynamic programming forced DP-SIB's local classifiers to classify what were completely new to them because they had been trained only on small portions of data. The result was highly unpredictable.

As expected, dynamic programming was much slower than local search. In fact DP-GT-BF was more than 60 times slower than local search. Somewhat surprisingly, it took only 18% more time than FLAT. This may be explained by the fact that DP-GT-BF was 16% smaller in size than FLAT.

Although DP-ALL was competitive with DP-GT and DP-GT-BF, it is notable that global training yielded much smaller models. Branching features brought further model size reduction along with almost consistent performance improvement. This result seems to support our hypothesis concerning the decision-making process of the human annotators. They do not select each label independently but consider the relative importance among competing labels.

5.5 Discussion

Table 2 shows the performance of several models on the training data. It is interesting that FLAT and DP-ALL scored much higher on the training data

model	iterations	time (min)	size	EBP	EBR	EBF
FLAT	10	266	73M	.4520	.4111	.3956
LS-ALL	10	**5**	115M	.3927	.4064	.3713
LS-SIB	10	5	**39M**	.4010	.4396	.3881
DP-ALL	10	329	115M	.4790	.4336	.4247
DP-SIB	10	298	**39M**	.0026	**.6804**	.0481
DP-GT	10	310	68M	**.5177**	.4096	.4317
DP-GT-BF	10	315	62M	.5172	.4121	**.4347**

model	LBMaP	LBMaR	LBMaF	LBMiP	LBMiR	LBMiF	hP	hR	hF
FLAT	.4260	.2549	.2578	.4155	.3727	.3930	.5343	.4746	.5027
LS-ALL	.3288	.2764	.2415	.3622	.3716	.3668	.4988	.5060	.5024
LS-SIB	.3291	.2989	.2515	.3415	.4066	.3712	.4750	.5359	.5036
DP-ALL	**.4576**	.2760	**.2799**	.4542	.3933	.4216	.6020	.5163	.5559
DP-SIB	.0267	**.5214**	.0406	.0184	**.6649**	.0358	.0031	**.8104**	.0600
DP-GT	.4301	.2708	.2659	.5085	.3655	.4253	.6458	.4843	.5535
DP-GT-BF	.4519	.2645	.2709	**.5132**	.3701	**.4300**	**.6493**	.4898	**.5584**

Table 1: Performance comparison of various models. Time is the one required to classify test data. Loading time was not counted. Size is defined as the number of elements in the weight vector whose absolute values are greater than 10^{-7}.

model	EBF	LBMiF	hF
FLAT	.9227	.9204	.9337
DP-ALL	.8977	.8951	.9114
DP-SIB	.0731	.0540	.0743
DP-GT-BF	.7126	.6942	.7508

Table 2: Performance on the training data.

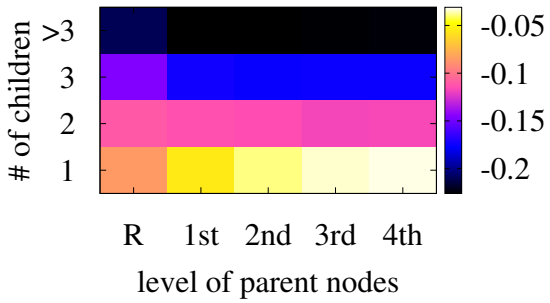


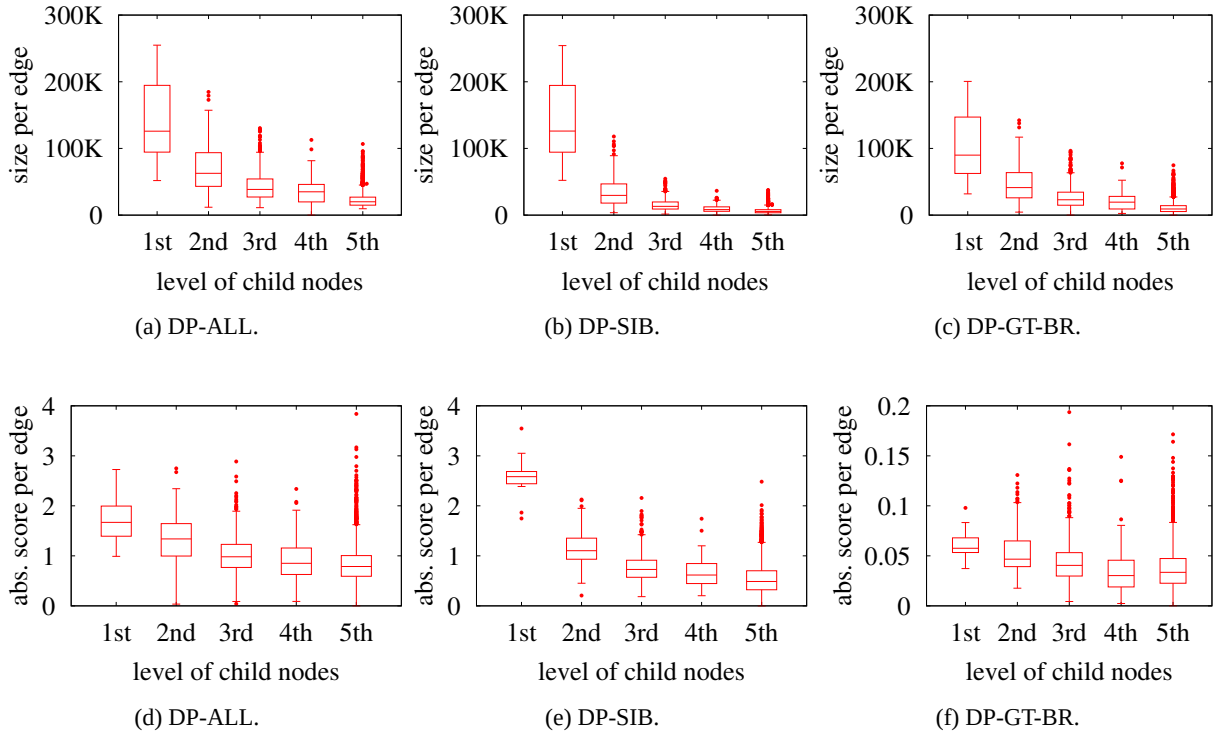
Figure 3: Heat map of the weight vector for branching features. R is the root level.

than DP-GT-BF although they were outperformed on the test data. It seems safe to conclude that local training caused overfitting.

We further investigated the models by decomposing them into edges. Figure 2 compares three models. The first three figures (a–c) report the number of non-trivial elements in each weight vector. Edges are grouped by the level of child nodes. Although DP-GT-BR was much smaller in total size than DP-ALL, the per-edge size distributions looked alike. The higher the level was, the larger number of non-trivial features each model required. Compared with DP-SIB, DP-GT-BR had compact local classifiers for the highest-level edges but the rest was generally larger. Intuitively, knowing its siblings is not enough for each local classifier, but it does not need to know all possible rivals.

The last three figures (d–f) report the averaged absolute scores of each edge that were calculated from the model output for the test data. By doing this, we would like to measure how edges of various levels affect the model output. Higher-level edges tended to have larger impact. However, we can see that in DP-GT-BR, their impact was relatively small. In other words, lower-level edges played more important roles in DP-GT-BR than in other models.

Figure 3 shows a heat map representation of the weight vector for DP-GT-BR's branching features. The value of each item is the weight value averaged over parent nodes. All averaged weight values were negative. The penalty monotonically increased with the number of children. It is not easy to compare different levels of nodes because weight values depended on other parts of the weight vector. However, the fact that lower-level nodes marked sharper contrasts between small and large number of children appears to support our



(a) DP-ALL. (b) DP-SIB. (c) DP-GT-BR.

(d) DP-ALL. (e) DP-SIB. (f) DP-GT-BR.

Figure 2: Comparison of model sizes and scores per edge. The definition of size is the same as that in Table 1.

hypothesis about the competitive nature of label candidates positioned proximally in the label hierarchy.

6 Conclusion

In this paper, we treated hierarchical multi-label text classification as a structured prediction problem. Under this framework, we proposed (1) dynamic programming that finds an exact solution, (2) global training and (3) branching features that capture inter-label dependencies. Branching features improve performance while reducing the model size. This result suggests that the selection of multiple labels by human annotators greatly depends on the relative importance among competing labels.

Exploring features that capture other types of inter-label dependencies is a good research direction. For example, "Others" labels probably behave atypically in relation to their siblings. While we focus on the setting where only the leaf nodes represent valid labels, internal nodes are sometimes used as valid labels. Such internal nodes often block the selection of their descendants. Also, we would like to work on directed acyclic graphs and to improve scalability in the future.

Acknowledgments

We thank the Department of Databases for Information and Knowledge Infrastructure, Japan Science and Technology Agency for providing JST-Plus and helping us understand the database. This work was partly supported by JST CREST.

References

Paul N. Bennett and Nam Nguyen. 2009. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 11–18.

Brent Berlin. 1992. *Ethnobiological classification: principles of categorization of plants and animals in traditional societies*. Princeton University Press.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Michael Collins and Brian Roark. 2004. Incremental parsing with the Perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online

passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 22–30. Springer Berlin Heidelberg.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured Perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151.

Svetlana Kiritchenko. 2005. *Hierarchical Text Categorization and Its Application to Bioinformatics*. Ph.D. thesis, University of Ottawa.

Yannis Labrou and Tim Finin. 1999. Yahoo! as an ontology: using Yahoo! categories to describe documents. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, pages 180–187.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations Newsletter*, 7(1):36–43, June.

LSHTC3. 2012. *ECML/PKDD-2012 Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured Perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464.

Arturo Montejo-Ráez and Luis Alfonso Ureña-López. 2006. Selection strategies for multi-label text categorization. In *Advances in Natural Language Processing*, pages 585–592. Springer.

Kunal Punera and Joydeep Ghosh. 2008. Enhanced hierarchical classification via isotonic smoothing. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 151–160.

Xipeng Qiu, Wenjun Gao, and Xuanjing Huang. 2009. Hierarchical multi-label text categorization with global margin maximization. In *Proc. of the ACL-IJCNLP Short*, pages 165–168.

Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *Proc. of ACL*, pages 598–602.

Yutaka Sasaki and Davy Weissenbacher. 2012. TTI'S system for the LSHTC3 challenge. In *ECML/PKDD-2012 Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining multi-label data. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer.

Xiao-Lin Wang, Hai Zhao, and Bao-Liang Lu. 2011. Enhance top-down method with meta-classification for very large-scale hierarchical classification. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1089–1097.