Large Scale Hierarchical Classification: Foundations, Algorithms and Applications

Huzefa Rangwala and Azad Naik

Department of Computer Science
MLBio+ Laboratory
Fairfax, Virginia, USA

SDM Tutorial, Miami, Florida

6th May, 2016

Part - I

1. Introduction and Background
   - Motivation
   - Hierarchical Classification (HC) problem description
   - Challenges
   - Methods for solving HC
2. State-of-the-Art HC Approaches
   - Parent-child regularization
   - Cost-sensitive learning
     - Package description/Software demo

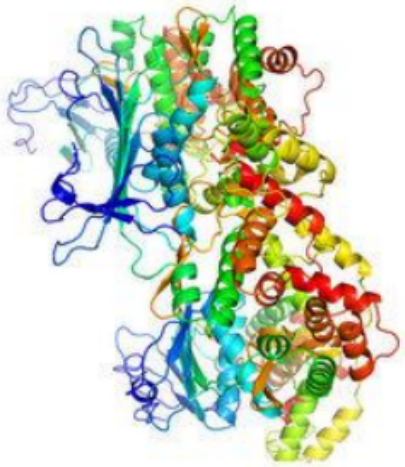
Part - II

1. Inconsistent Hierarchy
   - Motivation
   - Methods for resolving inconsistency
   - Optimal hierarchy search in hierarchical space
2. Other HC Methods
   - Learning using multiple hierarchies
   - Extreme and deep classification
3. Conclusion

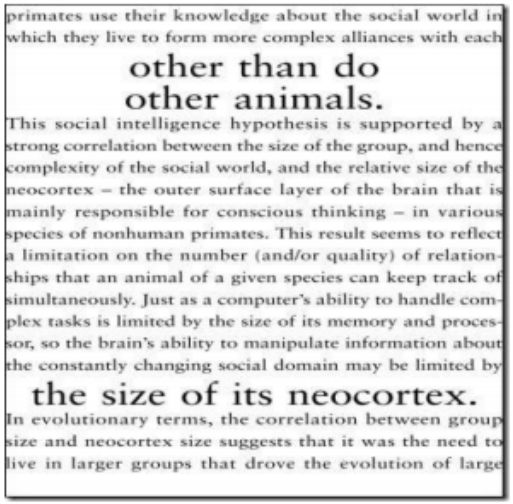
Motivation

- Exponential growth in data (image, text, video) over time
  - Big data era - megabytes & gigabytes to **terabytes** & **petabytes**
  - growth in almost all fields - astronomical, biological, web content

Astronomy



Proteins



primates use their knowledge about the social world in which they live to form more complex alliances with each

other than do other animals.

This social intelligence hypothesis is supported by a strong correlation between the size of the group, and hence complexity of the social world, and the relative size of the neocortex – the outer surface layer of the brain that is mainly responsible for conscious thinking – in various species of nonhuman primates. This result seems to reflect a limitation on the number (and/or quality) of relationships that an animal of a given species can keep track of simultaneously. Just as a computer's ability to handle complex tasks is limited by the size of its memory and processor, so the brain's ability to manipulate information about the constantly changing social domain may be limited by

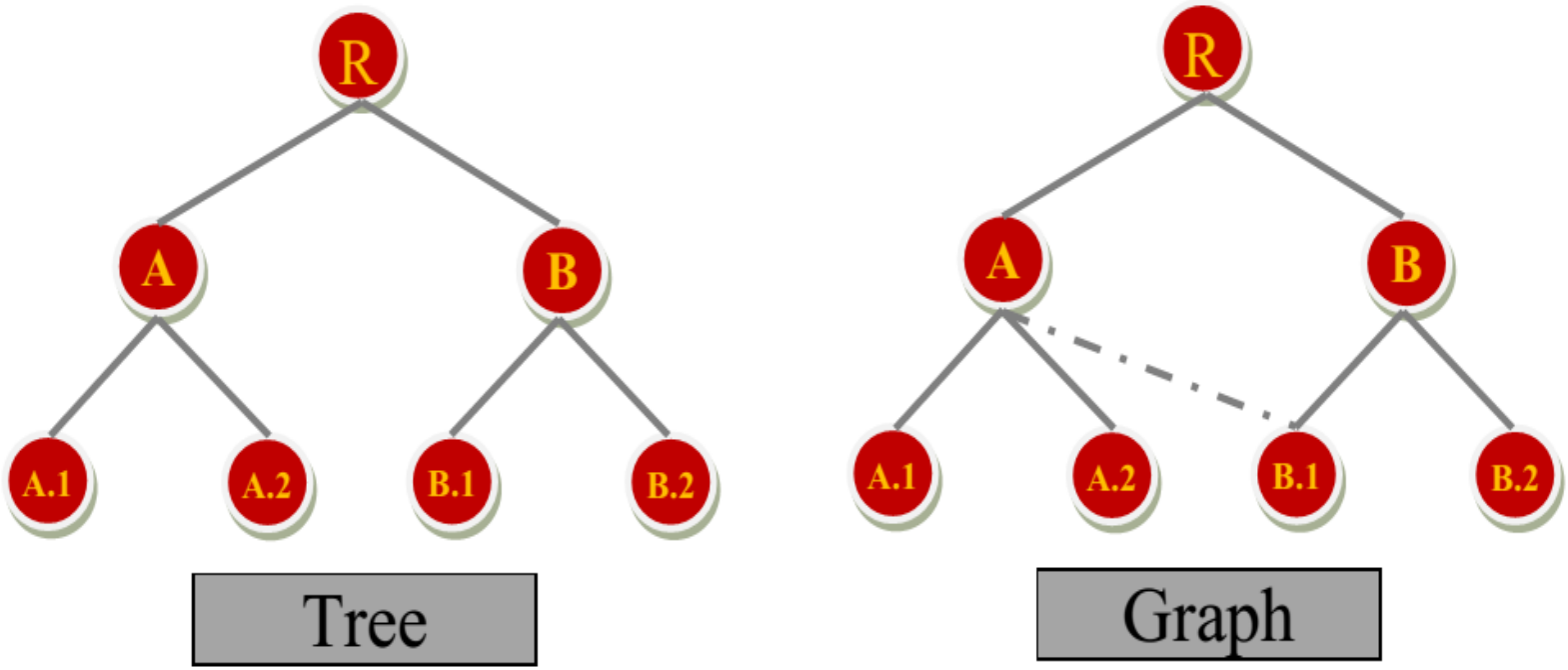
the size of its neocortex.

In evolutionary terms, the correlation between group size and neocortex size suggests that it was the need to live in larger groups that drove the evolution of large

Text

Data Organization

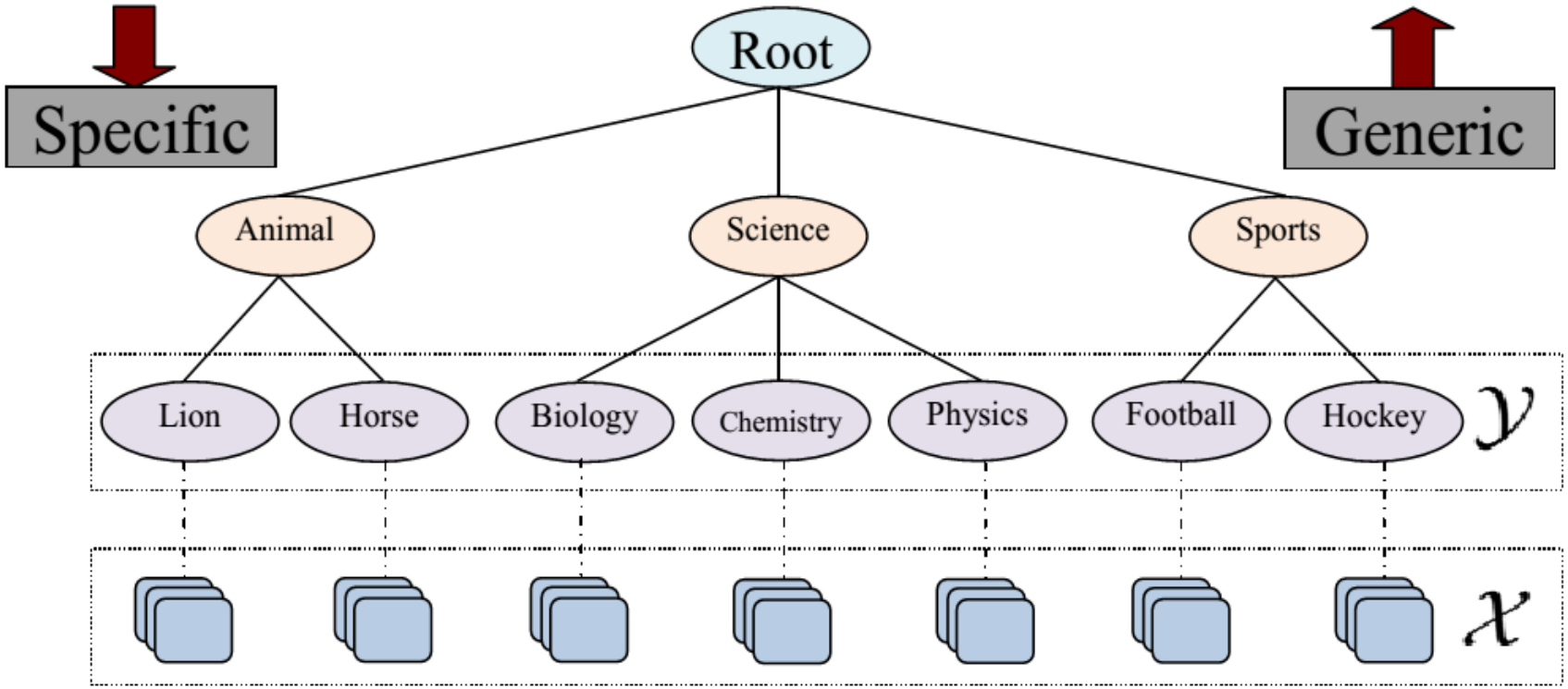
- Organize data into structure
  - tree, graph [LSHTC, BioASQ and ILSVRC challenge]



- Useful in various applications
  - query search, browsing and categorizing products

Hierarchical Structure

- Classes organized into the hierarchical structure
- **Generic (↑)** to **specific (↓)** categories in top-down order



Hierarchical Classification

Goal

Given hierarchy of classes **exploit the hierarchical structure** to learn models and classify unlabeled test examples (instances) to one or more nodes in the hierarchy

Hierarchical Classification

Goal

Given hierarchy of classes **exploit the hierarchical structure** to learn models and classify unlabeled test examples (instances) to one or more nodes in the hierarchy

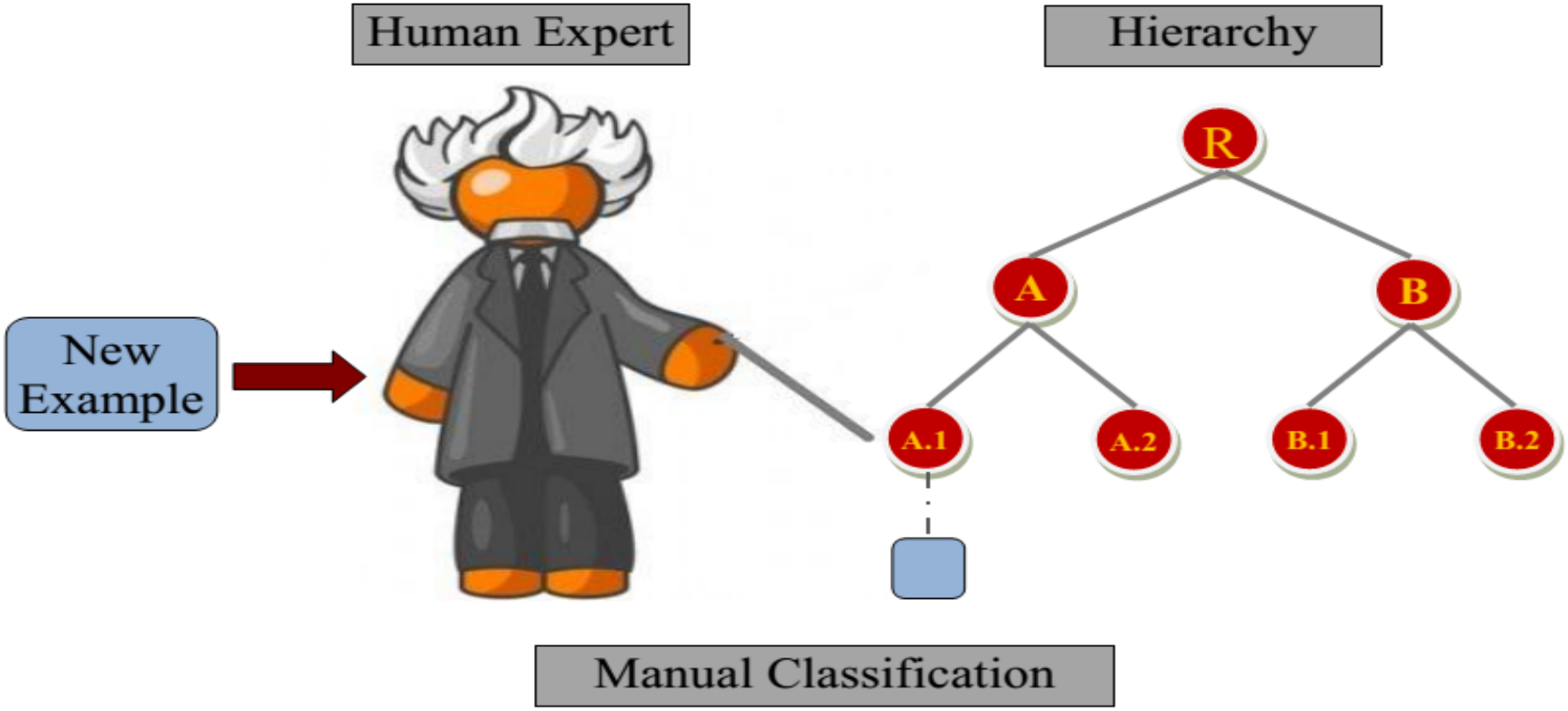
Solution

(i) Manual Classification

(ii) Automated Classification

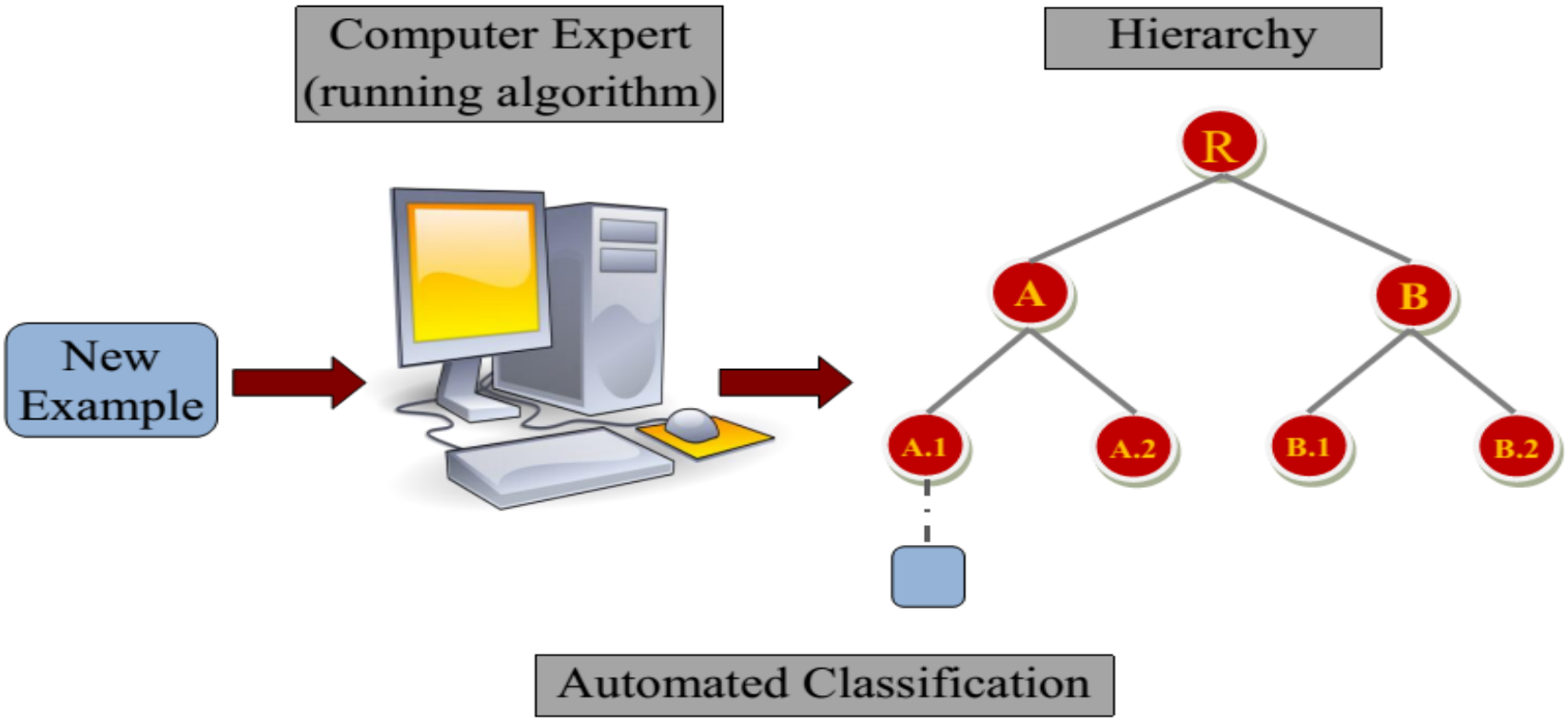
Manual Classification

- Requires human understanding and expertise
- Infeasible for huge data



Automated Classification

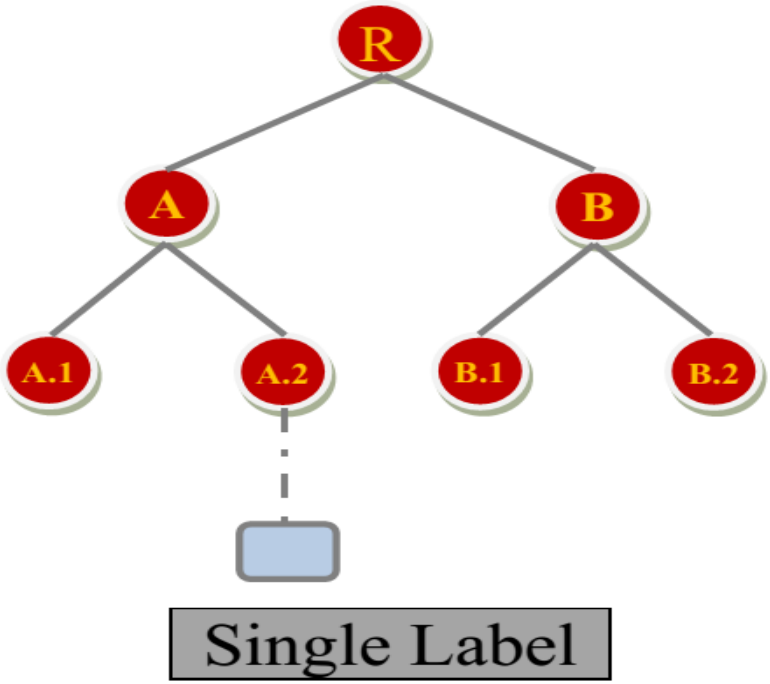
- Trained expert (such as computer)
- Scalable for huge data

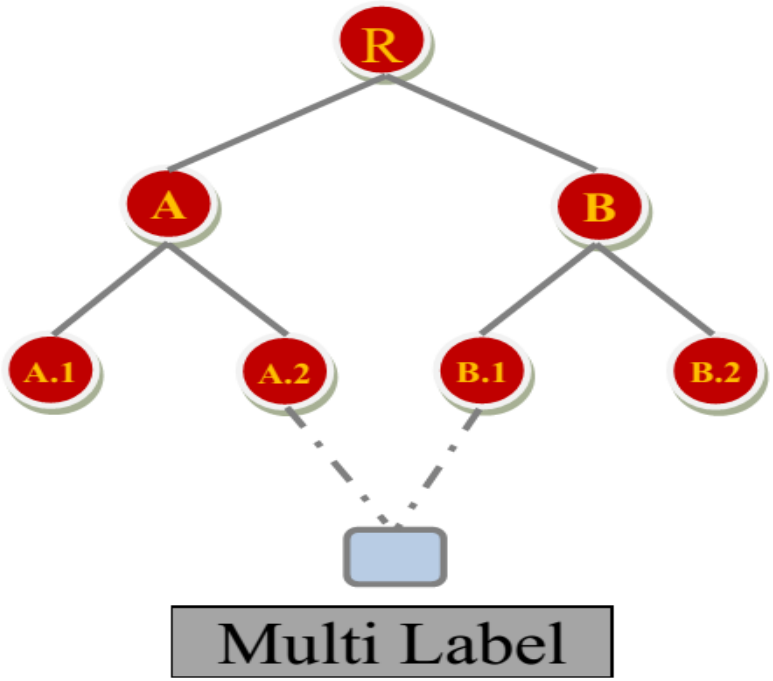


Challenges - I

Single label vs. multi-label

- **Single label classification** - each example belongs exclusively to one class only
- **Multi-label classification** - example may belong to more than one class

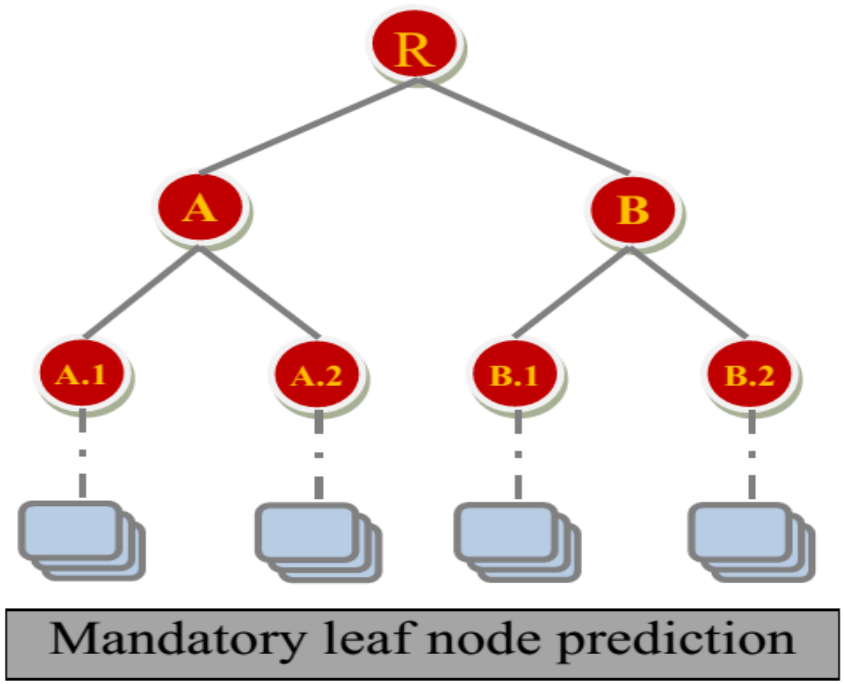


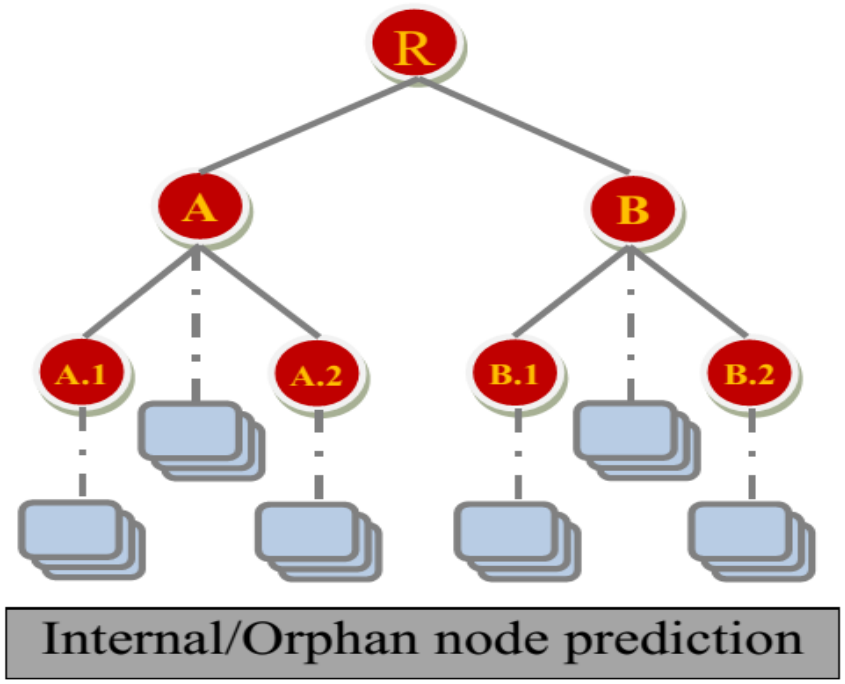


Challenges - II

Mandatory leaf node vs. internal node prediction

- Example may be assigned to internal nodes

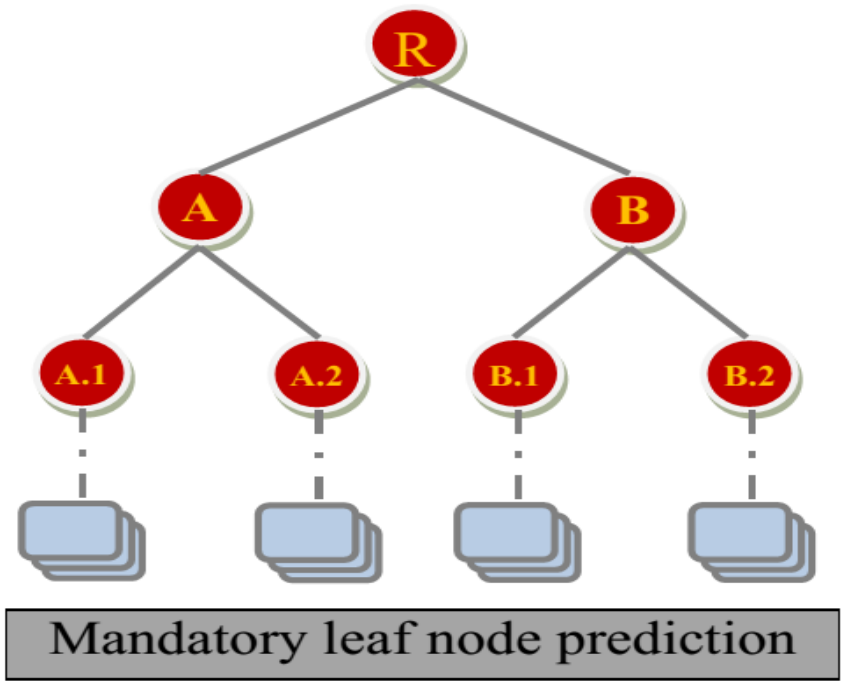


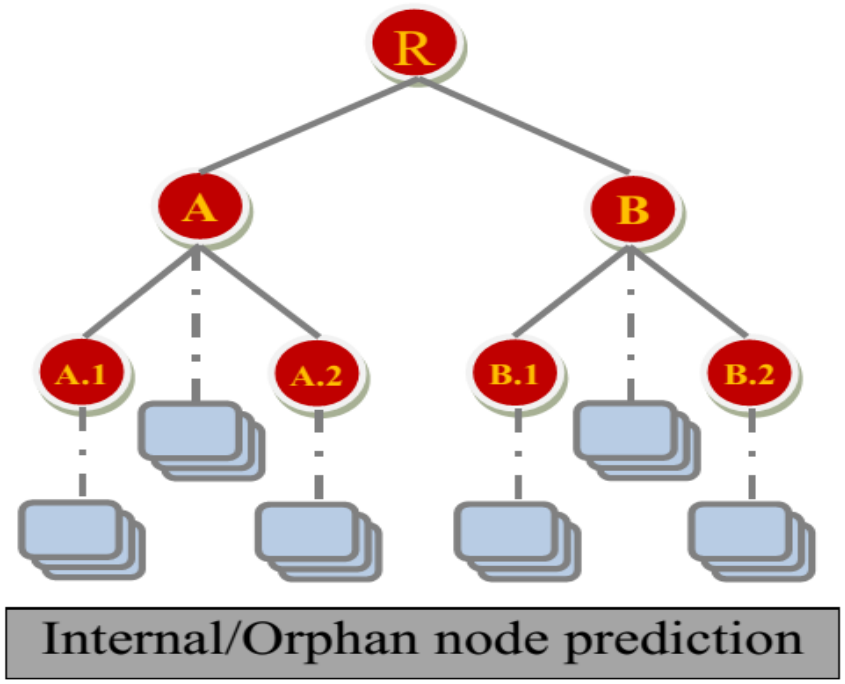


Challenges - II

Mandatory leaf node vs. internal node prediction

- Example may be assigned to internal nodes
- **Orphan node** detection problem

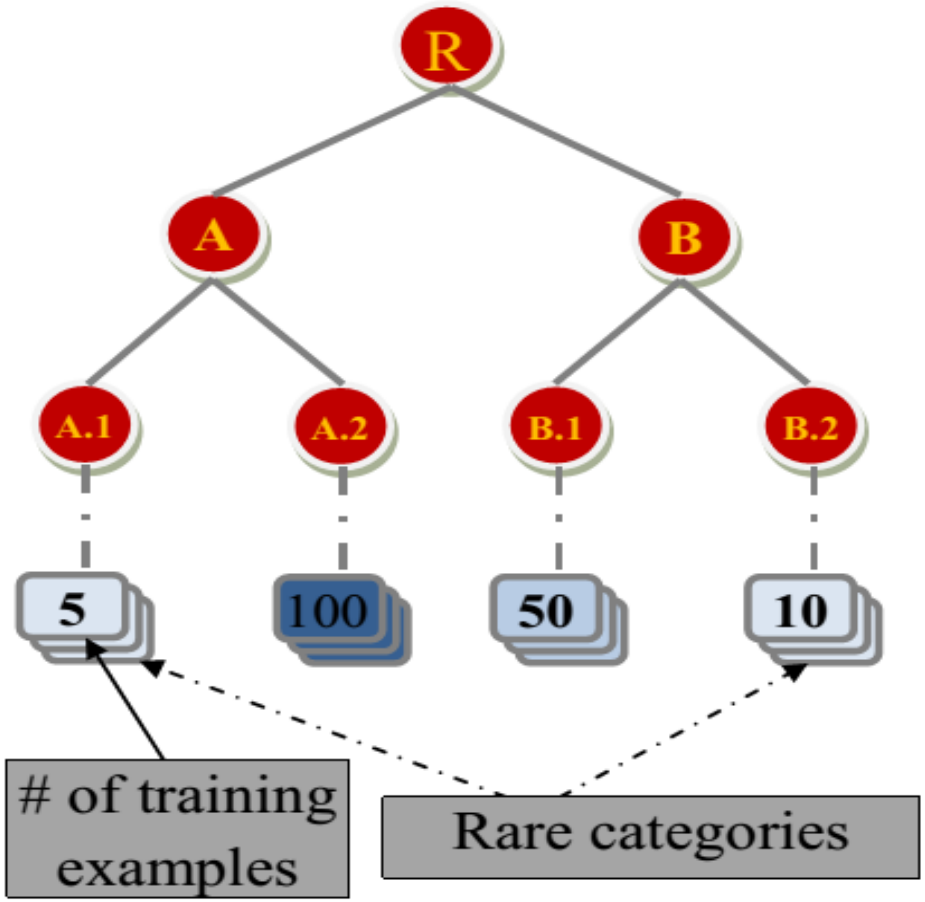




Challenges - III

Rare categories

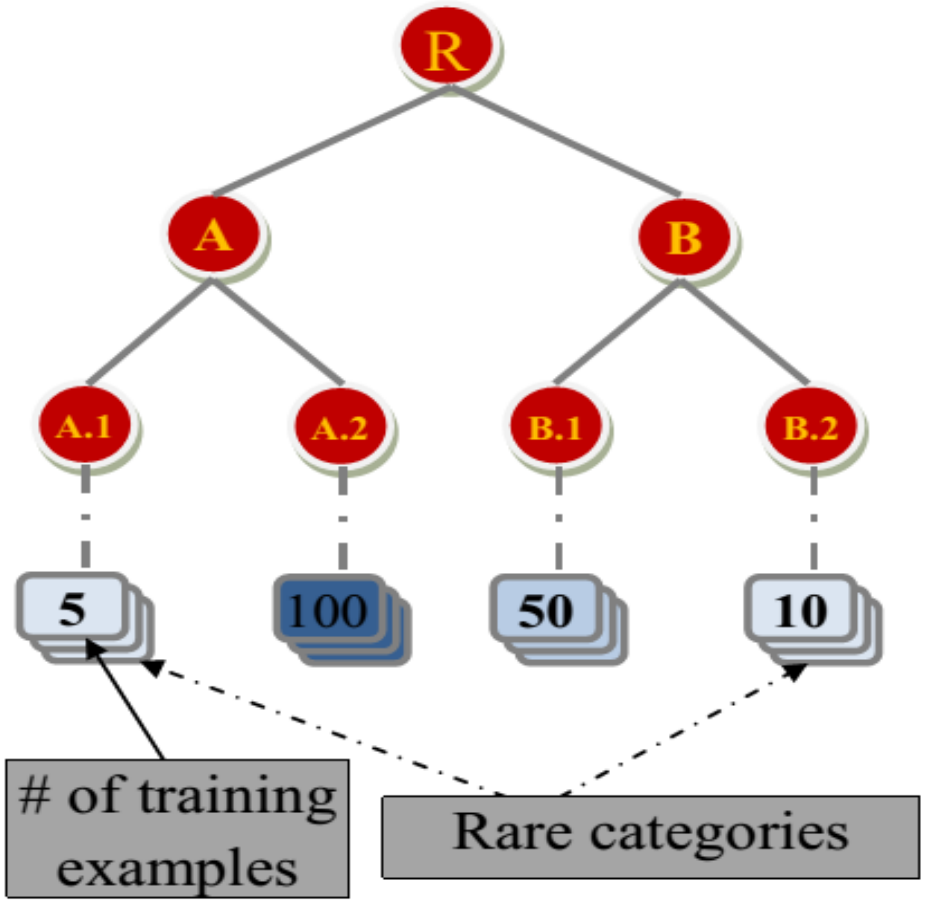
- Many classes with very few labeled examples

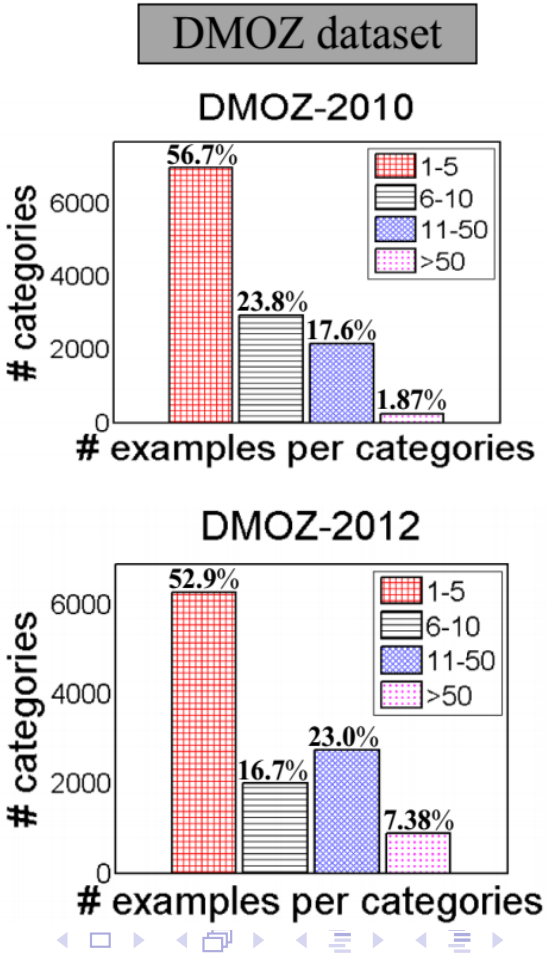


Challenges - III

Rare categories

- Many classes with very few labeled examples
- More prevalent in large scale datasets - $\geq$**70%** have $\leq$**10** examples

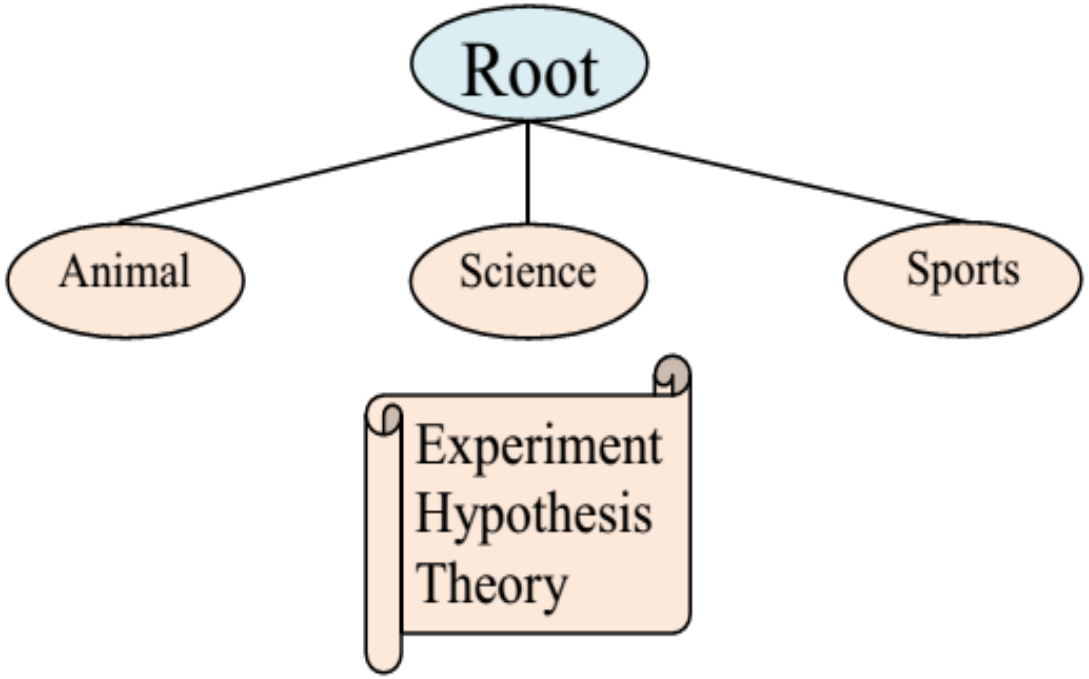




Challenges - IV

Feature selection

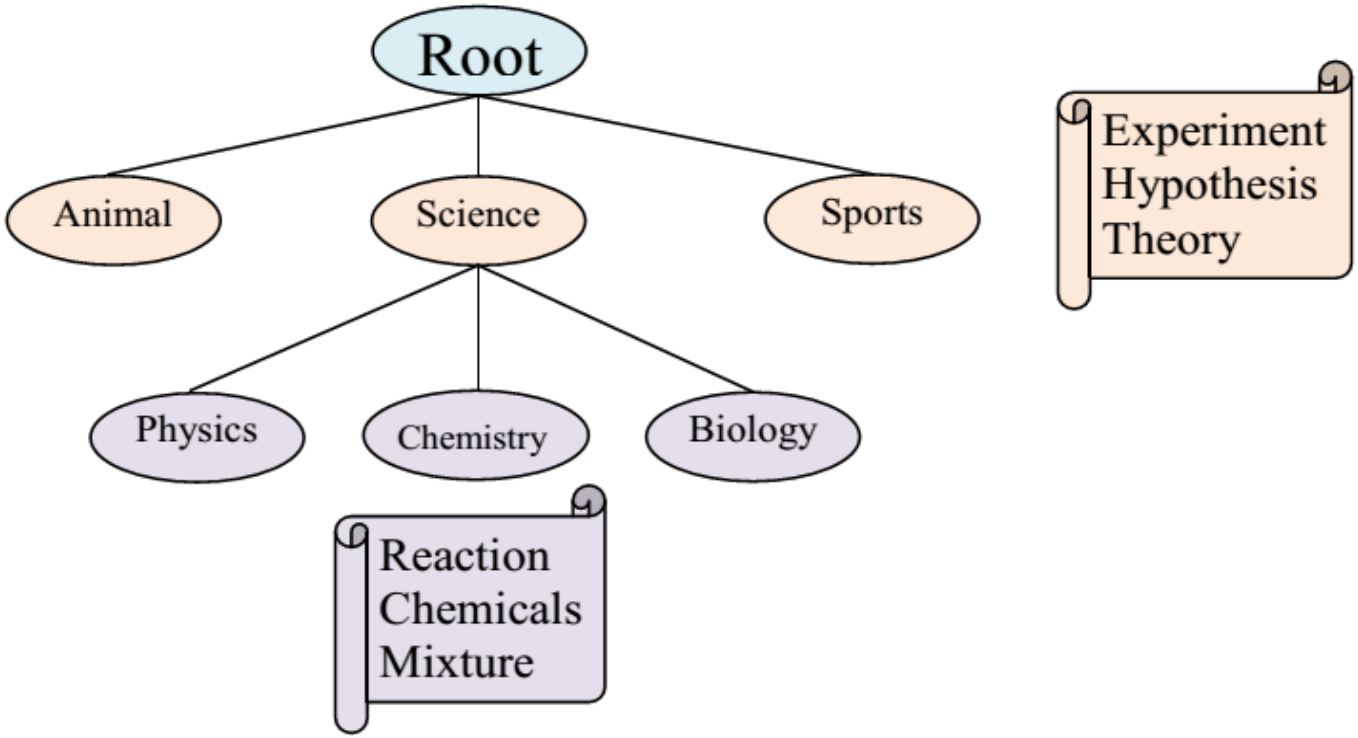
- All features are **not essential** to **discriminate** between classes
- Identify features to improve classification performance



Challenges - IV

Feature selection

- All features are **not essential** to **discriminate** between classes
- Identify features to improve classification performance



Other Challenges

- **Parameter optimization**
  - incorporate relationships (**parent-child, silings**) information

Other Challenges

- **Parameter optimization**
  - incorporate relationships (**parent-child, silings**) information
- **Scalability**
  - large # of classes, features and examples require **distributed computation**

Dataset	#Training examples	#Leaf node (classes)	#Features	#Parameters	Parameter size (approx)
DMOZ-2010	128,710	12,294	381,580	4,652,986,520	18.5 GB
DMOZ-2012	383,408	11,947	348,548	4,164,102,956	16.5 GB

Other Challenges

- **Parameter optimization**
  - incorporate relationships (**parent-child, silings**) information
- **Scalability**
  - large # of classes, features and examples require **distributed computation**

Dataset	#Training examples	#Leaf node (classes)	#Features	#Parameters	Parameter size (approx)
DMOZ-2010	128,710	12,294	381,580	4,652,986,520	18.5 GB
DMOZ-2012	383,408	11,947	348,548	4,164,102,956	16.5 GB

- **Inconsistent hierarchy**
  - not suitable for classification (more details later)

Notation

$n = \#$ of training examples (instances)	$D =$ dimension of each instance
$\mathcal{N} =$ set of nodes in the hierarchy	$L =$ set of leaf node (classes)
$\mathcal{C}(t) =$ children of node t	$\pi(t) =$ parent of node t

FEATURES

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^D \\ x_2^1 & x_2^2 & \cdots & x_2^D \\ \vdots & \vdots & \ddots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^D \end{bmatrix}$$

INSTANCES

INSTANCE MATRIX

CLASSES

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & \cdots & 1 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

INSTANCES

LABEL MATRIX

FEATURES

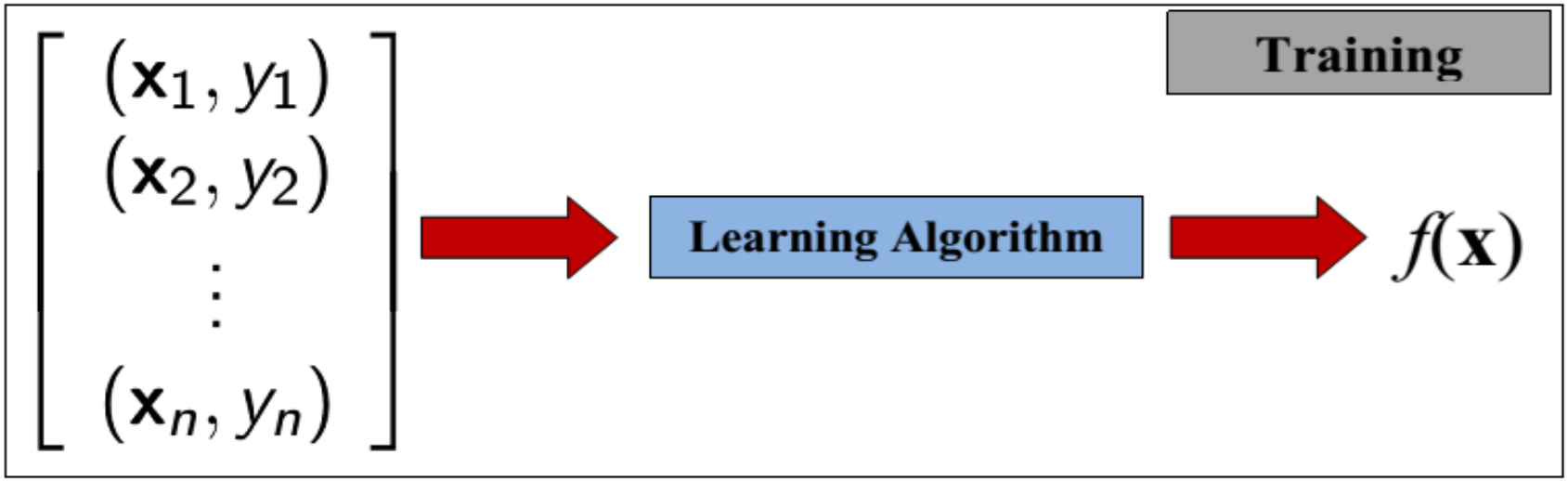
$$\mathbf{W} = \begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^D \\ w_2^1 & w_2^2 & \cdots & w_2^D \\ \vdots & \vdots & \ddots & \vdots \\ w_L^1 & w_L^2 & \cdots & w_L^D \end{bmatrix}$$

CLASSES

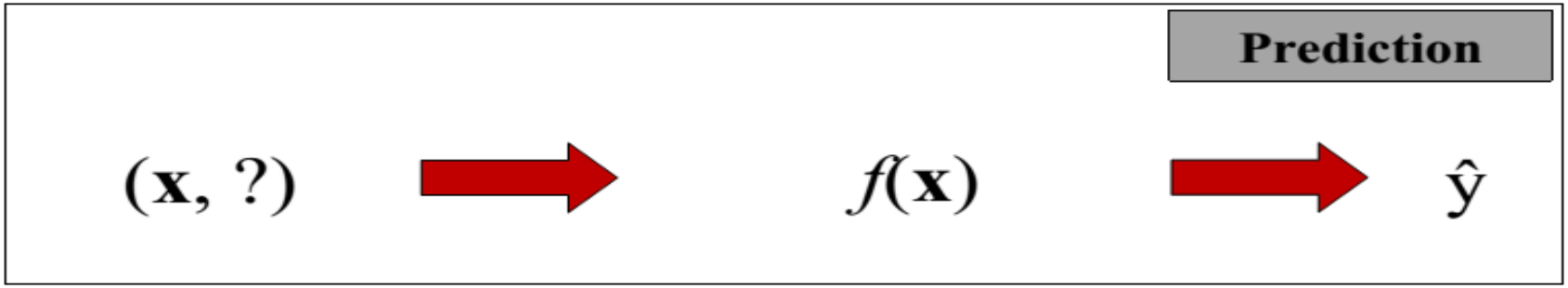
WEIGHT MATRIX

Classification

Training - Learn mapping function using training data



Testing - Predict the label of test example



Learning Algorithm: General Formulation

Combination of two terms:

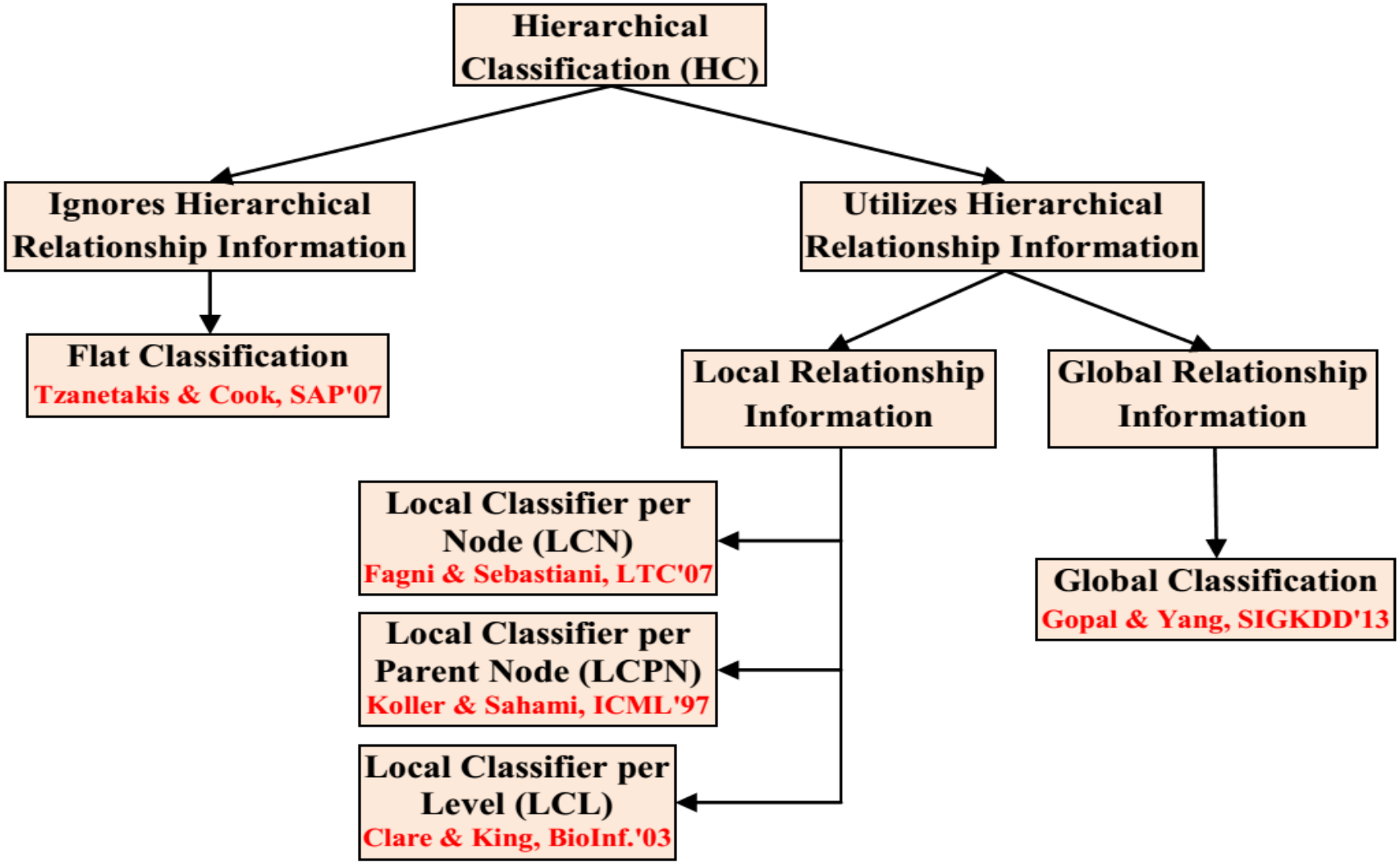
1. **Empirical loss** - controls how well the learnt models fits the training data
2. **Regularization** - prevent models from over-fitting and encodes additional information such as hierarchical relationships

$$\min_{\mathbf{W}} \mathcal{L}(f(\mathbf{X}, \mathbf{W}), \mathbf{Y}) + \lambda \Omega(\mathbf{W})$$

Empirical Loss

Regularization

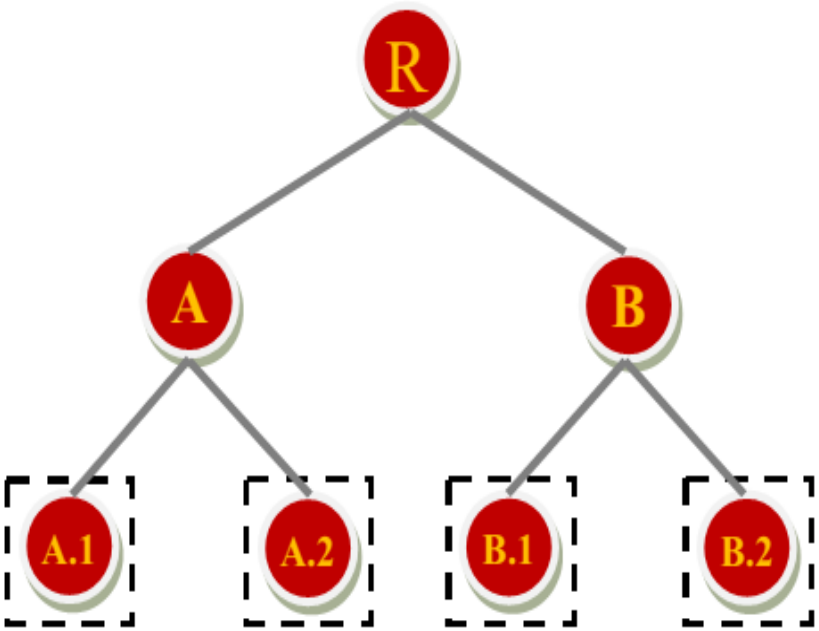
Different Approaches for Solving HC Problem



Flat Classification Approach

- Simplest method (ignores hierarchy)
- Learn discriminant classifiers for each leaf node in the hierarchy
- Unlabeled test example classified using the rule:

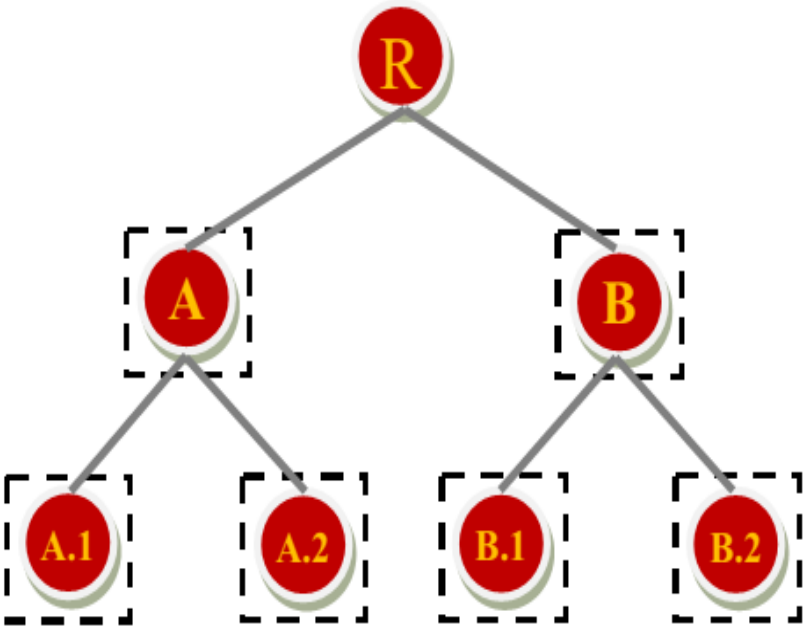
$$\boxed{\hat{y} = \underset{y \in \mathcal{Y}}{\textbf{arg max}}\ f(\mathbf{x}, y|\mathbf{w})}$$



Local Classification Approach - I

Local Classifier per Node (LCN)

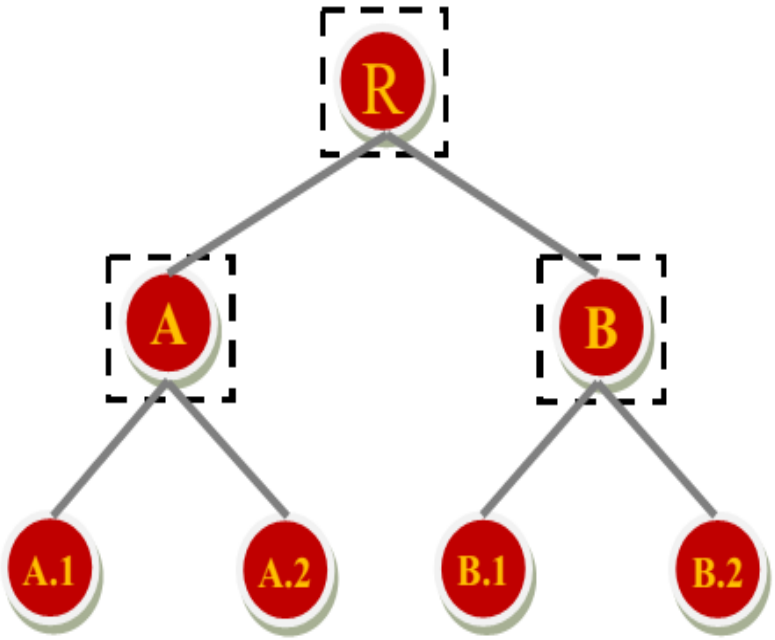
- Learn binary classifiers for all non-root nodes
- Goal is to effectively discriminate between the siblings
- Top-down approach is followed for classifying unlabeled test examples



Local Classification Approach - II

Local Classifier per Parent Node (LCPN)

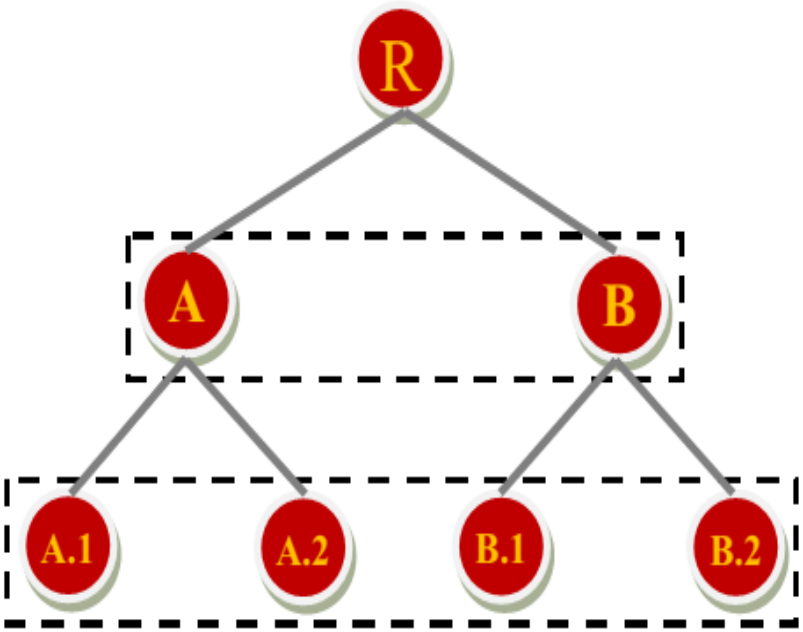
- Learn multi-class classifiers for all non-leaf nodes
- Like **LCN** goal is to effectively discriminate between the siblings
- Top-down approach is followed for classifying unlabeled test examples



Local Classification Approach - III

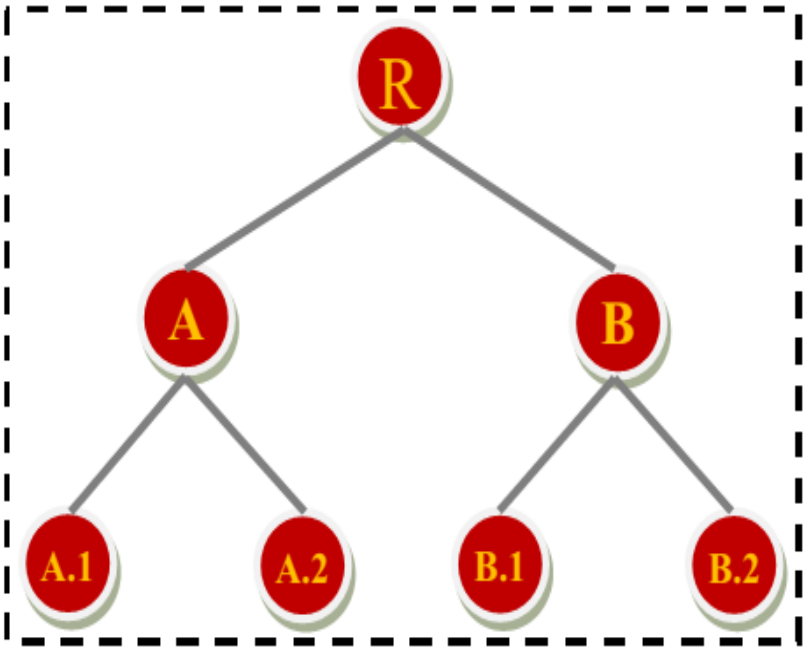
Local Classifier per Level (LCL)

- Learn multi-class classifiers for all levels in the hierarchy
- Least popular among local approaches
- Prediction inconsistency may occur and hence post-processing step is required



Global Classification Approach

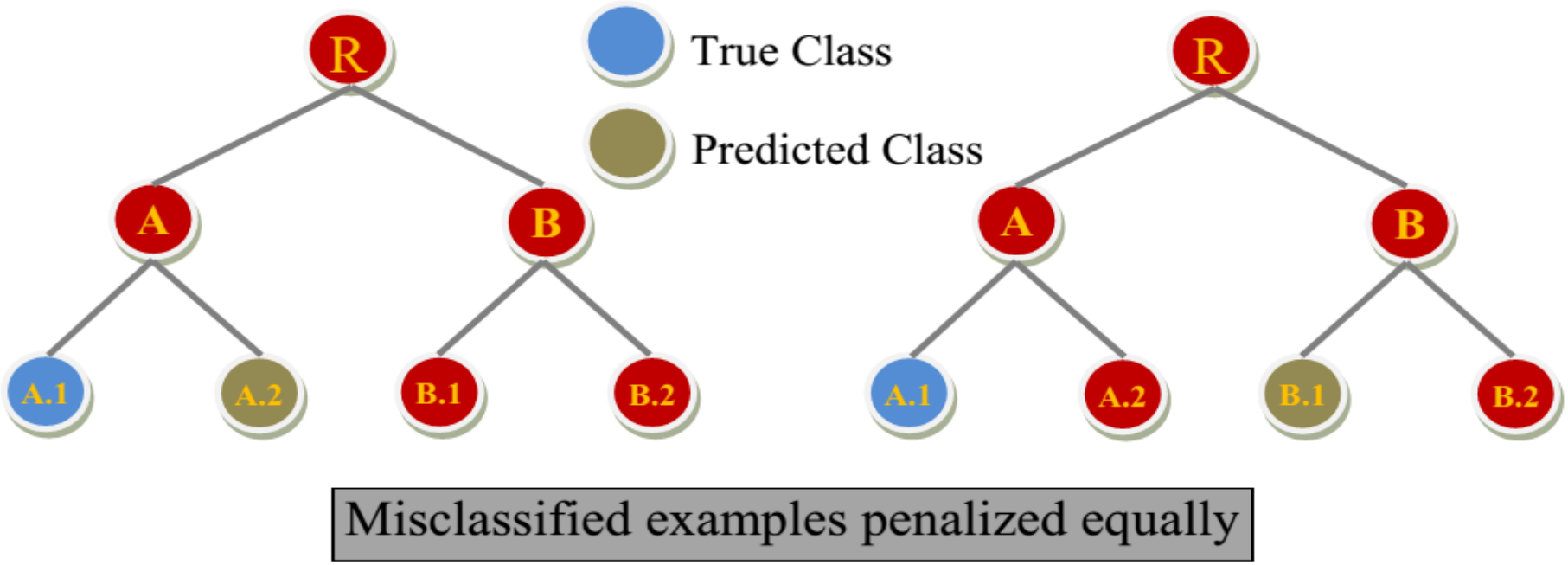
- Learn global function considering all hierarchical relationships
- Often referred as Big-Bang approach
- Unlabeled test instance is classified using an approach similar to flat or local methods



Evaluation Metrics - I

Flat evaluation measures

- Misclassifications treated equally

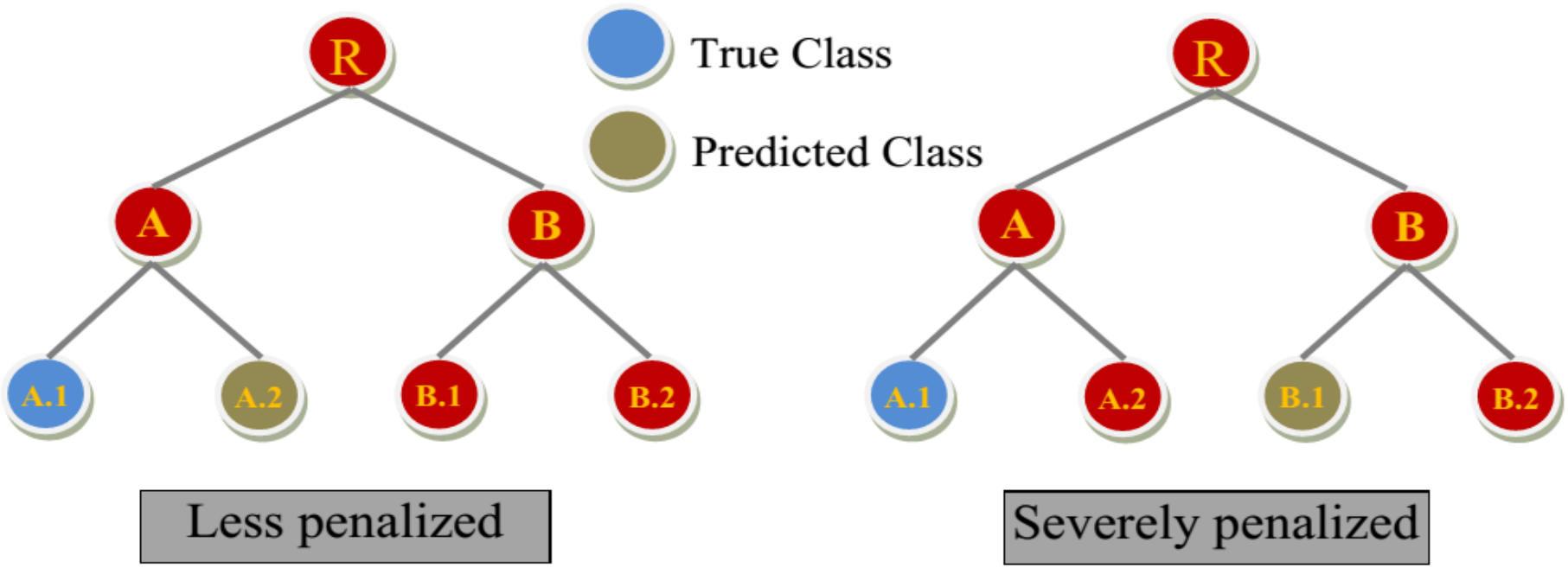


- Common evaluation metrics:
  - **Micro-F1** - gives equal weightage to all examples, dominated by common class
  - **Macro-F1** - gives equal weightage to each class

Evaluation Metrics - II

Hierarchical evaluation measures

- Hierarchical distance between the true and predicted class taken into consideration for performance evaluation



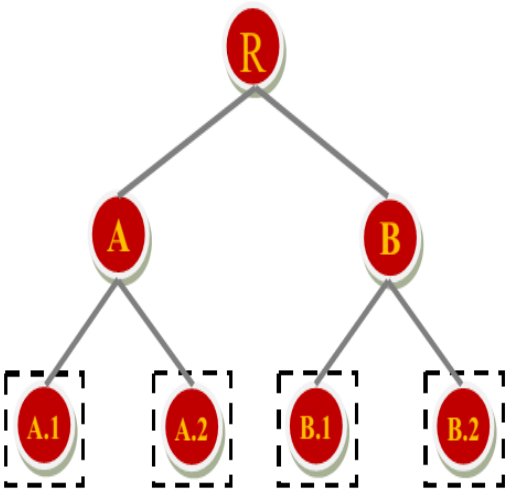
- Common evaluation metrics:
  - **Hierarchical-F1** - common ancestors between true and predicted class
  - **Tree Error** - average hierarchical distance b/w true and predicted class

Multi-Task Learning (MTL)

- Involves **joint training** of multiple related tasks to improve generalization performance
- Independent learning problems can utilize the **shared knowledge**
- Exploits **inductive biases** that are helpful to all the related tasks
  - similar set of parameters
  - common feature space
- Examples
  - personal email spam classification - many person with same spam
  - automated driving - brakes and accelerator

Parent-child Regularization, Gopal and Yang, SIGKDD'13

Motivation



- Traditional approach learn classifiers for each leaf node (task) to discriminate one class from other

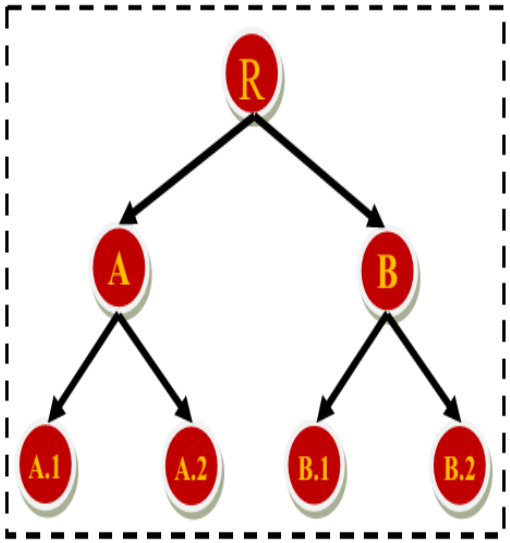
$$\min_{\mathbf{w}_t} \frac{1}{2}||\mathbf{w}_t||_2^2 + C\sum_{i=1}^{n}\left[1 - \mathbf{Y}_{it}\mathbf{w}_t^T\mathbf{x}_i\right]_+$$

- Works well if:
  - Dataset is small
  - Balanced
  - Sufficient positive examples per class to learn generalized discriminant function

Drawbacks

- Real world datasets suffers from rare categories issue
  Remember: **70% classes have less than 10 examples per class**
- Large number of classes (**scalability issue**)

Motivation - II



- Can we improve the performance of **data sparse** leaf nodes by taking advantage of **data rich** nodes at higher levels?
- Incorporate **inter-class dependencies** to improve classification
  - examples belonging to *Soccer* category is less likely to belong to *Software* category

$$\min_{\mathbf{w}_t} \frac{1}{2}||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + C \sum_{k \in \mathcal{C}(t)} \sum_{i=1}^{n} \left[1 - \mathbf{Y}_{ik} \mathbf{w}_t^T \mathbf{x}_i\right]_+$$

Objective

- How to effectively incorporate the hierarchical relationships into the objective function to improve generalization performance
- Make it scalable for larger datasets

Proposed Formulation

- Enforces model parameters (weights) to be similar to the parent in regularization
- Proposed state-of-the-art: HR-SVM and HR-LR global formulation

HR-SVM

$$\min_{\mathbf{W}} \sum_{t \in \mathcal{N}} \frac{1}{2} ||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + C \sum_{k \in L} \sum_{i=1}^{n} \left[1 - \mathbf{Y}_{ik} \mathbf{w}_k^T \mathbf{x}_i \right]_+$$

Internal Node

$$\min_{\mathbf{w}_t} \frac{1}{2} ||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + \frac{1}{2} \sum_{c \in \mathcal{C}(t)} ||\mathbf{w}_c - \mathbf{w}_t||_2^2$$

Leaf Node

$$\min_{\mathbf{w}_t} \frac{1}{2} ||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + \frac{1}{2} \sum_{i=1}^{n} \left[1 - \mathbf{Y}_{it} \mathbf{w}_t^T \mathbf{x}_i \right]_+$$

HR-LR Models

- Similar formulation as **HR-SVM**
- **Logistic loss** instead of hinge loss

HR-LR

$$\min_{\mathbf{W}} \sum_{t \in \mathcal{N}} \frac{1}{2} ||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + C \sum_{k \in L} \sum_{i=1}^{n} log(1 + exp(-\mathbf{Y}_{ik} \mathbf{w}_k^T \mathbf{x}_i))$$

Internal Node

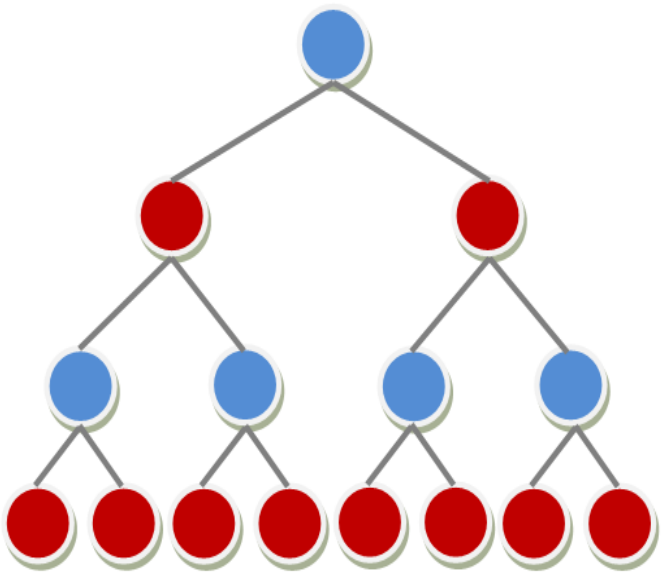
$$\min_{\mathbf{w}_t} \frac{1}{2} ||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + \frac{1}{2} \sum_{c \in \mathcal{C}(t)} ||\mathbf{w}_c - \mathbf{w}_t||_2^2$$

Leaf Node

$$\min_{\mathbf{w}_t} \frac{1}{2} ||\mathbf{w}_t - \mathbf{w}_{\pi(t)}||_2^2 + \frac{1}{2} \sum_{i=1}^{n} log(1 + exp(-\mathbf{Y}_{it} \mathbf{w}_t^T \mathbf{x}_i))$$

Proposed Parallel Implementation

- Each node is independent of all other nodes except its neighbours
- Objective function is block separable. Therefore, Parallel Block Coordinate Descent (CD) can be used for optimization



Alternate level optimized in each iteration

1. Fix odd-levels parameters, optimize even-levels in parallel
2. Fix even-levels parameters, optimize odd-levels in parallel
3. Repeat untill convergence

- Extended to graph by first finding the minimum graph coloring [Np-hard] and repeatedly optimizing nodes with the same color in paralle during each iteration

Experiments

Dataset description

- Wide range of single and multi-label dataset with varying number of features and categories were used for model evaluation

Datasets	# Features	# Categories	Type	Avg # labels (per instance)
CLEF	89	87	Single-label	1
RCV1	48,734	137	Multi-label	3.18
IPC	541,869	552	Single-label	1
DMOZ-SMALL	51,033	1,563	Single-label	1
DMOZ-2010	381,580	15,358	Single-label	1
DMOZ-2012	348,548	13,347	Single-label	1
DMOZ-2011	594,158	27,875	Multi-label	1.03
SWIKI-2011	346,299	50,312	Multi-label	1.85
LWIKI	1,617,899	614,428	Multi-label	3.26

Table: **Dataset statistics**

Comparison Methods

Flat baselines

- **SVM** - one-vs-rest binary support vector machines
- **LR** - one-vs-rest regularized logistic regression

Hierarchical baselines

- **Top-down SVM (TD)**[**Liu et al., SIGKDD'05**] - a pachinko machine style SVM
- **Hierarchical SVM (HSVM)**[**Tsochantaridis et al., JMLR'05**] - a large-margin discriminative method with path dependent discriminant function
- **Hierarchical Orthogonal Transfer (OT)**[**Lin et al., ICML'11**] - a large-margin method enforcing orthogonality between the parent and the children
- **Hierarchical Bayesian Logistic Regression (HBLR)**[**Gopal et al., NIPS'12**]- a bayesian methods to model hierarchical dependencies among class labels using multivariate logistic regression

Flat Baselines Comparison - I

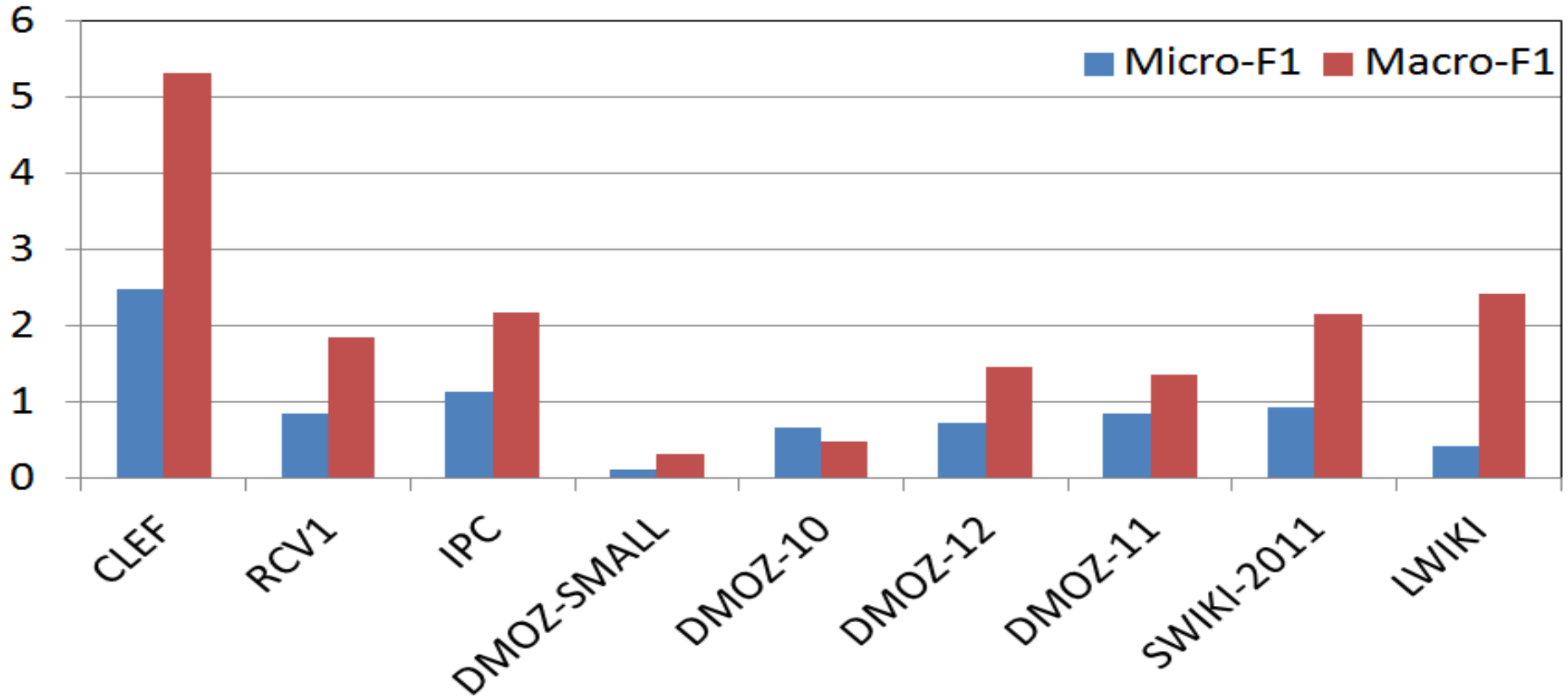


Figure: **Performance improvement: HR-SVM vs. SVM**

Flat Baselines Comparison - II

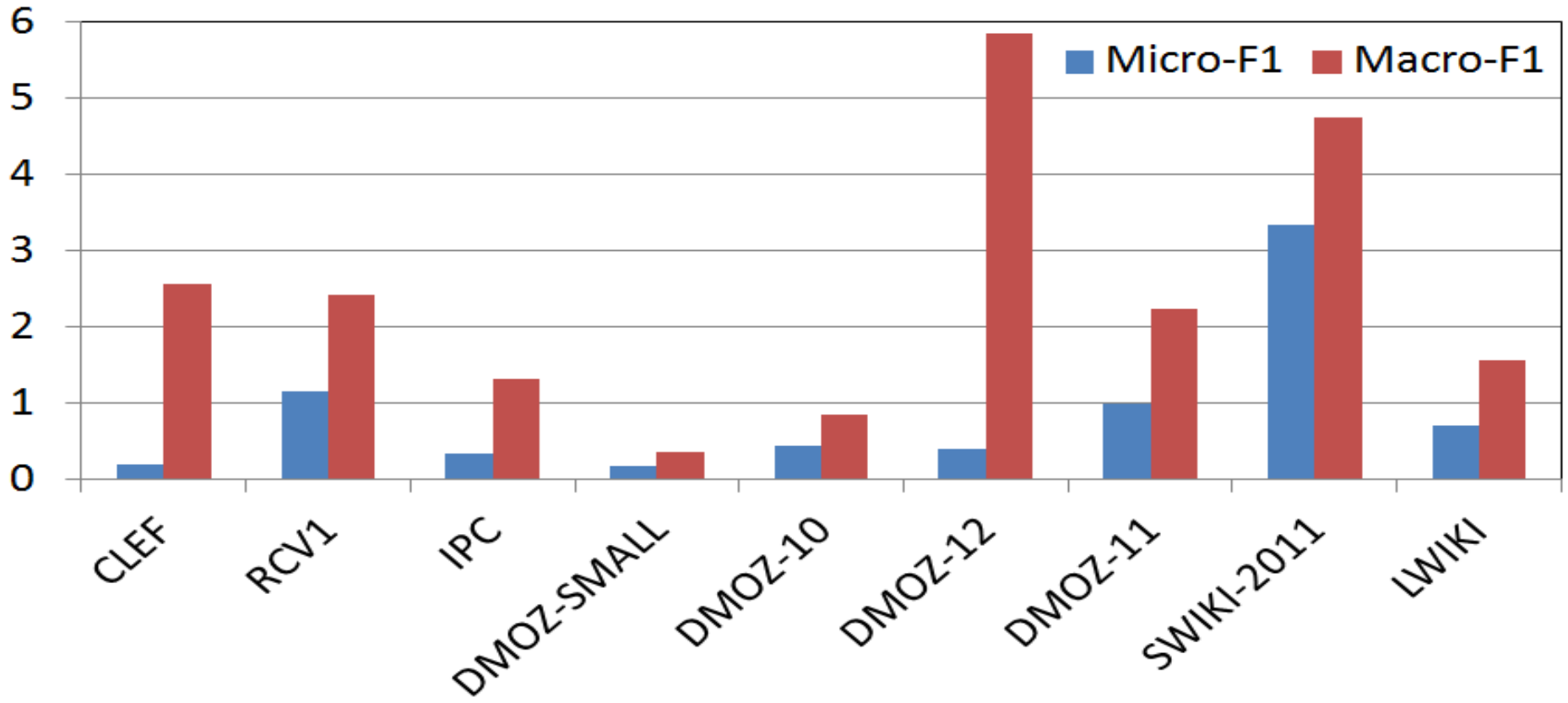


Figure: **Performance improvement: HR-LR vs. LR**

Hierarchical Baselines Comparison

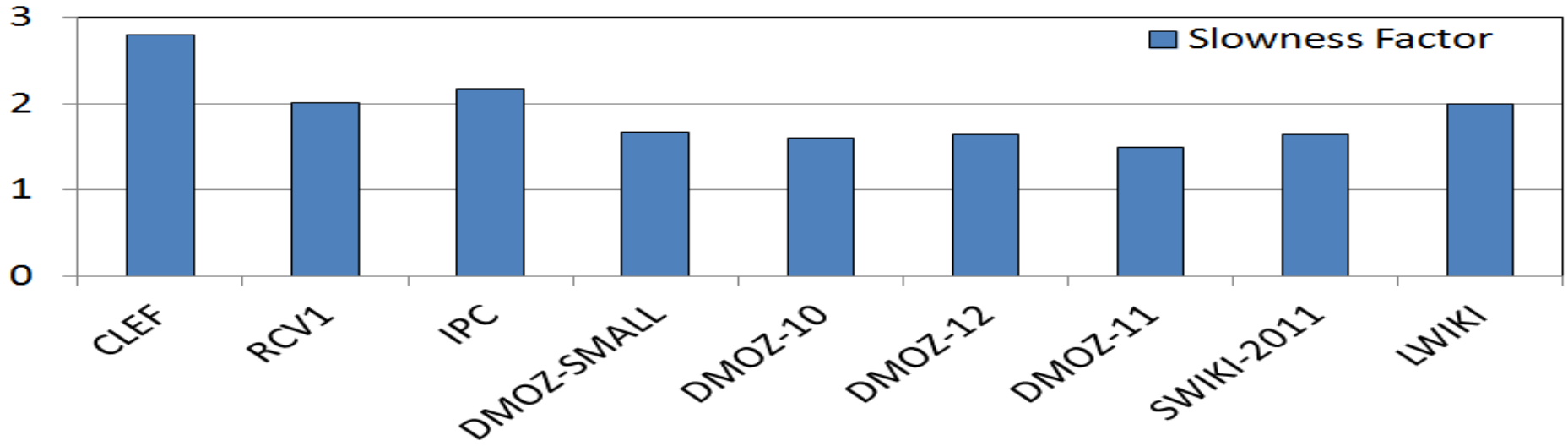
Datasets	HR-SVM	HR-LR	TD	HSVM	OT	HBLR
CLEF	80.02	80.12	70.11	79.72	73.84	**81.41**
RCV1	**81.66**	81.23	71.34	NA	NS	NA
IPC	54.26	55.37	50.34	NS	NS	**56.02**
DMOZ-SMALL	45.31	45.11	38.48	39.66	37.12	**46.03**
DMOZ-2010	**46.02**	45.84	38.64	NS	NS	NS
DMOZ-2012	**57.17**	53.18	55.14	NS	NS	NS
DMOZ-2011	**43.73**	42.27	35.91	NA	NS	NA
SWIKI-2011	**41.79**	40.99	36.65	NA	NA	NA
LWIKI	**38.08**	37.67	NA	NA	NA	NA

[**NA - Not Applicable; NS - Not Scalable**]

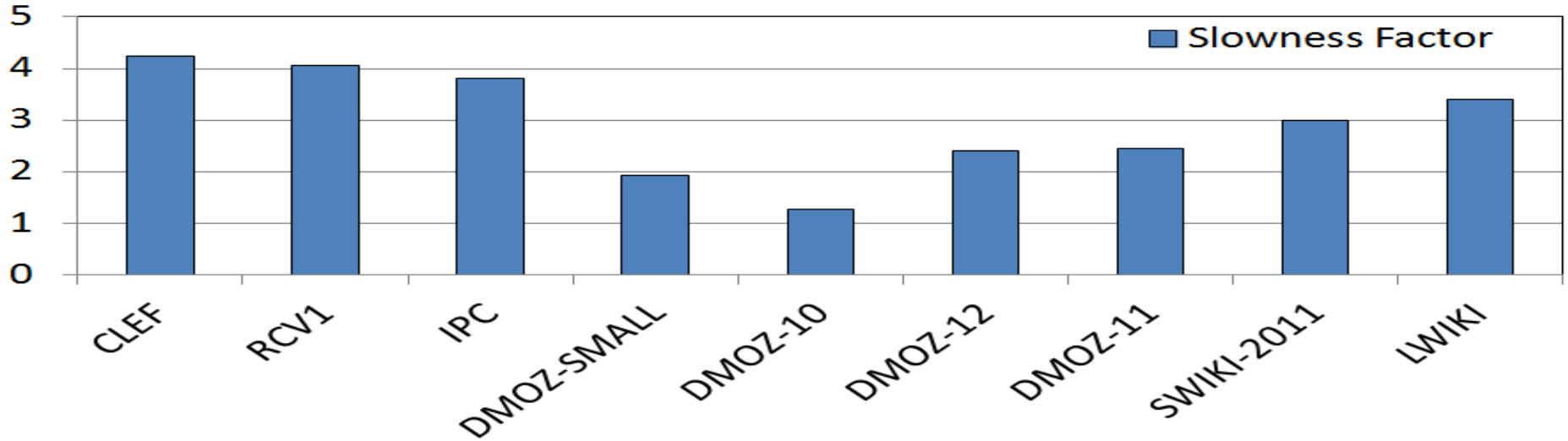
Table: **Micro-F1 performance comparison**

Runtime Comparison - flat baselines

HR-SVM vs. SVM



HR-LR vs. LR



Runtime Comparison - hierarchical baselines

Datasets	HR-SVM	HR-LR	TD	HSVM	OT	HBLR
CLEF	0.42	1.02	0.13	3.19	1.31	3.05
RCV1	0.55	11.74	0.21	NA	NS	NA
IPC	6.81	15.91	2.21	NS	NS	31.20
DMOZ-SMALL	0.52	3.73	0.11	289.60	132.34	5.22
DMOZ-2010	8.23	123.22	3.97	NS	NS	NS
DMOZ-2012	36.66	229.73	12.49	NS	NS	NS
DMOZ-2011	58.31	248.07	16.39	NA	NS	NA
SWIKI-2011	89.23	296.87	21.34	NA	NA	NA
LWIKI	2230.54	7282.09	NA	NA	NA	NA

[**NA - Not Applicable; NS - Not Scalable**]

Table: **Training runtime comparison (in mins)**

Cost-sensitive Learning, Charuvaka & Rangwala, ECML'15

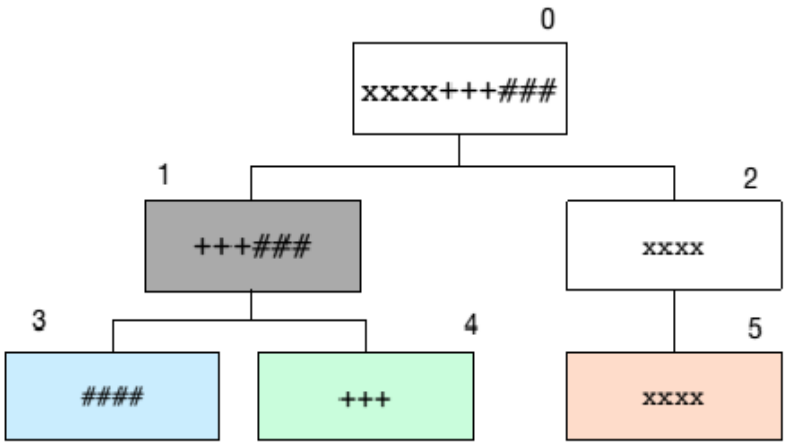
Motivation

- Drawbacks of Recursive Regularization
  - scalable, but more expensive to train than flat classification
  - requires specialized implementation and communication between processing node
  - Does not deal with class imbalance directly

Objective

- Decouple models so that they can be trained in parallel without dependencies between models
- Account for class imbalance in the optimization framework

Hierarchical Regularization Re-examination - I

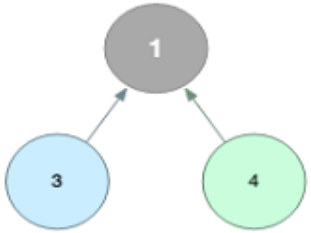


Hypothetical hierarchy

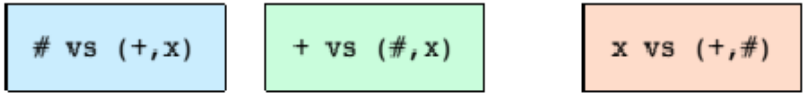
$$\min_{\mathbf{W}} \mathcal{L}(f(\mathbf{X}, \mathbf{W}), \mathbf{Y}) + \lambda \Omega(\mathbf{W})$$

Empirical Loss

Regularization



Regularizer promotes similarity



One-vs-rest classification

Loss promotes dissimilarity

Hierarchical Regularization Re-examination - II

- Opposing learning influences:
  - **loss term** - model for a node is forced to be dissimilar to all other nodes
  - **regularization term** - model is forced to be similar to its neighbors; greater similarity to nearer neighbors
- Resultant effect:
  - Mistakes on negative examples that come from near nodes is less severe than those coming from far nodes while still taking advantage of the hierarchy

Cost-sensitive Loss

- Consider the loss term for class "t" which is separable over examples

$$\sum_i loss(y_i, \mathbf{w}_i^T \mathbf{x}_i)$$

- Each loss value is multiplied by importance of the example for this class

$$\sum_i loss(y_i, \mathbf{w}_i^T \mathbf{x}_i) \times \phi(t, y_i)$$

- This is an example of "instance-based" cost sensitive learning

$$c_i^t = \phi(t, y_1)$$

Hierarchical Costs

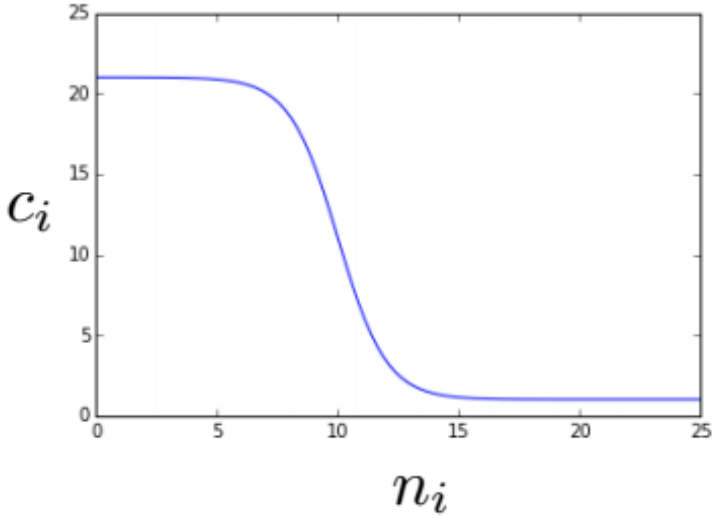
How to define costs based on hierarchy?

- **Tree Distance (TrD)** - undirected graph distance between between nodes
- **Number Common Ancestors (NCA)** - the number of ancestors in common to target class and class label
- **Exponentiated Tree Distance (ExTrD)** - squash tree distance into a suitable range using validation

Imbalance Costs

- Using the same formulation of cost-sensitive learning, data imbalance can also be addressed
- Due to very large skew, inverse class size can result in extremely large weights. Fix using squashing function shown in Fig.
- Multiply to combine with Hierarchical costs

$$c_i = 1 + L/[1 + exp|n - n_0|]$$



n_i = num examples
n_0, L = user defined constants

Experiments

Dataset

- For comparison purpose same dataset has been used as proposed in the paper [**Gopal and Yang, SIGKDD'13**]

Comparison Methods
Flat baseline

- **LR** - one-vs-rest binary logistic regression is used in the conventional flat classification setting

Hierarchical baselines

- **Top-down Logistic Regression (TD-LR)** - one-vs-rest multi-class classifier trained at each internal node
- **HR-LR** [**Gopal and Yang, SIGKDD'13**] - a recursive regularization approach based on hierarchical relationships

Results (Hierarchical Costs)

Datasets		Micro-F1 (↑)	Macro-F1 (↑)	hF1 (↑)	TE (↓)
CLEF	LR	79.82	53.45	85.24	0.994
	TrD	80.02	55.51	**85.39**	0.984
	NCA	80.02	57.48	85.34	0.986
	ExTrD	**80.22**	**57.55**†	85.34	**0.982**
DMOZ-SMALL	LR	46.39	30.20	67.00	3.569
	TrD	**47.52**‡	**31.37**‡	**68.26**	**3.449**
	NCA	47.36‡	31.20‡	68.12	3.460
	ExTrD	47.36‡	31.19‡	68.20	3.456
IPC	LR	55.04	48.99	72.82	1.974
	TrD	55.24‡	50.20‡	73.21	1.954
	NCA	**55.33**‡	**50.29**‡	**73.28**	**1.949**
	ExTrD	55.31‡	**50.29**‡	73.26	1.951
RCV1	LR	78.43	60.37	80.16	0.534
	TrD	79.46‡	60.61	82.83	0.451
	NCA	**79.74**‡	60.76	**83.11**	**0.442**
	ExTrD	79.33‡	**61.74**†	82.91	0.466

Table: **Performance comparison of hierarchical costs**

Results (Imbalance Costs)

Datasets		Micro-F1 (↑)	Macro-F1 (↑)	hF1 (↑)	TE (↓)
CLEF	IMB + LR	79.52	53.11	85.19	1.002
	IMB + TrD	79.92	52.84	85.59	0.978
	IMB + NCA	79.62	51.89	85.34	0.994
	IMB + ExTrD	**80.32**	**58.45**	**85.69**	**0.966**
DMOZ-SMALL	IMB + LR	48.55‡	32.72‡	68.62	3.406
	IMB + TrD	**49.03**‡	33.21‡	69.41	3.334
	IMB + NCA	48.87‡	33.27‡	69.37	3.335
	IMB + ExTrD	tbf49.03‡	**33.34**‡	**69.54**	**3.322**
IPC	IMB + LR	55.04	49.00	72.82	1.974
	IMB + TrD	55.60‡	**50.45**†	73.56	1.933
	IMB + NCA	55.33	50.29	73.28	1.949
	IMB + ExTrD	**55.67**‡	50.42	**73.58**	**1.931**
RCV1	IMB + LR	78.59‡	60.77	81.27	0.511
	IMB + TrD	**79.63**‡	61.04	**83.13**	**0.435**
	IMB + NCA	79.61	61.04	82.65	0.458
	IMB + ExTrD	79.22	**61.33**	82.89	0.469

Table: **Peformance comparison with imbalance cost included**

Results (our best with other methods)

Datasets		Micro-F1 (↑)	Macro-F1 (↑)	hF1 (↑)	TE (↓)
CLEF	TD-LR	73.06	34.47	79.32	1.366
	LR	79.82	53.45	85.24	0.994
	HR-LR	80.12	55.83	NA	NA
	HierCost	**80.32**	**58.45**†	**85.69**	**0.966**
DMOZ-SMALL	TD-LR	40.90	24.15	**69.99**	**3.147**
	LR	46.39	30.20	67.00	3.569
	HR-LR	45.11	28.48	NA	NA
	HierCost	**49.03**‡	**33.34**‡	69.54	3.322
IPC	TD-LR	50.22	43.87	69.33	2.210
	LR	55.04	48.99	72.82	1.974
	HR-LR	55.37	49.60	NA	NA
	HierCost	**55.67**‡	**50.42**†	**73.58**	**1.931**
RCV1	TD-LR	77.85	57.80	**88.78**	0.524
	LR	78.43	60.37	80.16	0.534
	HR-LR	**81.23**	55.81	NA	NA
	HierCost	79.22‡	**61.33**	82.89	**0.469**

Table: **Performance comparison of HierCost with other baseline methods**

Runtime comparison

Datasets	TD-LR	LR	HierCost
CLEF	<1	<1	<1
DMOZ-SMALL	4	41	40
IPC	27	643	453
RCV1	20	29	48
DMOZ-2010	196	15191	20174
DMOZ-2012	384	46044	50253

Table: **Total training runtimes (in mins)**

Demo of Software

- Freely available for research and education purpose at:

  https://cs.gmu.edu/∼mlbio/HierCost/

- Software: implemented in python using **scikit-learn** machine learning and **svmlight-loader** package
- Other prerequisite package:
  - numpy
  - scipy
  - networkx
  - pandas

Command Line Interface (CLI) Options

- Two main CLI is exposed for easy training and classification
  - **train.py**
    - -d : train data file path
    - -t : hierarchy file path
    - -m : path to save learned model parameters
    - -f : number of features
    - -r : regularization parameter (0; default = 1)
    - -i : to incorporate imbalance cost
    - -c : cost function to use (lr, trd, nca, etrd) (default = 'lr')
    - -u : for multi-label classification (default = single-label classification)
    - -n: set of nodes to train (for parallelization)
  - **predict.py**
    - -p : path to save prediction for test examples
    - -m, -d, -t, -f, -u : similar functionalities

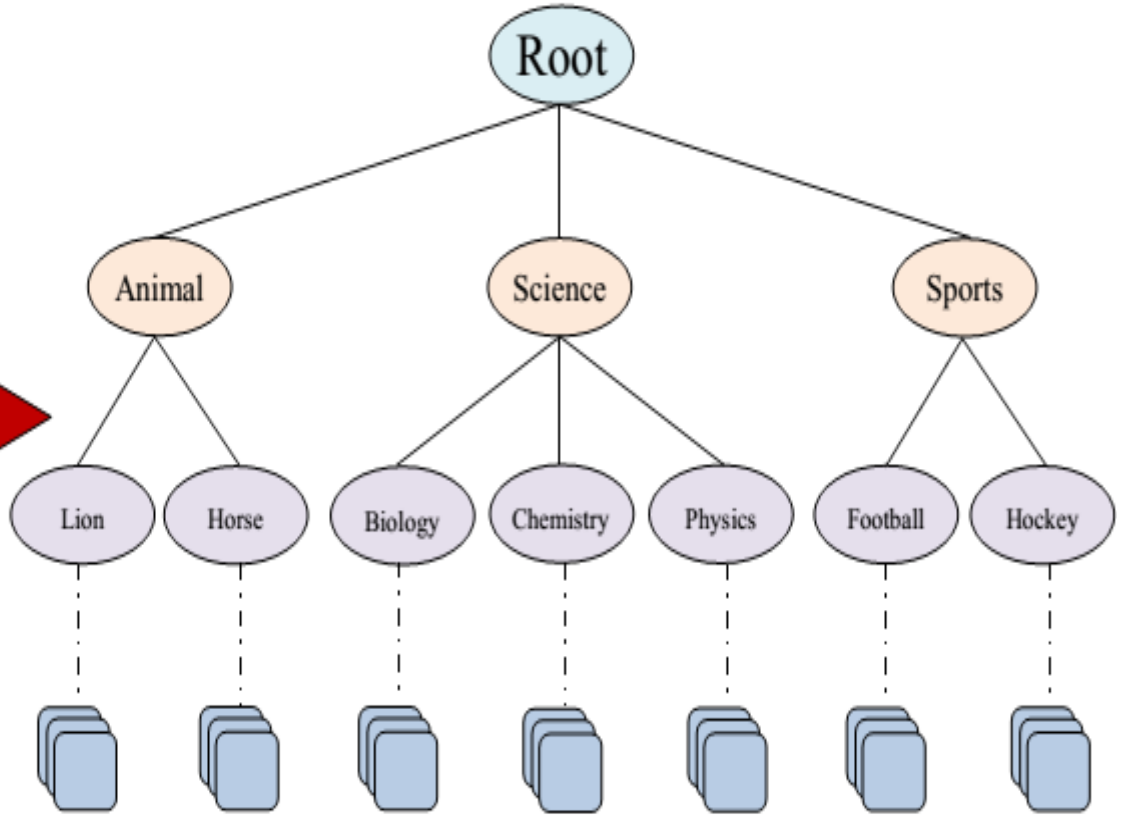
Inconsistent Hierarchy
(5 minutes break)

Motivation

Predefined Hierarchy

- Hierachy defined by the domain experts
- Reflects human-view of the domain - may not be optimal for machine learning classification algorithms

defines



Case Study

Question

- Can we trust the predefined expert's hierarchy for achieving the good classification performance?
- Can we tweak (adjust) the hierarchy to improve the performance?

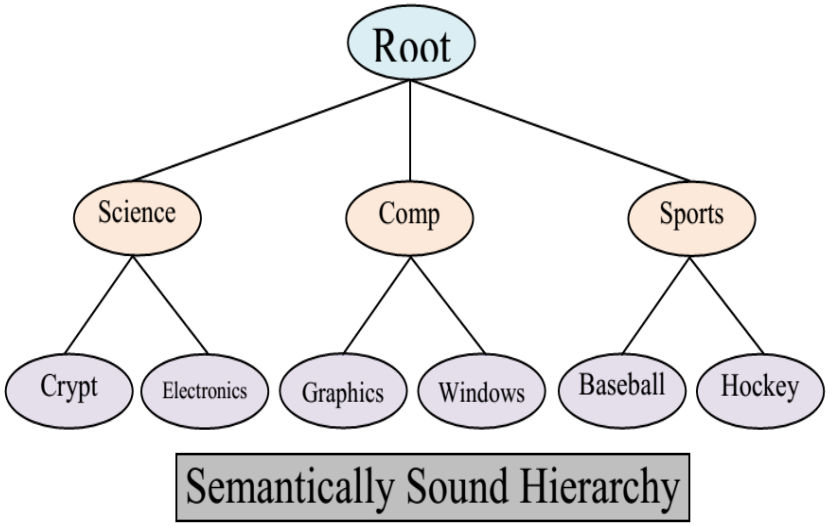
Case Study

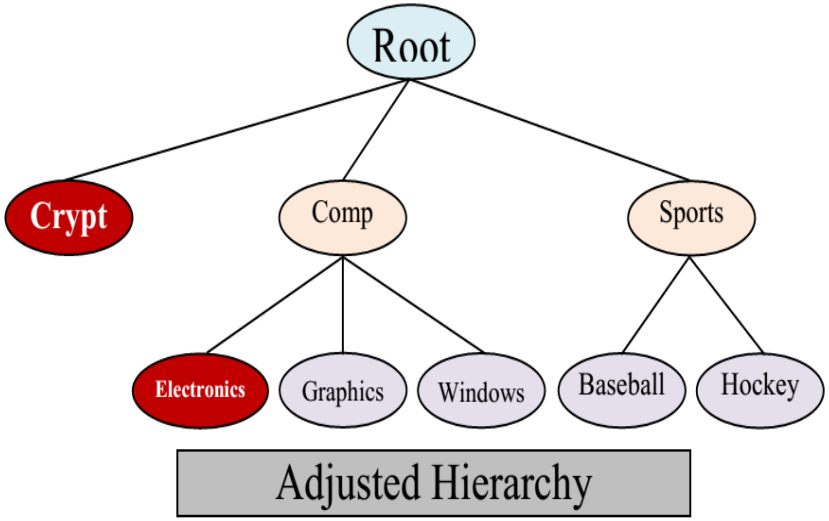
Question

- Can we trust the predefined expert's hierarchy for achieving the good classification performance?
- Can we tweak (adjust) the hierarchy to improve the performance?

Answer

- Case study on subset of newsgroup dataset





"Adjusted hierarchy classification performance **comparatively better** than semantically sound hierarchy"

Reason for Inconsistencies within Predefined Hierarchy - I

- Hierarchy is designed for the sole purpose of **easy search and navigation** without taking classification into consideration

Reason for Inconsistencies within Predefined Hierarchy - I

- Hierarchy is designed for the sole purpose of **easy search and navigation** without taking classification into consideration
- Hierarchy is created based on semantics which is independent of data

whereas

classification depends on **data characteristics** such as term frequency

"Our expectation: **data-driven hierarchy** can be much powerful"

Reason for Inconsistencies within Predefined Hierarchy - I

- Hierarchy is designed for the sole purpose of **easy search and navigation** without taking classification into consideration
- Hierarchy is created based on semantics which is independent of data

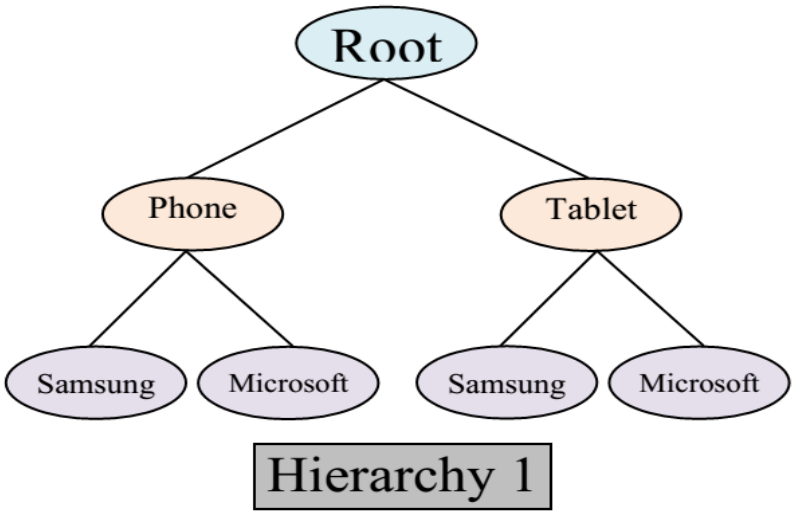
  whereas

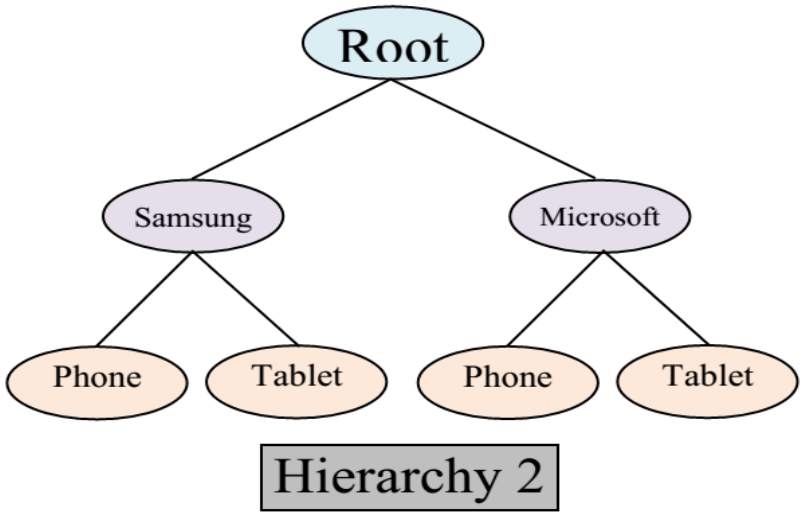
  classification depends on **data characteristics** such as term frequency

  "Our expectation: **data-driven hierarchy** can be much powerful"
- Apriori it is not clear to domain experts when to generate new nodes (hierarchy expansion) or merge two or more nodes (link creation) in the hierarchy

Reason for Inconsistencies within Predefined Hierarchy - II

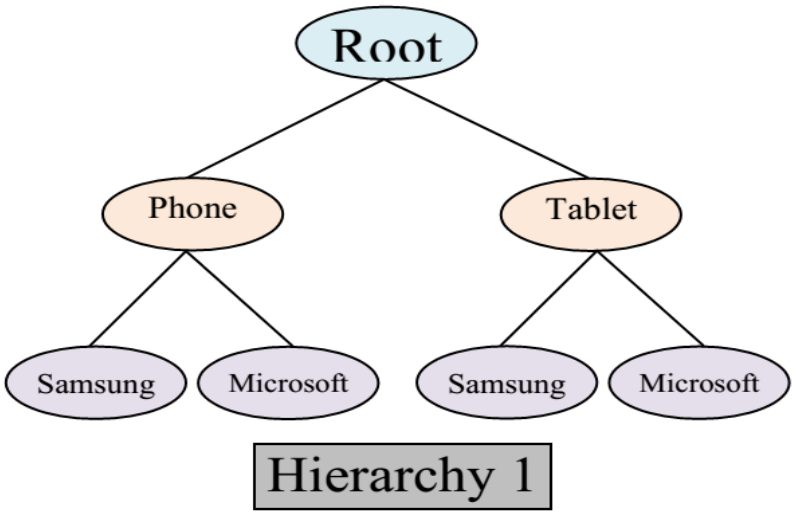
- Given list of categories: different experts may come up with **different hierarchies** with completely different classification results

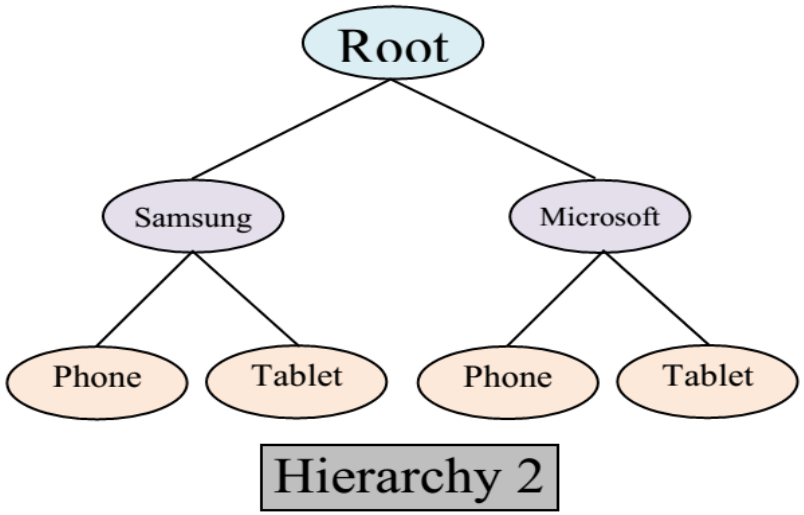




Reason for Inconsistencies within Predefined Hierarchy - II

- Given list of categories: different experts may come up with **different hierarchies** with completely different classification results





- Large number of classes with confusing labels pose a unique challenge for manual design of good hierarchy

Reason for Inconsistencies within Predefined Hierarchy - III

- Dynamic changes can affect hierarchical relationships
- "**Flood**" is the sub-group of geography class but during chennai flood it becomes political news

2015: Chennai **FLOOD** Image

Sub-group of **Geography** class

During chennai flood becomes sub-group of **Political** class

"Predefined Hierarchy"

What we want?

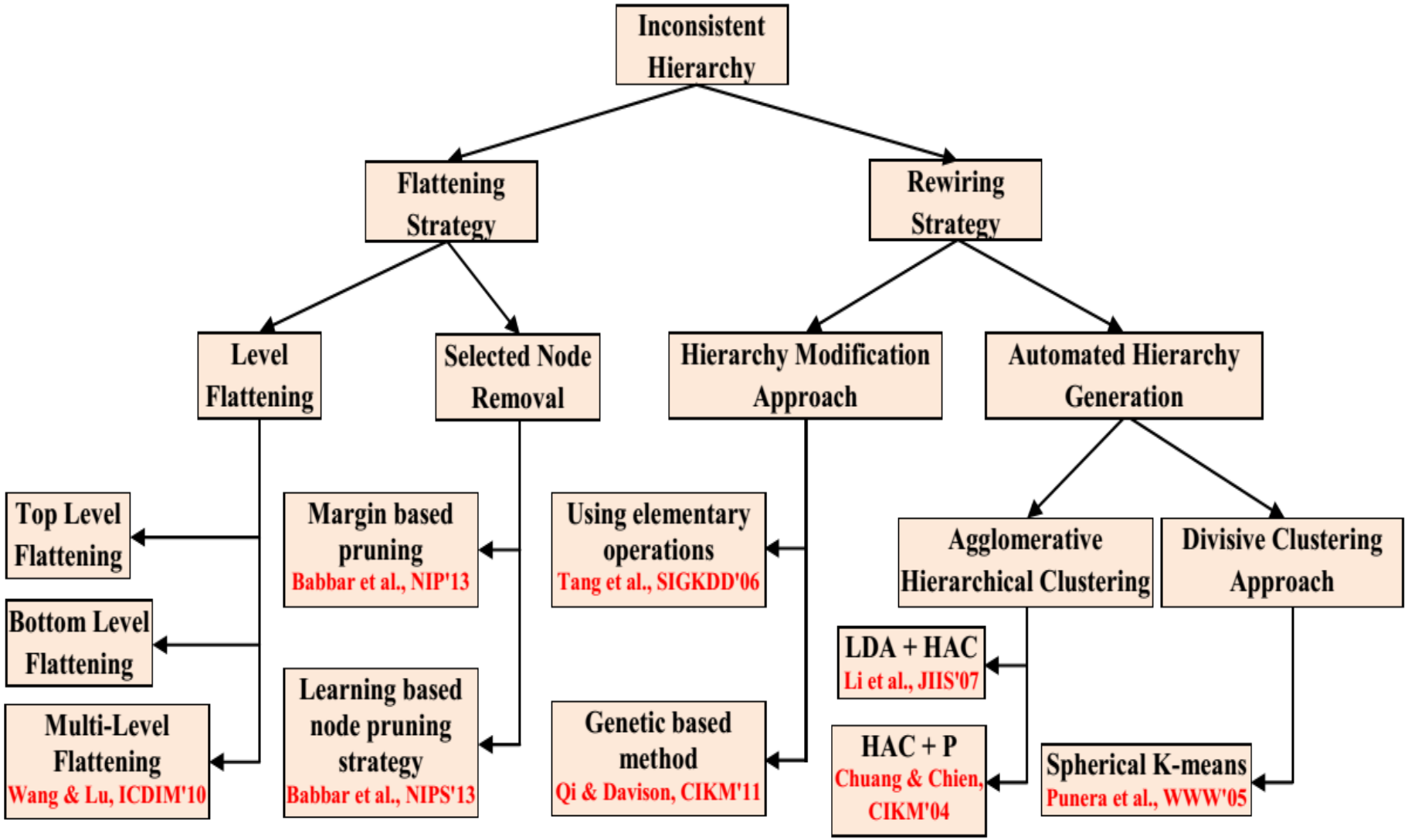
"Predefined Hierarchy"

$\downarrow$

"Data-driven Hierarchy"

(for improving classification performance)

Literature Overview



Flattening Strategy

Motivation

- For large scale datasets top-down (TD) hierarchical models are preferred over flat models due to computational benefite (training and prediction time)
- TD models performance suffers due to **error propagation** *i.e.* compounding of errors from misclassifications at higher levels which cannot be rectified at the lower levels

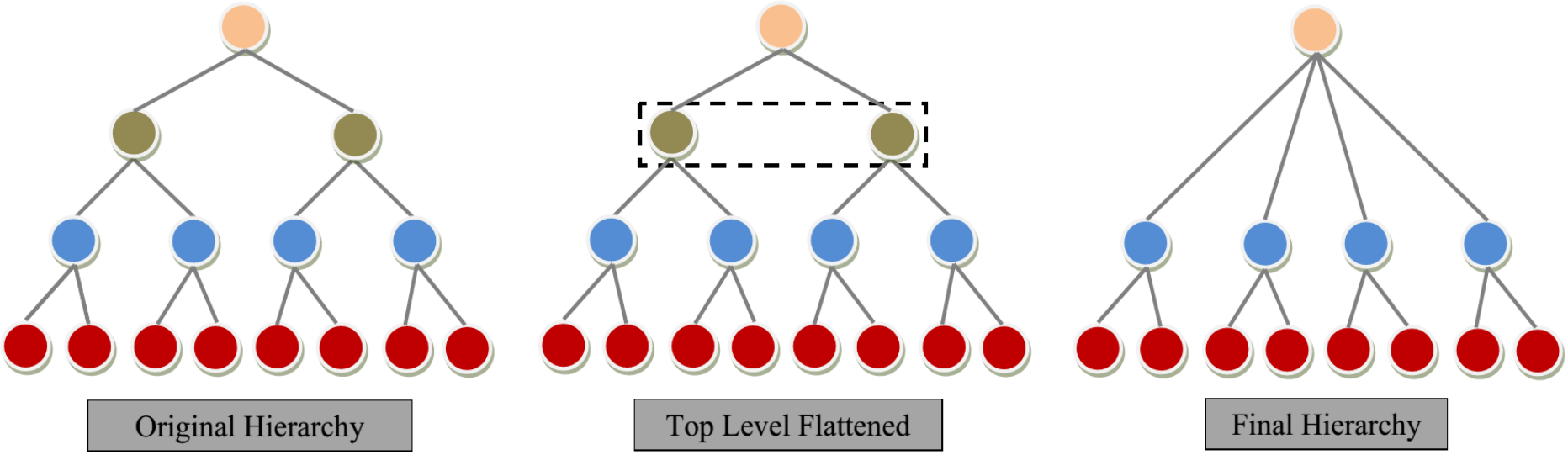
Objective

- Modify predefined hierarchy by removing (flattening) inconsistent nodes to improve the classification performance of TD models
- Reduces top-down error propagation due to less number of decisions for classifying unlabeled example

Flatten Hierarchies, Wang and Lu, ICDIM'10

Level Flattening Techniques

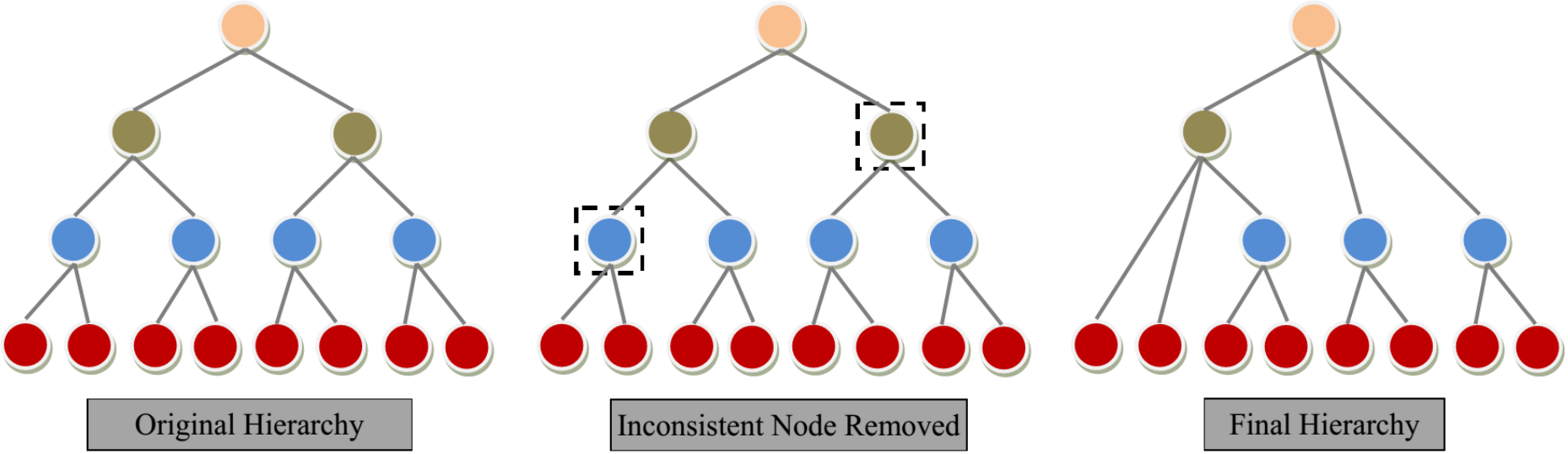
- Single or multiple levels within the hierarchy is flattened
- Based on level(s) flattened various methods exist, *for e.g.* TLF, BLF, MLF



- Drawback - all nodes in the level(s) is identified as inconsistent which may not be true; resulting in poor classification performance

Selected Inconsistent Node Removal (Flattening)

- Rather than flattening entire level(s) only **subset** of the inconsistent nodes are removed from the hierarchy

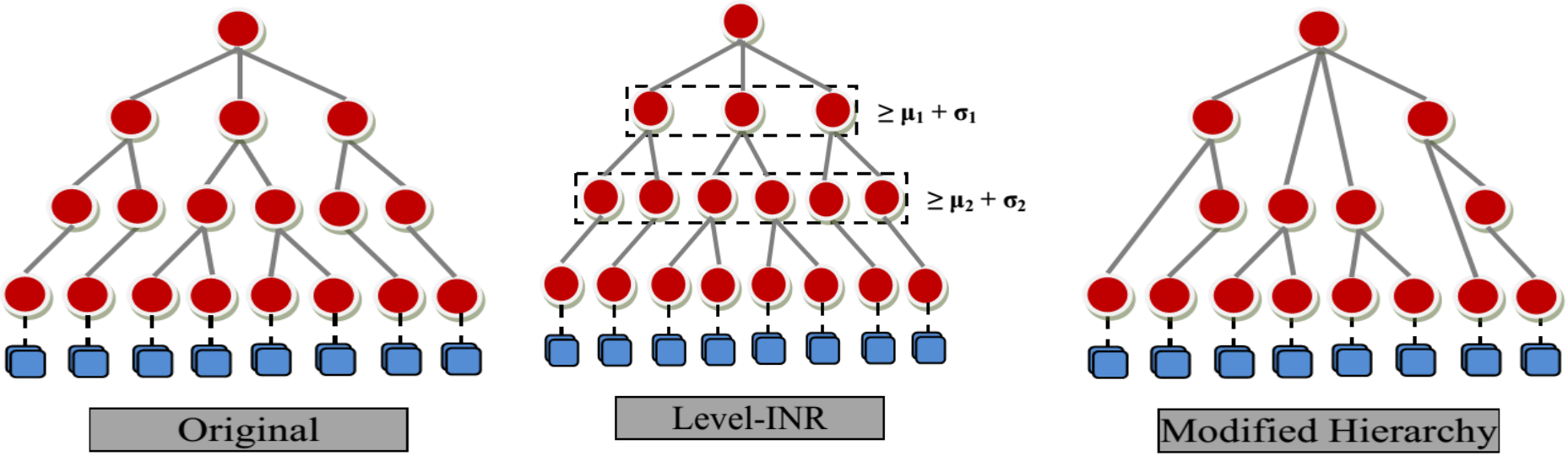


- Criterion to decide inconsistent nodes - degree of error made at the node, margin-based or learning-based strategy
- Comparatively better performance than level flattening methods

Inconsistent Node Removal - I

Local Approach for INR (Level-INR)

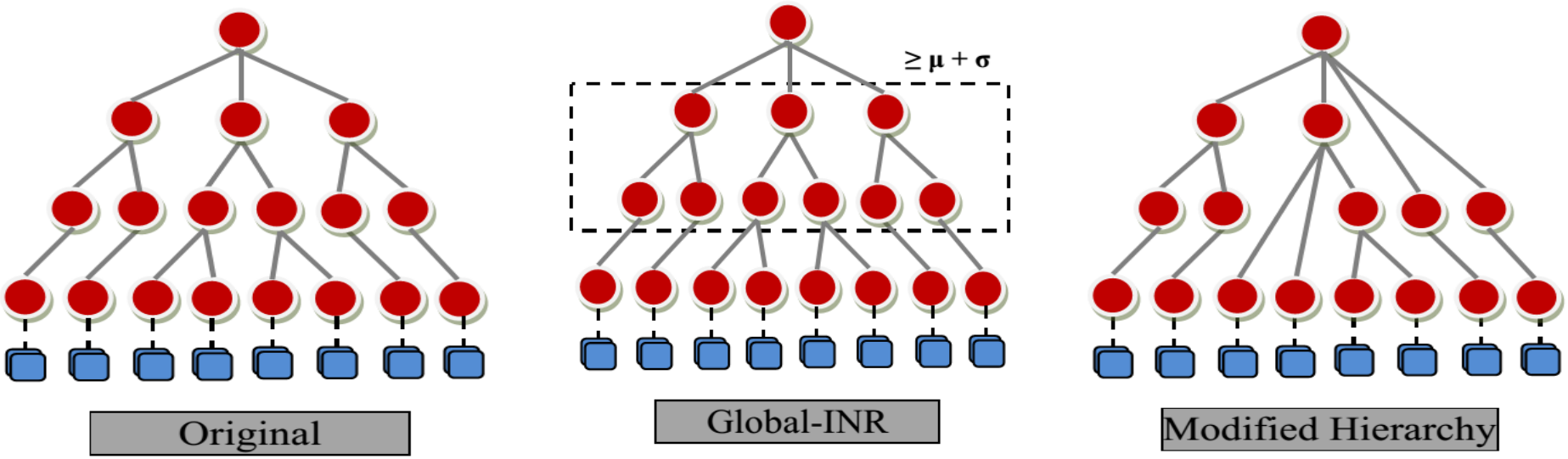
- Inconsistent set of nodes determined for each level based on loss function values (such as logistic loss) obtained for nodes at that level
- Criterion for flattening nodes - mean and standard deviation per level
- Different levels have different threshold for node flattening



Inconsistent Node Removal - II

Global Approach for INR (Global-INR)

- Inconsistent node determined by considering loss function of all internal nodes in the hierarchy
- Criterion for flattening nodes - mean and standard deviation of all nodes
- All levels have same threshold for node flattening



Comparison Methods

- One-vs-rest models is trained for each node (except root) in the hierarchy
- Predictions are made starting from the root node and recursively selecting the best child nodes until a leaf node is reached

Hierarchical baselines

- **Top-down Logistic Regression (TD-LR)** - predefined hierarchy used for training the models
- **Level Flattening** [**Wang and Lu, ICDIM'10**] - flattened hierarchy used for training the models, based on level flattened we have:
  - TLF - Top level flattened
  - BLF - Bottom level flattened and
  - MLF - Multiple level flattened
- **MTA** [**Babbar et al., NIP'13**] - hierarchy is modified using the margin value computed at each node

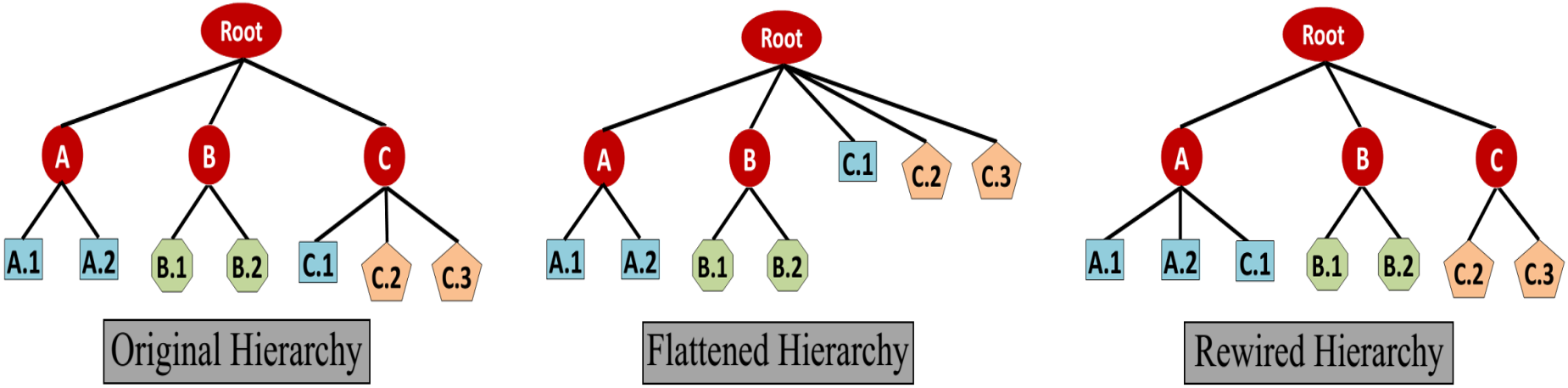
Performance Results

- Level-INR and Global-INR performed comparatively better than other TD hierarchical baselines
- Global approach has better performance compared to local approach

Datasets		Top-down Hierarchical Baselines					Proposed Models	
		TD-LR	TLF	BLF	MLF	MTA	Level-INR	Global-INR
CLEF	$\mu F_1(\uparrow)$	72.74	75.84	73.76	X	74.48	75.25	**77.14**†
	$MF_1(\uparrow)$	35.92	38.45	40.93	X	39.53	39.89	**46.54**‡
DIATOMS	$\mu F_1(\uparrow)$	53.27	56.93	53.27	X	58.36	58.32	**61.31**‡
	$MF_1(\uparrow)$	44.46	45.17	44.30	X	45.21	48.77	**51.85**‡
IPC	$\mu F_1(\uparrow)$	49.32	51.28	50.36	X	51.36	50.40	**52.30**†
	$MF_1(\uparrow)$	42.51	44.99	43.74	X	42.80	43.26	**45.65**†
DMOZ-SMALL	$\mu F_1(\uparrow)$	45.10	45.48	44.34	45.80	46.01	45.43	**46.61**†
	$MF_1(\uparrow)$	30.65	30.60	30.94	30.62	30.82	30.34	**31.26**‡
DMOZ-2010	$\mu F_1(\uparrow)$	40.22	41.32	40.34	41.77	41.82	40.71	**42.37**
	$MF_1(\uparrow)$	28.37	29.05	28.41	29.11	29.18	28.66	**30.11**
DMOZ-2012	$\mu F_1(\uparrow)$	50.13	50.32	50.11	48.05	50.31	49.90	**50.64**
	$MF_1(\uparrow)$	29.89	29.89	29.73	27.65	30.04	30.52	**30.58**

Drawbacks of Flattening Strategy

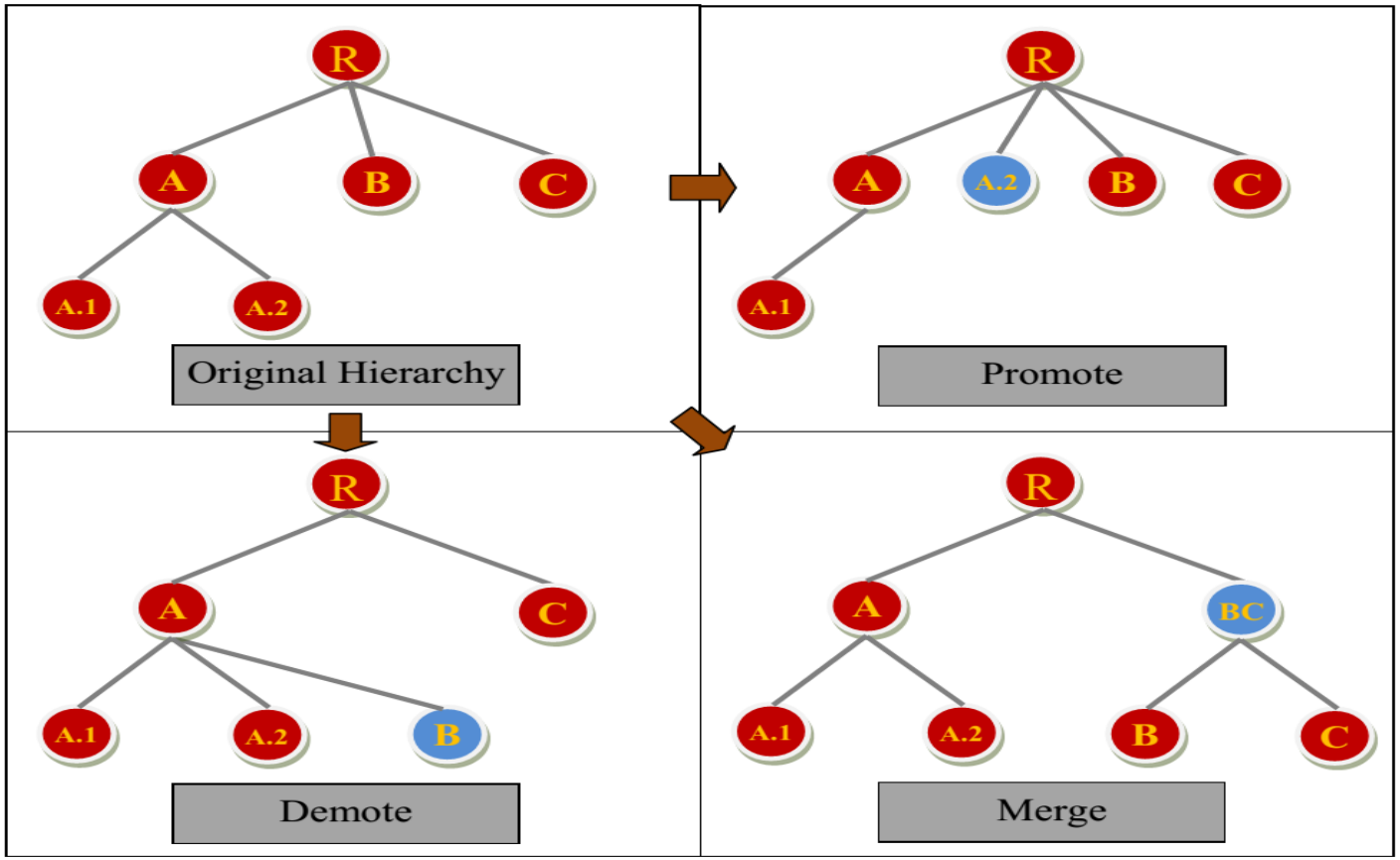
- Flattening strategy although useful upto certain extent has few limitations
  - Inability to deal with inconsistencies in different branches of the hierarchy



- Rewiring strategy can be used to resolve inconsistencies that occurs in different branch

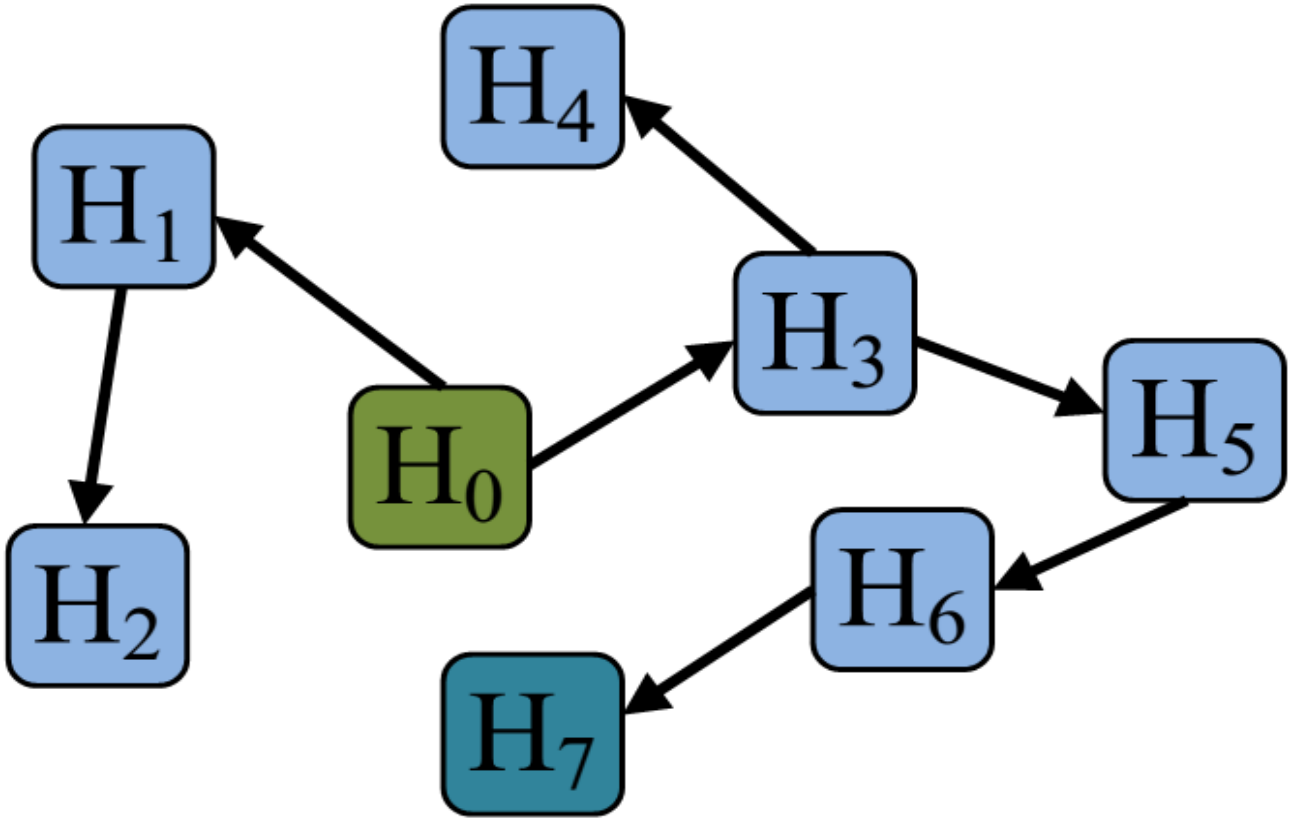
Hierarchy Adjustment using Elementary Operation - I, Tang et al., SIGKDD'06

- Elementary operation: **promote, demote, merge**



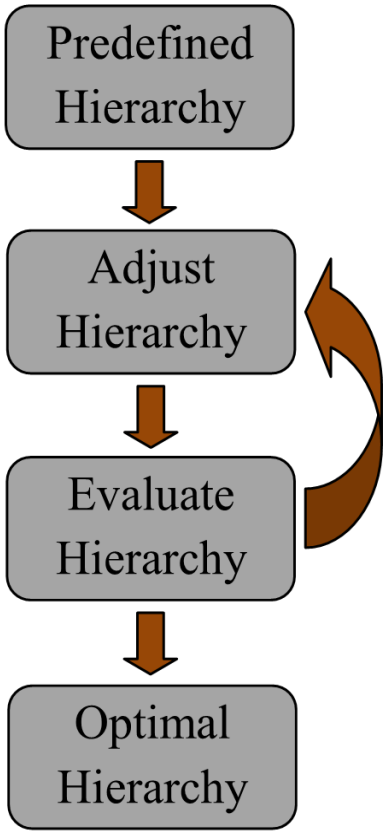
Hierarchy Adjustment using Elementary Operation - II

- **Assumption**: the optimal hierarchy is near the neighborhood of predefined taxonomy
- Search for **constrained optimal hierarchy** by applying sequence of elementary operations and searching in the hierarchy space



Proposed Hierarchy Adjustment Algorithm

- Wrapper based approach for hierarchy modification; requires hierarchy evaluation after each modification which is computationally expensive



Input: Predefined hierarchy (H_0), Training data ($\mathcal{D}_t$), Validation data ($\mathcal{D}_v$)

1. Generate neighbor hierarchies for H_0
2. Train hierarchical classification models for each neighbor on $\mathcal{D}_t$
3. Evaluate hierarchical classifiers on $\mathcal{D}_v$
4. Pick the best neighbor hierarchy as H_0
5. Until no improvement, repeat from step 1

Dataset for Experimental Evaluation

- Sub-branch from AOL database hierarchy is used for evaluation
- Data is small *w.r.t* no. of features and classes

Datasets	#Total Node	#Leaf Node	Height	#Training	#Features
Soc	83	69	4	5,248	34,003
Kids	299	244	5	15,795	48,115

Table: **Dataset Statistics**

Performance Results

- **Adjusted hierarchy** shows significant performance improvement in comparison to predefined (original) hierarchy and hierarchy generated using clustering approach

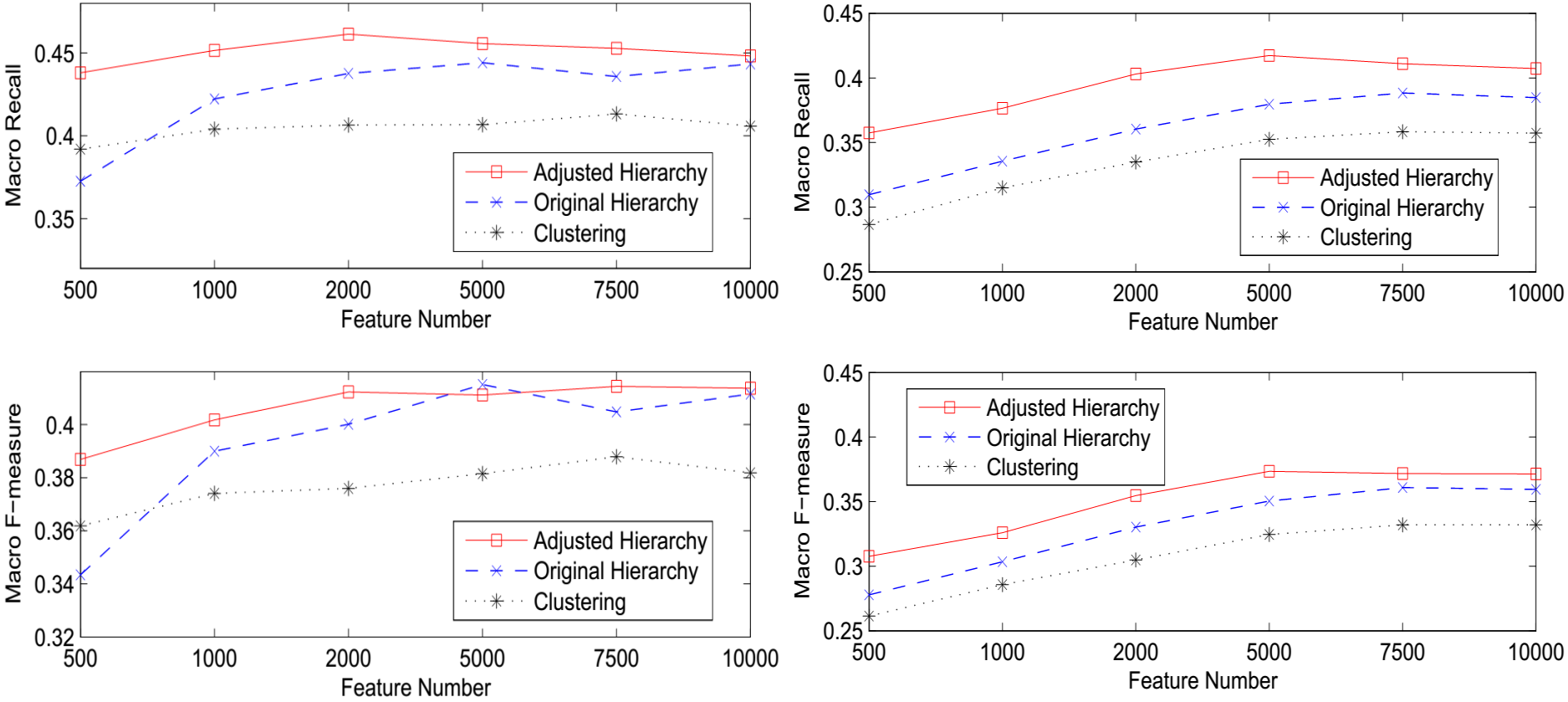


Figure: **Soc (left) and Kids (right) dataset results**

Other Evaluation Results

Fold	Iterations	Evaluations	Operations	Height
1	6	582	43	6
2	6	585	37	9
3	3	265	33	7
4	5	449	41	6
5	7	805	67	13
6	8	754	68	6
7	5	448	48	9
8	5	494	42	11
9	3	268	44	5
10	8	748	62	7
ave	5.6	539.8	48.5	7.9

Fold	Iterations	Evaluations	Operations	Height
1	12	4244	217	13
2	9	3014	172	10
3	10	3483	224	13
4	8	2275	190	13
5	7	2379	159	12
6	12	4181	166	15
7	10	3640	230	16
8	9	3312	185	18
9	11	3733	215	12
10	9	3174	221	16
ave	9.7	3343.5	197.9	13.8

Figure: **Soc (left) and Kids (right) performance statistics**

Filter based Rewiring Strategy

Motivation

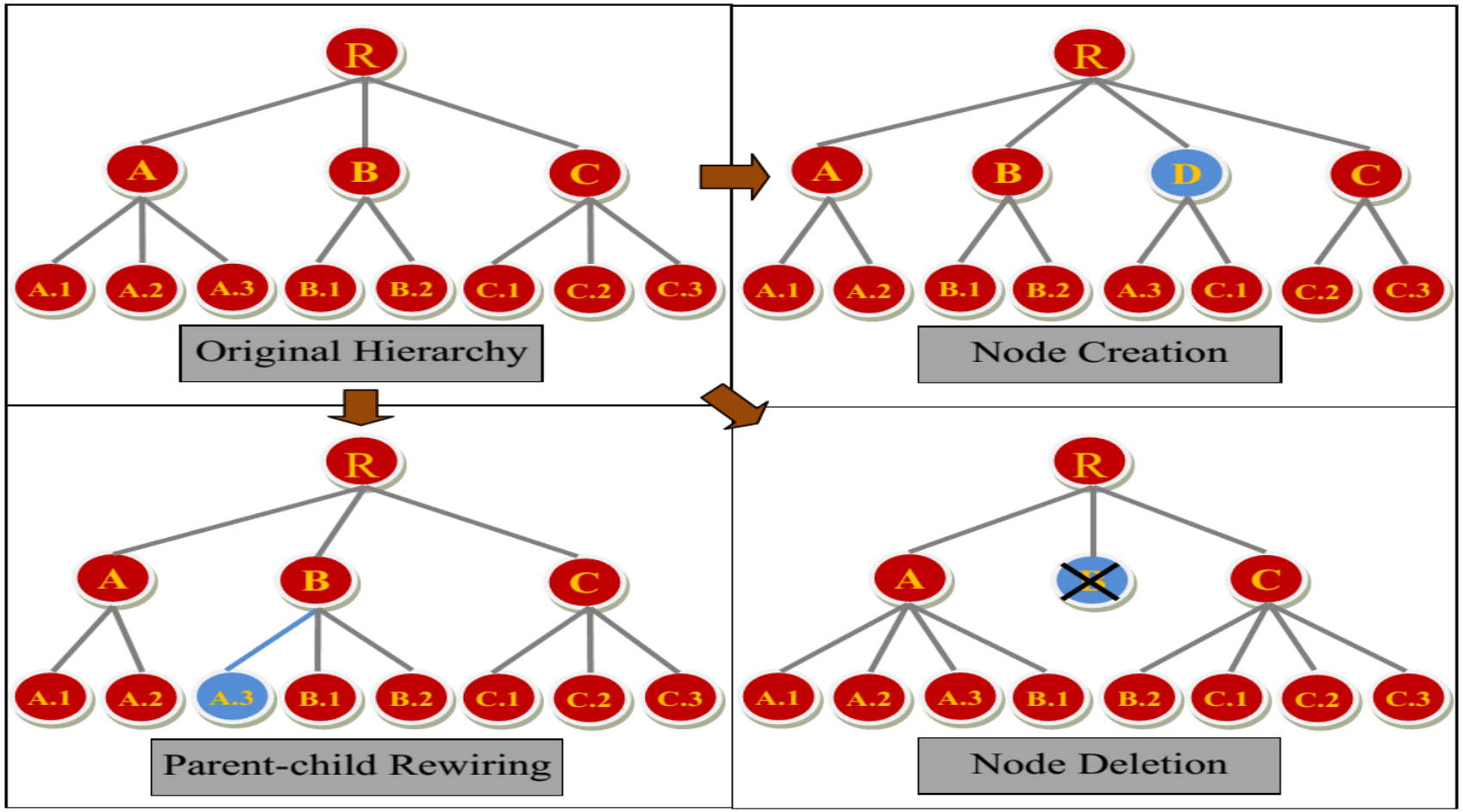
- For large scale datasets wrapper based approaches are intractable due to multiple hierarchy evaluations

Objective

- Modify predefined hierarchy using filter based rewiring strategy that does not requires multiple hierarchy evaluations
- Without significant loss in performance

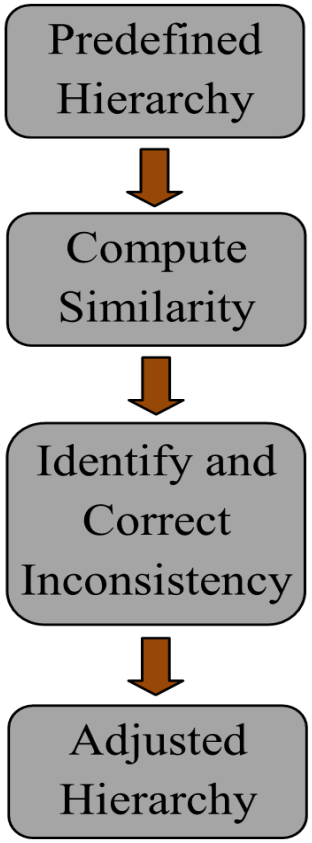
Proposed Rewiring Strategy

- Elementary operation: **node creation, parent-child rewiring, node deletion**



Proposed Rewiring Strategy Algorithm

- Filter based approach for hierarchy modification



Input: Predefined hierarchy (H_0), Train data ($\mathcal{D}_t$)

1. Compute pairwise similarity between classes defined in H_0 on $\mathcal{D}_t$
2. Group together most similar classes
3. Identify inconsistencies within the hierarchy
4. Apply elementary operations: node creation or parent-child rewiring to correct inconsistencies and obtain new hierarchy H_1
5. Perform post-processing step (node deletion) on H_1 to obtain new hierarchy H_2
6. Train and evaluate hierarchical classification models on H_2

Performance Results

Datasets		Flattening	Rewiring	
		Best TD Model (Global-INR)	Tang et al.	Proposed Filter Model
CLEF	$\mu F_1(\uparrow)$	77.14 (0.01)	**78.12 (0.16)**	78.00 (0.22)
	$MF_1(\uparrow)$	46.54 (0.06)	**48.83‡ (0.08)**	47.10 (0.03)
	$hF_1(\uparrow)$	79.06 (0.01)	**81.43 (0.03)**	80.14 (0.02)
DIATOMS	$\mu F_1(\uparrow)$	61.31 (0.53)	**62.34‡ (0.28)**	62.05‡ (0.10)
	$MF_1(\uparrow)$	51.85 (0.23)	**53.81‡ (0.11)**	52.14‡ (0.14)
	$hF_1(\uparrow)$	62.80 (0.04)	**64.28 (0.22)**	63.24 (0.13)
IPC	$\mu F_1(\uparrow)$	52.30 (0.12)	53.94† (0.24)	**54.28† (0.18)**
	$MF_1(\uparrow)$	45.65 (0.11)	**46.10† (0.21)**	46.04† (0.22)
	$hF_1(\uparrow)$	64.73 (0.12)	67.23 (0.24)	**68.34 (0.18)**
DMOZ-SMALL	$\mu F_1(\uparrow)$	46.61 (0.28)	NS	**48.25‡ (0.13)**
	$MF_1(\uparrow)$	31.26 (0.64)		**33.92‡ (0.22)**
	$hF_1(\uparrow)$	63.37 (0.44)		**66.18 (0.15)**
DMOZ-2010	$\mu F_1(\uparrow)$	42.37 (0.27)	NS	**43.10 (0.28)**
	$MF_1(\uparrow)$	30.11 (0.64)		**31.21 (0.34)**
DMOZ-2012	$\mu F_1(\uparrow)$	50.64 (0.22)	NS	**51.82 (0.02)**
	$MF_1(\uparrow)$	30.58 (0.28)		**31.24 (0.12)**
	$hF_1(\uparrow)$	73.19 (0.02)		**74.21 (0.03)**

Runtime Comparison

Datasets	Flattening	Rewiring	
	Best TD Model (Global-INR)	Tang et al.	Proposed Filter Model
CLEF	3.5	59	7.5
DIATOMS	10	268	24
IPC	830	26432	1284
DMOZ-SMALL	65	NS	168
DMOZ-2010	25600	NS	42000
DMOZ-2012	63000	NS	94800

Table: **Total training runtimes (in mins)**

Elementary Operation Comparisons

# elementary operation executed	**CLEF**	**DIATOMS**	**IPC**
Tang et al. (promote, demote, merge)	52	156	412
Proposed Rewiring Filter Model (node creation, PCRewire, node deletion)	25	34	42

Learning using Multiple Hierarchies (MTL), Charuvaka and Rangwala, ICDM'12

Motivation

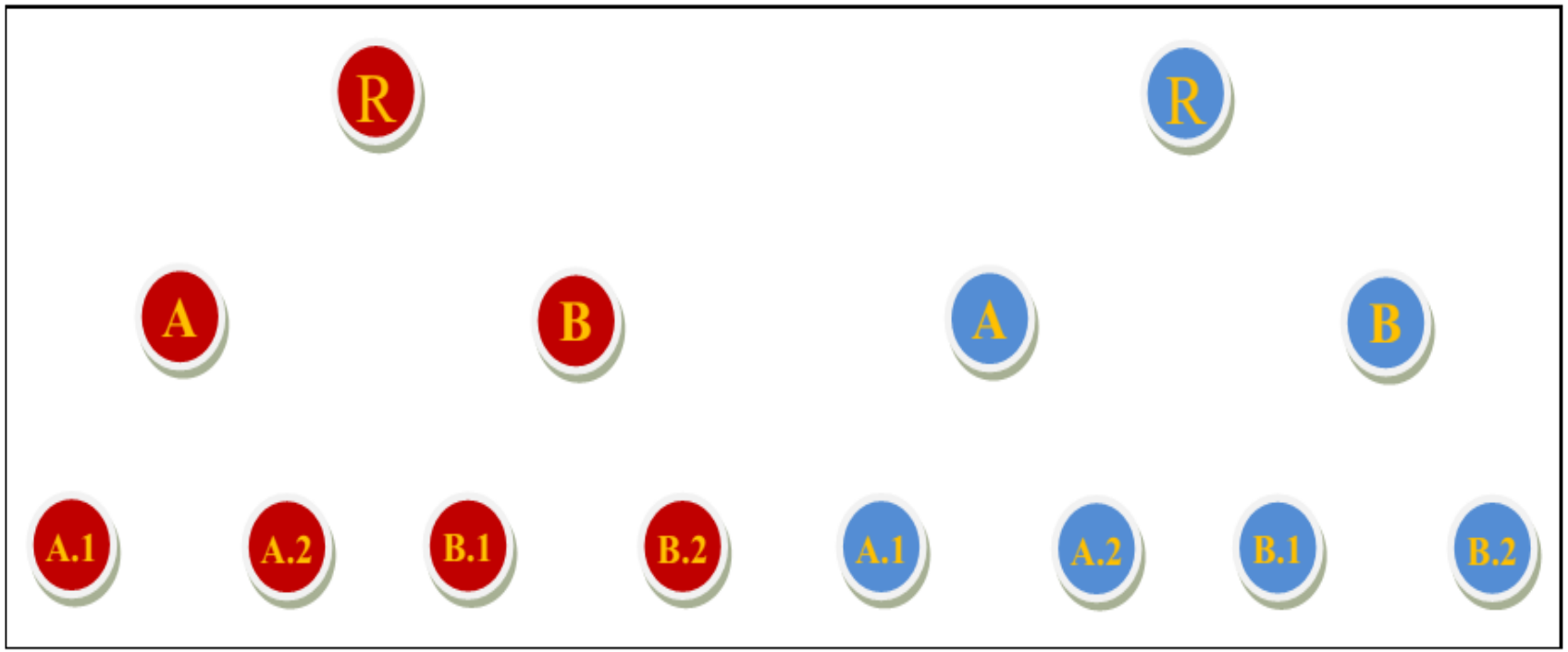
- Hierarchies are so common that sometimes multiple hierarchies classify similar data
- Heterogenous label view provide additional knowledge which should be exploited by learners
- Examples
  - **protein structure classification** - several hierarchical schemes for organizing proteins based on curation process or 3D structure
  - **web-page classification** - several hierarchy exist for categorizing such as DMOZ and wikipedia datasets

Objective

- Utilize multiple hierarchical label views in multi-task learning context to improve classification performance

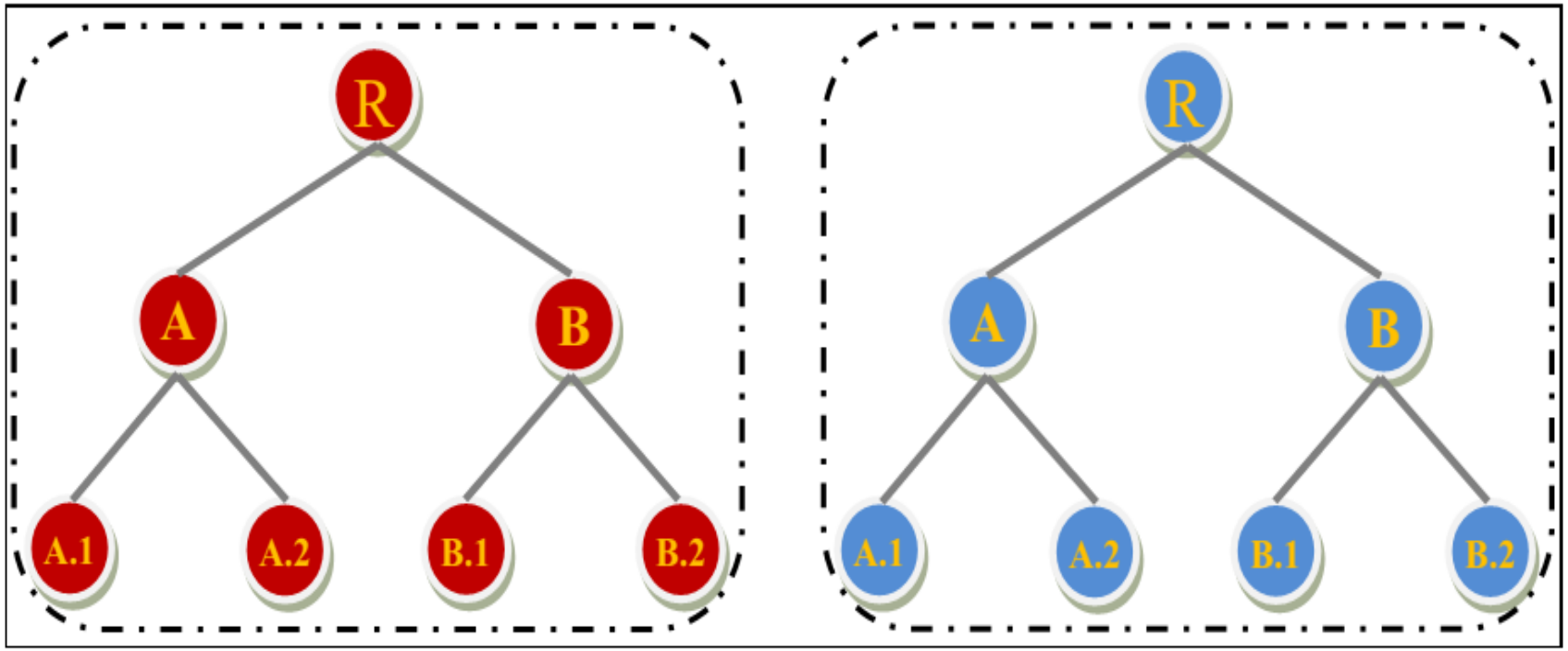
Three Different Learning Settings - I

(i) **Single Task Learning (STL)** - each task model parameters learned independently



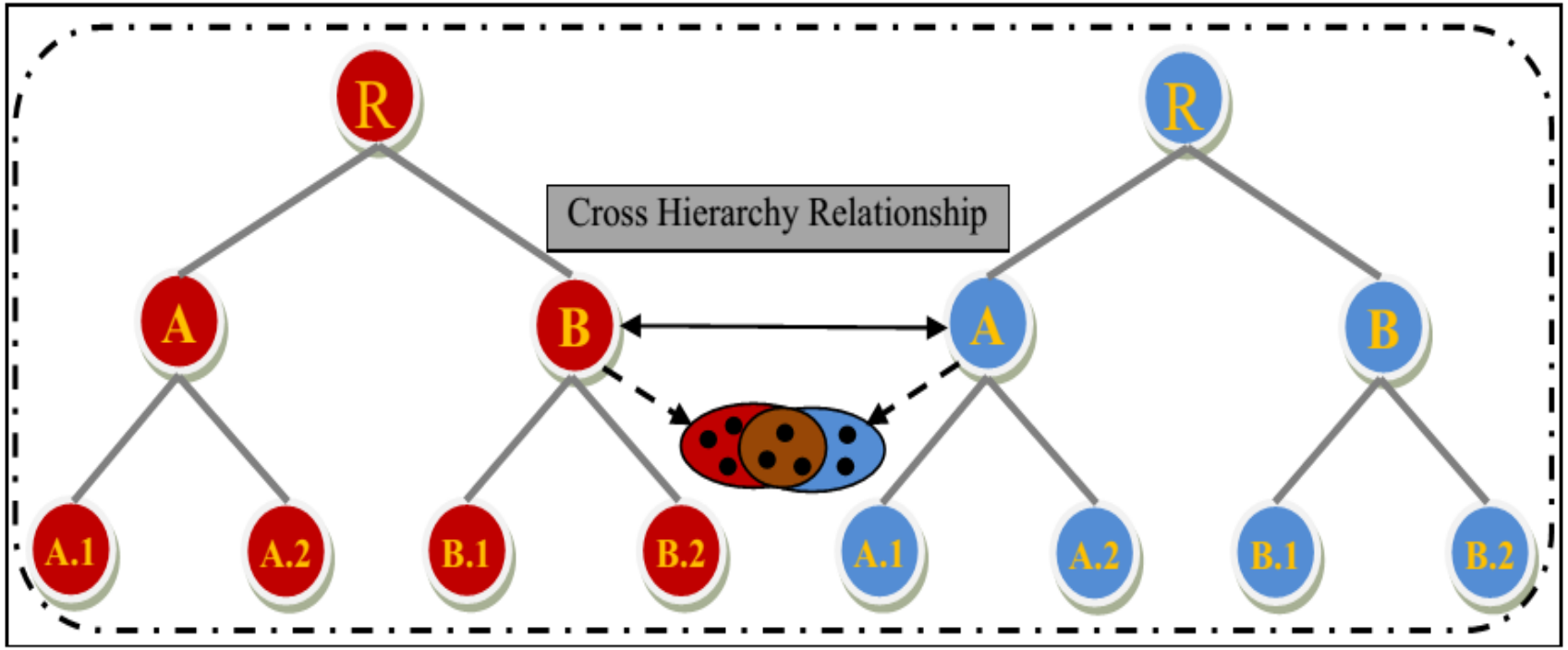
Three Different Learning Settings - II

(ii) **Single Hierarchy Multi-Task Learning (SHMTL)** - relationship between tasks within a hierarchy are combined individually



Three Different Learning Settings - III

(iii) **Multiple Hierarchy Multi-Task Learning (MHMTL)** - relationship between tasks from different hierarchies are extracted using common examples



MTL Formulations

- General MTL formulation:

$$\sum_{t=1}^{T}\sum_{i=1}^{n_t}\mathcal{L}(f(\mathbf{x}_i,\mathbf{W}_t),y_i)+\lambda\Omega\Big(\{\mathbf{W}_t\}_{t=1}^{T}\Big)$$

Empirical Loss **Regularization**

- Different MTL formulation based on regularization:
  - **Sparse** - All tasks share a single set of useful features

$$\Omega(\mathbf{W})=||\mathbf{W}||_{2,1}$$

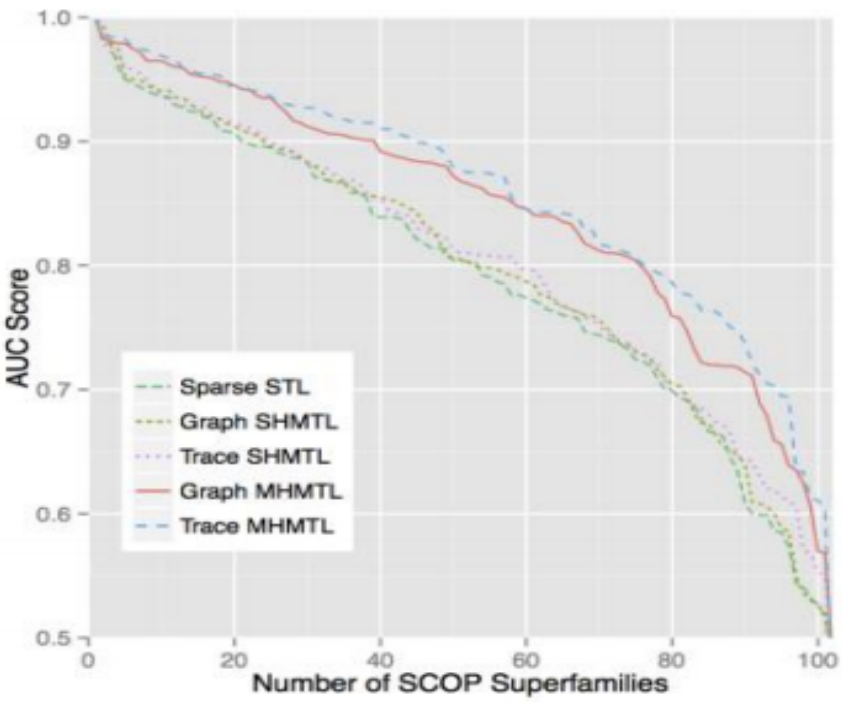
  - **Graph Regularization** - Related tasks have similar parameters

$$\Omega(\mathbf{W})=\sum_{(a,b)\in\mathcal{E}}||\mathbf{W}_a-\mathbf{W}_b||_2^2$$

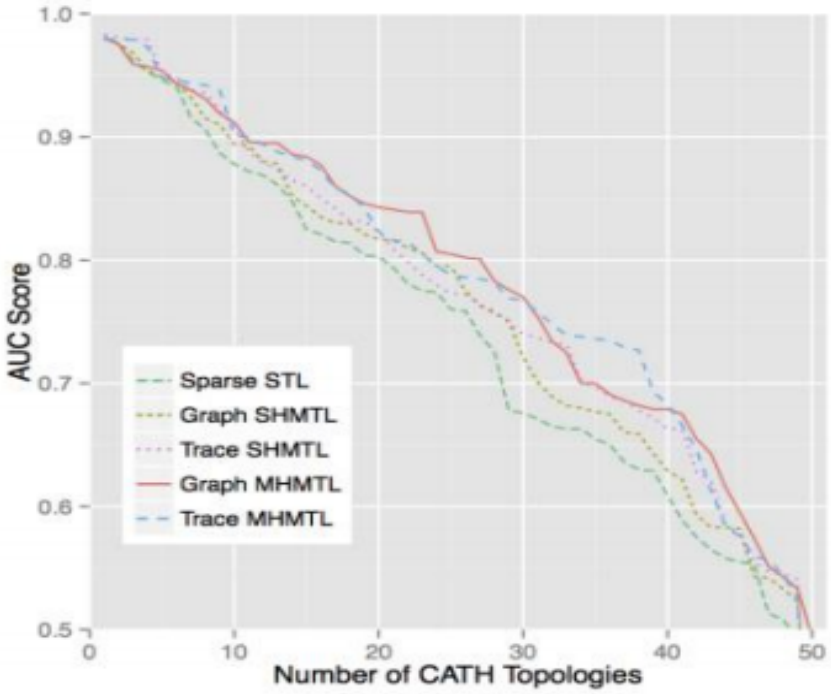
  - **Trace** - Task parameters are drawn from a low dimensional sub-space

$$\Omega(\mathbf{W})=||\mathbf{W}||_*=TraceNorm(\mathbf{W})$$

Performance: AUC Comparison

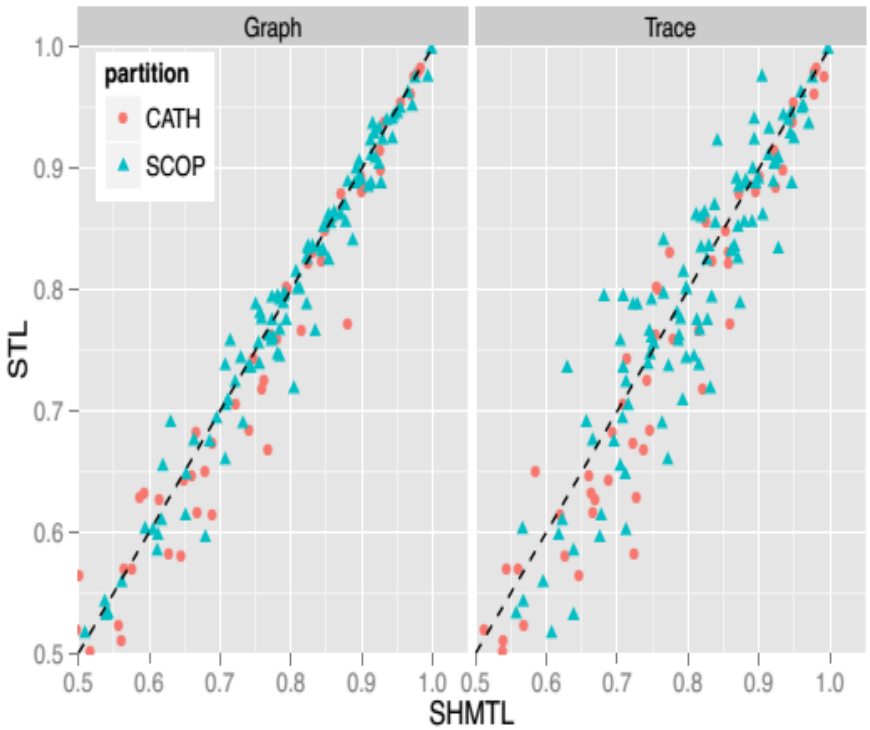


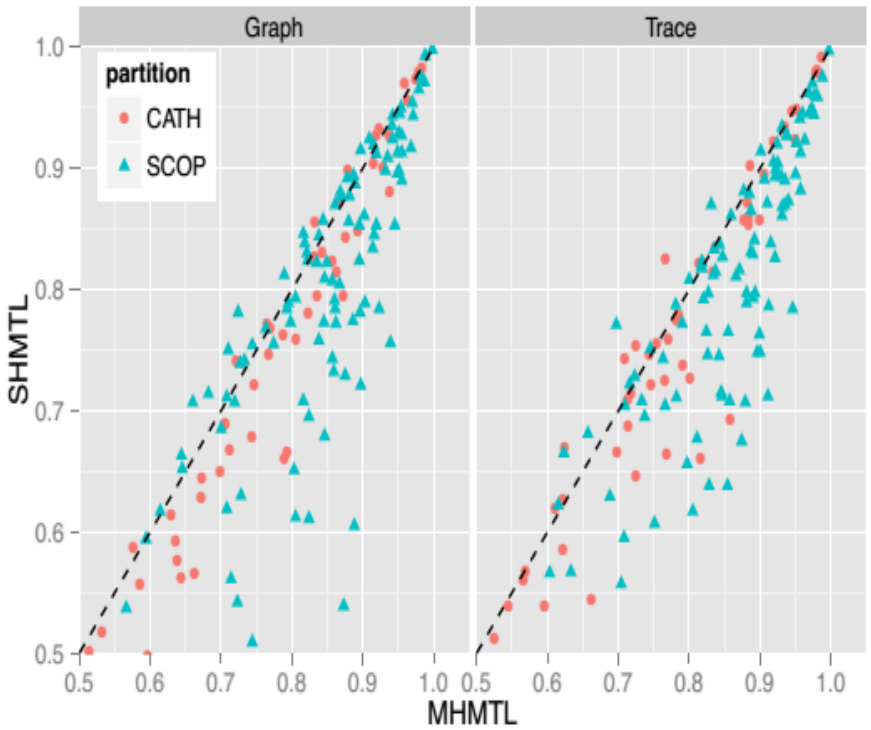
SCOP



CATH

STL, SHMTL and MHMTL Comparison





Extreme Classification

Motivation

- Many real world problems with multi-class and multi-label involving an extremely large number of labels or output space
- Learning classifiers corresponding to each labels is almost an impossible task
- Inter-label dependency not available
- Examples
  - Predict hashtags from tweets
  - LSHTC (Kaggle competition): Predict wikipedia tags from documents

Objective

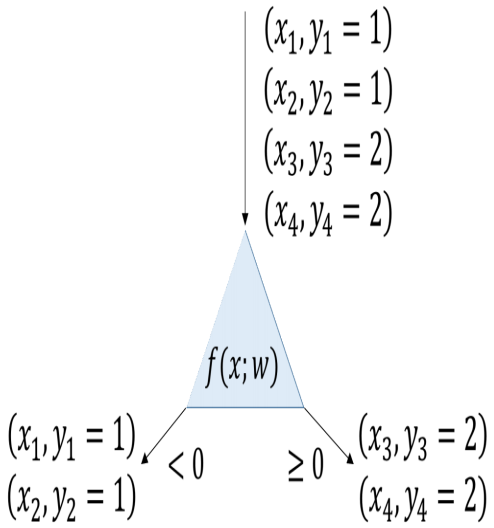
- Given huge set of labels, identify the labels that can be assigned to unlabeled instances (examples), **efficiently and accurately**

Extreme Classification Challenges

- **Statistical Challenges**
  - increase in number of classes is prone to decrease in accuracy due to complexity in discriminating between different classes
- **Computational Challenges**
  - training classifiers for large number of classes is computationally infeasible
  - predicting label for unlabeled test instances is also compute intensive task

Eigenpartition trees, Mineiro and Karampatziakis, NIPS workshop'15

- Compute small set of plausible labels using eigenpartition decomposition at each node in the tree



- At each node try to send each classs examples exclusively left or right
- While sending roughly the same number of examples left or right in aggregate
  - can be achieved through tree decomposition [**Choromanska and Langford, NIPS'15**]

Optimization function

$$\begin{aligned} \textit{maximize} \quad & w^T (X^T X) w \\ s.t. \quad & w^T w \leq 1 \\ & 1^T X^T w = 0 \end{aligned}$$

- Invoke expensive classifiers on set of plausible labels only

Experiments

Dataset description

- Largs-scale text dataset used for model evaluation

Dataset	Average Labels Per Example	Examples	Features	Classes
Twitter	1.27	25M	1M	264K
ALOI	1	97K	128	1K
ODP	1	1.5M	0.5M	100K
LSHTC	3.26	2.4M	1.6M	325K

Results - LSHTC dataset

- FastXML [**Prabhu and Varma, SIGKDD'14**] - node partitioning formulation which optimized an nDCG based ranking loss over all the labels
- X1 [**Bhatia et al., NIPS'15**] - effective number of labels is reduced by projecting the high dimensional label vectors onto a low dimensional linear subspace

Model	Precision at			Inference Speed (ex/s)
	1	3	5	
FastXML	49.35%	32.69%	24.03%	2000*
X1	55.57%	33.84%	24.07%	125*
Tree + ILR	53.0%	33.9%	24.8%	1058
$k = 6000$	53.7%	34.5%	25.3%	688
$k = 12000$	54.3%	34.9%	25.6%	370

Deep Classification, Xue et al., SIGIR'08

Motivation

- Large scale taxonomies are more prevelant due to the more specific topic related class information that is beneficial in several domains
- Examples
  - **web search browsing** - finding documents relevant to query
  - **modeling user's for personalized web search** - "java" means different to tourist and programmer
  - **advertisement matching** - finding related ads corresponding to web-page
- Traditional algorithms cannot be directly scaled to large scale problems due to several drawbacks
  - **large scale hierarchies**
  - **longer training time** and
  - incorporating **structural information** into learning framework

Motivation - II

Observations

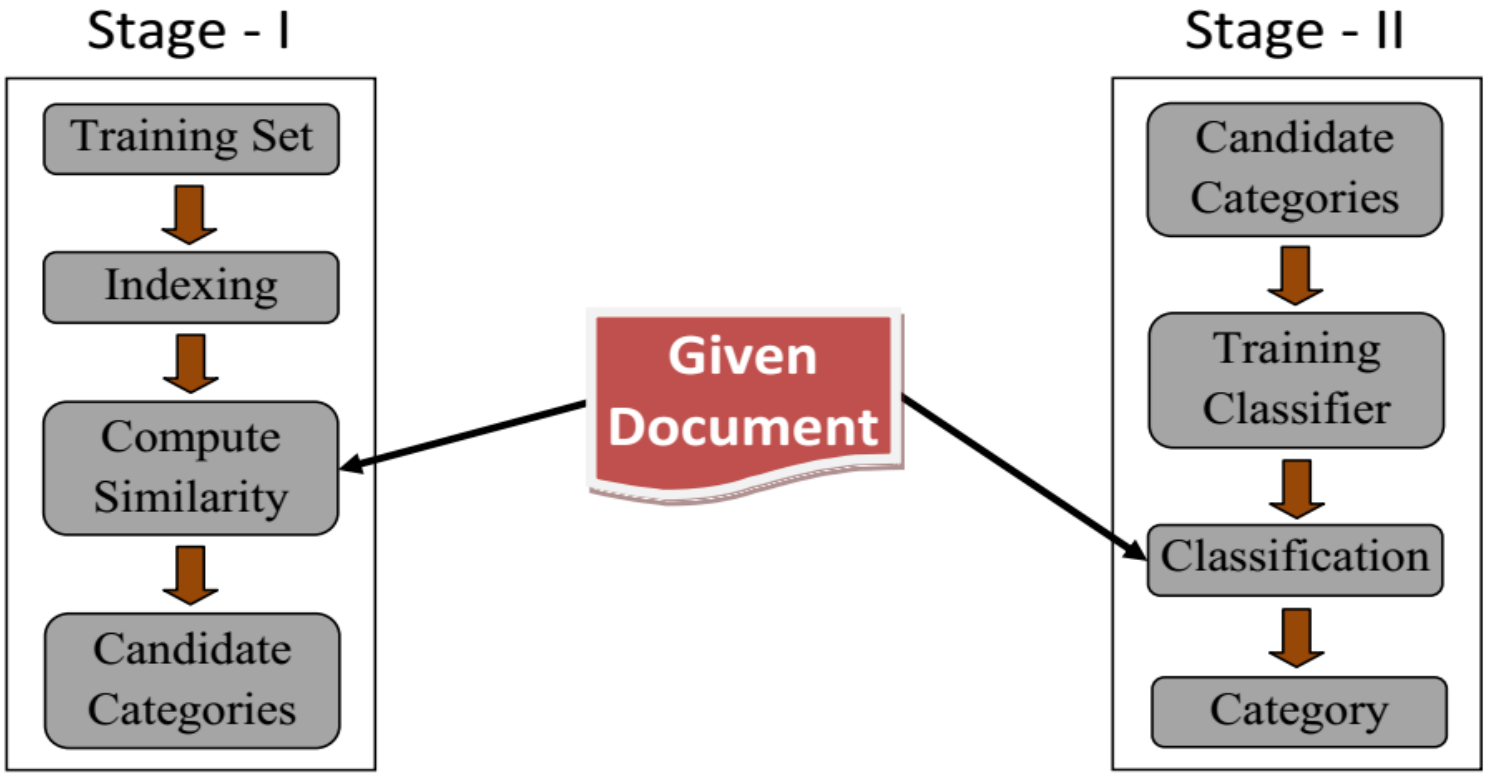
- Related categories corresponding to the query document are smaller than the number of unrelated categories
- Performance on smaller set of categories is easier and much better compared to the large set of categories

Objective

- Given large and deep hierarchies identify the relevant subset of categories for effectively finding the label of unlabeled test instances

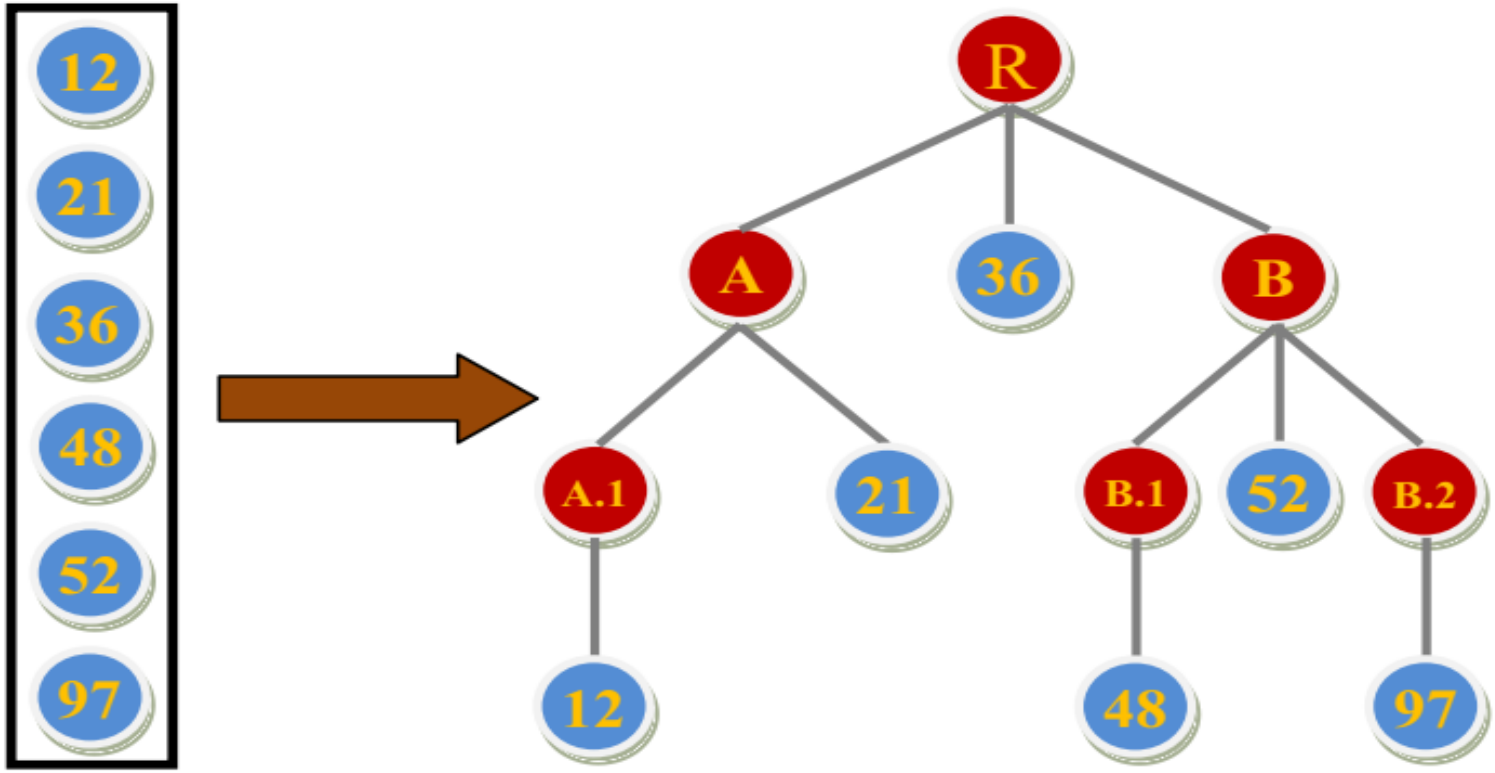
Two Stages for Classification

- First stage - **Search stage**
  - Identify related candidate categories corresponding to the test example
- Second stage - **Classification stage**
  - Select the best candidate categories using classification algorithm as the label for test document



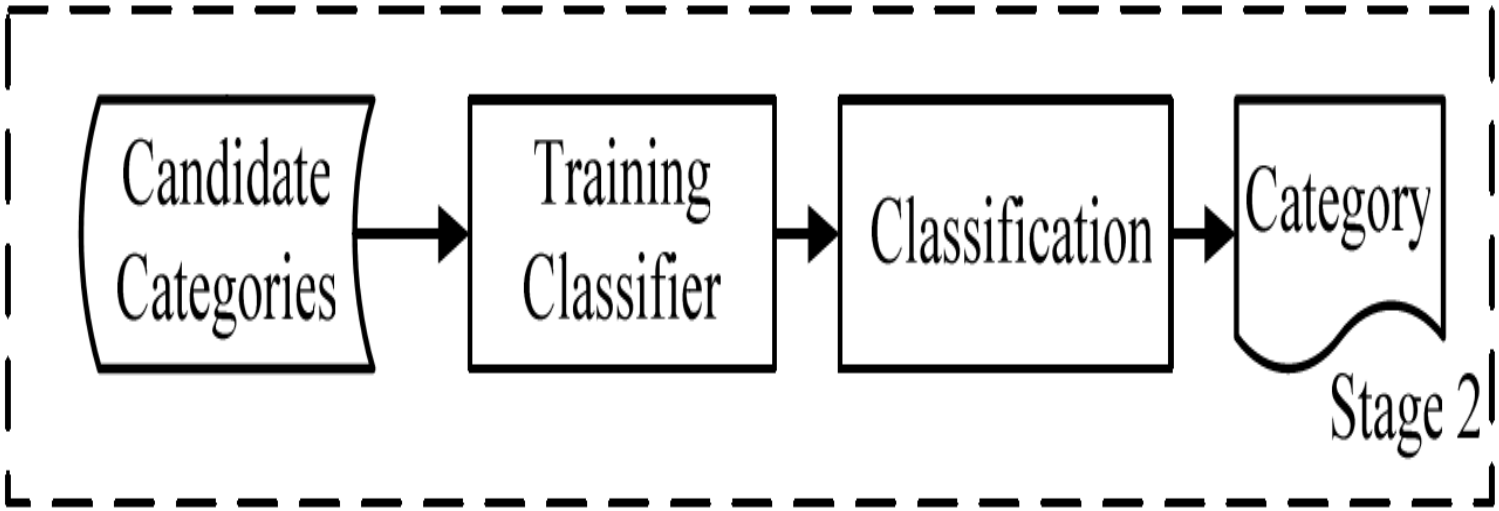
First Stage

- Large hierarchy is pruned into smaller subset of hierarchy with candidate categories and its ancestors only



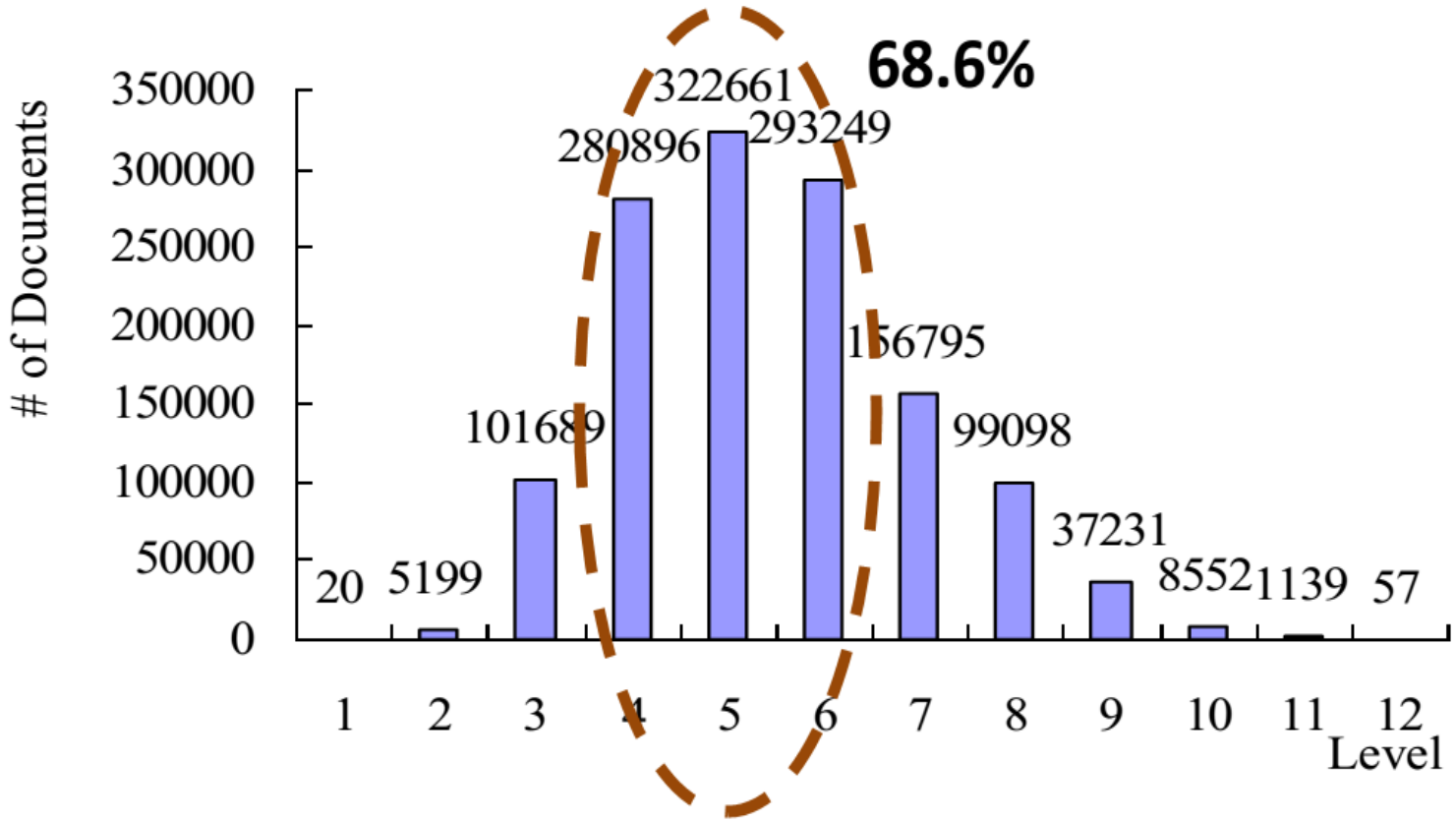
Second Stage

- Classifiers are trained on candidate categories
- Best category for test example is selected using trained classifiers



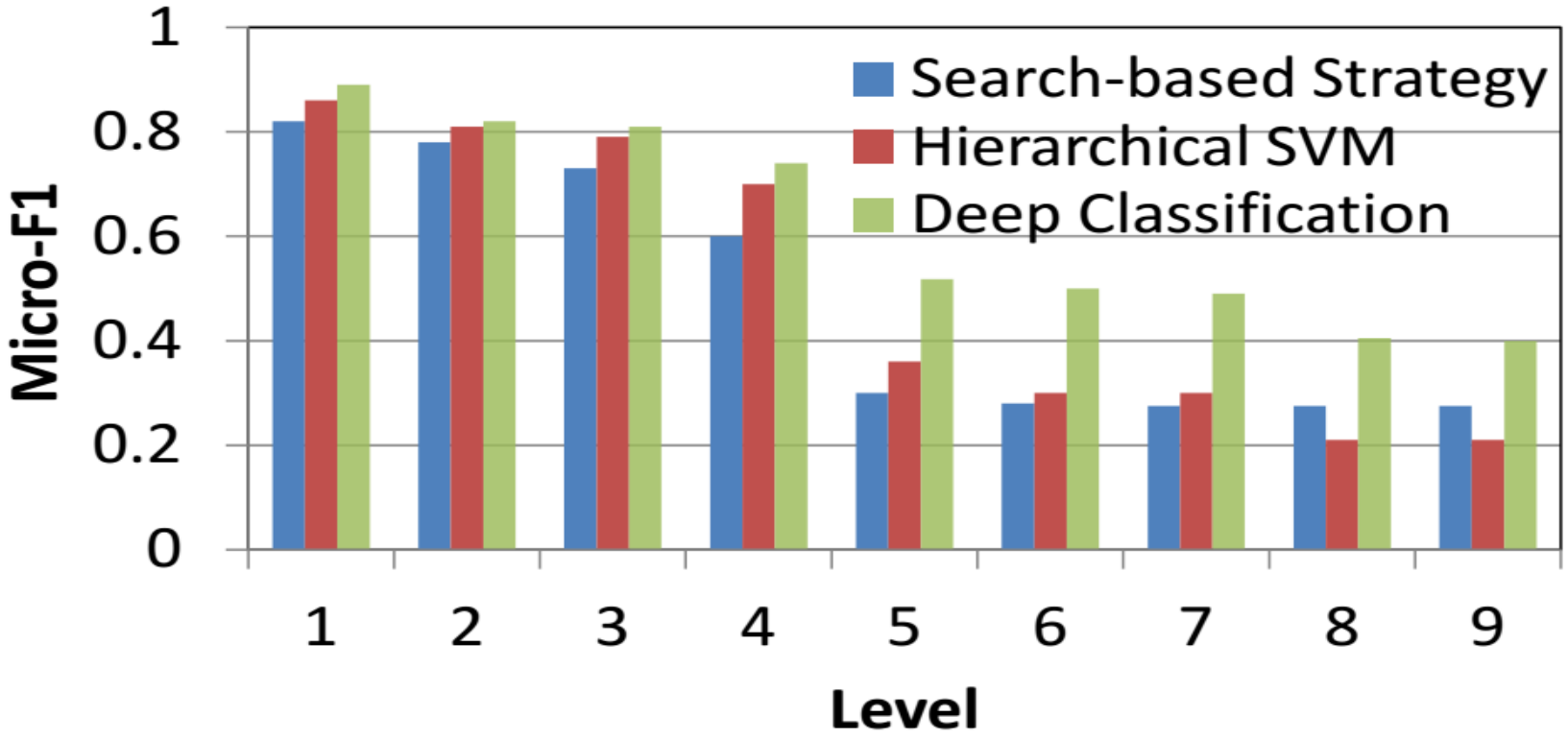
Dataset

- Open Directory Projects (ODP) dataset used for evaluation
- Dataset statistics
  - Trainining dataset - 1,174,586 web pages, 130,000 categories organized into 15 levels
  - Test dataset - 130,000 web pages



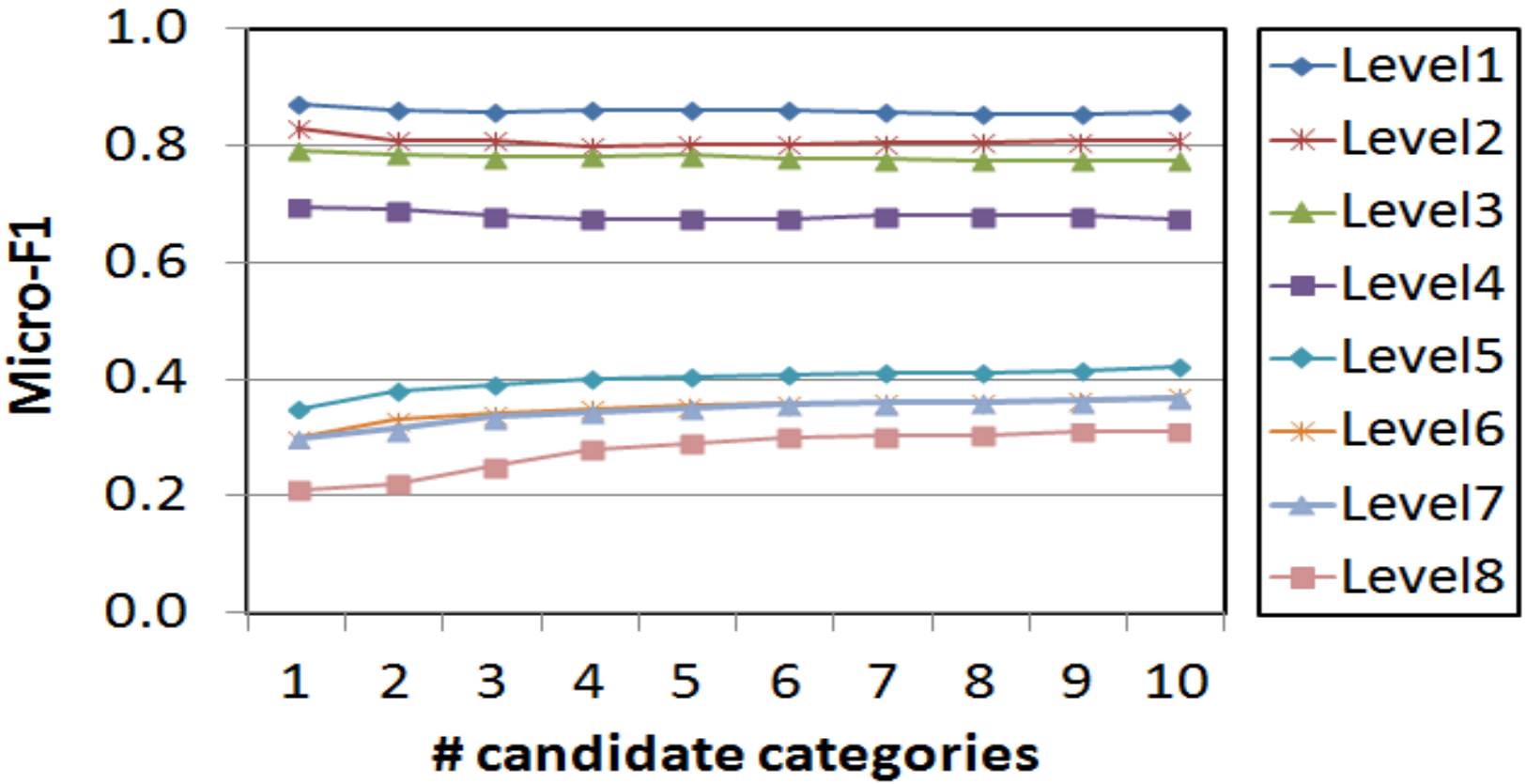
Results - Micro-F1 performance comparison

- **Search based Strategy** - best neighbor is chosen as the label for test example
- **Hierarchical SVM** [**Liu et al., SIGKDD'05**] - a pachinko machine style SVM
- **Deep Classification** - top ten neighbors are used as the candidate categories



Number of candidate categories selection

- As the number of candidate categories chosen by the search stage increases; chances for finding the correct label for test example in the classification stage increases
- Evaluation time increases with increasing number of category selection



Conclusion

- Large scale hierarchical classification is an important research problem in machine learning community due to its wide applicability across several domains
- Discussed various challenges associated with the hierarchical classification
- Discussed various state-of-the-art existing approaches; Demo of the software package developed by the author
- Emerging topics:
  - Large-scale classification with deep hierarchies
  - Orphan node prediction

References - I

- Gopal, Siddharth, and Yiming Yang. "Recursive regularization for large-scale classification with hierarchical and graphical dependencies." SIGKDD, 2013.
- Charuvaka, Anveshi, and Huzefa Rangwala. "HierCost: Improving Large Scale Hierarchical Classification with Cost Sensitive Learning." ECML, 2015.
- Tang, Lei, Jianping Zhang, and Huan Liu. "Acclimatizing taxonomic semantics for hierarchical content classification." SIGKDD, 2006.
- Li, Tao, Shenghuo Zhu, and Mitsunori Ogihara. "Hierarchical document classification using automatically generated hierarchy." Journal of Intelligent Information Systems 29.2 (2007): 211-230.
- Charuvaka, Anveshi, and Huzefa Rangwala. "Multi-task learning for classifying proteins using dual hierarchies." ICDM, 2012.
- Punera, Kunal, Suju Rajan, and Joydeep Ghosh. "Automatically learning document taxonomies for hierarchical classification." WWW, 2005.
- Qi, Xiaoguang, and Brian D. Davison. "Hierarchy evolution for improved classification." CIKM, 2011.
- Bennett, Paul N., and Nam Nguyen. "Refined experts: improving classification in large taxonomies." SIGIR, 2009.

References - II

- Silla Jr, Carlos N., and Alex A. Freitas. "A survey of hierarchical classification across different application domains." DMKD, 2011.
- Naik, Azad, A. Charuvaka, and H. Rangwala. "Classifying documents within multiple hierarchical datasets using multi-task learning." ICTAI, 2013.
- Babbar, Rohit, et al. "On flat versus hierarchical classification in large-scale taxonomies." NIPS, 2013.
- Wang, Xiao-Lin, and Bao-Liang Lu. "Flatten hierarchies for large-scale hierarchical text categorization." ICDIM, 2010.
- Chuang, Shui-Lung, and Lee-Feng Chien. "A practical web-based approach to generating topic hierarchy for text segments." CIKM, 2004.
- Fagni, Tiziano, and Fabrizio Sebastiani. "On the selection of negative examples for hierarchical text categorization." LTC, 2007.
- Clare, Amanda, and Ross D. King. "Predicting gene function in Saccharomyces cerevisiae." Bioinformatics, 2003.
- Koller, Daphne, and Mehran Sahami. "Hierarchically Classifying Documents Using Very Few Words." ICML, 1997.

References - III

- Xue et al. "Deep classification in large-scale text hierarchies." SIGIR, 2008.
- Tzanetakis, G., and P. Cook. "Musical genre classification of audio signals." IEEE transactions on Speech and Audio Processing, 2007.
- Gopal, Siddharth, et al. "Bayesian models for large-scale hierarchical classification." NIPS, 2012.
- Xiao, Lin, Dengyong Zhou, and Mingrui Wu. "Hierarchical classification via orthogonal transfer." ICML, 2011.
- Naik, Azad, and Huzefa Rangwala. "A ranking-based approach for hierarchical classification." DSAA, 2015.
- Tsochantaridis, Ioannis, et al. "Large margin methods for structured and interdependent output variables." JMLR, 2005.
- Liu, Tie-Yan, et al. "Support vector machines classification with a very large-scale taxonomy." SIGKDD, 2005.
- Caruana, Rich. "Multitask learning." Machine learning, 1997.
- Anveshi Charuvaka and Huzefa Rangwala. "Approximate block coordinate descent for large scale hierarchical classification." SAC, 2015.

References - IV

- Dumais, Susan, and Hao Chen. "Hierarchical classification of Web content." SIGIR, 2000.
- Mineiro, Paul, and Karampatziakis, Nikos. "A Hierarchical Spectral Method for Extreme Classification." eprint arXiv:1511.03260 (NIPS workshop), 2015.
- Choromanska, Anna, et al. "Extreme Multi Class Classification." NIPS Workshop: eXtreme Classification, 2013.
- McCallum, Andrew, et al. "Improving Text Classification by Shrinkage in a Hierarchy of Classes." ICML, 1998.
- Babbar, Rohit, et al. "Maximum-margin framework for training data synchronization in large-scale hierarchical classification." NIP, 2013.
- Choromanska, Anna E., and John Langford. "Logarithmic time online multiclass prediction." NIPS, 2015.
- Prabhu, Yashoteja, and Manik Varma. "FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning." SIGKDD, 2014.
- Bhatia, Kush, et al. "Sparse Local Embeddings for Extreme Multi-label Classification." NIPS, 2015.

References - V

- Naik, A., and Rangwala, H. "Filter based taxonomy modification for improving hierarchical classification." http://arxiv.org/abs/1603.00772, 2016.
- Peng, Hanchuan, Fuhui Long, and Chris Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy." PAMI, 2005.
- Ding, Chris, and Hanchuan Peng. "Minimum redundancy feature selection from microarray gene expression data." Journal of bioinformatics and computational biology, 2005.

Acknowledgement

Presenter:

Huzefa Rangwala

Azad Naik

Slides available for download at:

- **https://cs.gmu.edu/∼mlbio/sdm2016tutorial.html**

Thank You!