Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

大规模多标签分类方法

张睿智

导师：罗森林

2020年12月13日

内容提要

- 背景简介
- 基本概念
- 解决方法
- 算法原理
- 应用总结
- 参考文献

预期收获

- 预期收获
  - 1.了解大规模多标签分类基本思想
  - 2.了解大规模多标签分类的常用方法
  - 3.了解大规模多标签分类的应用

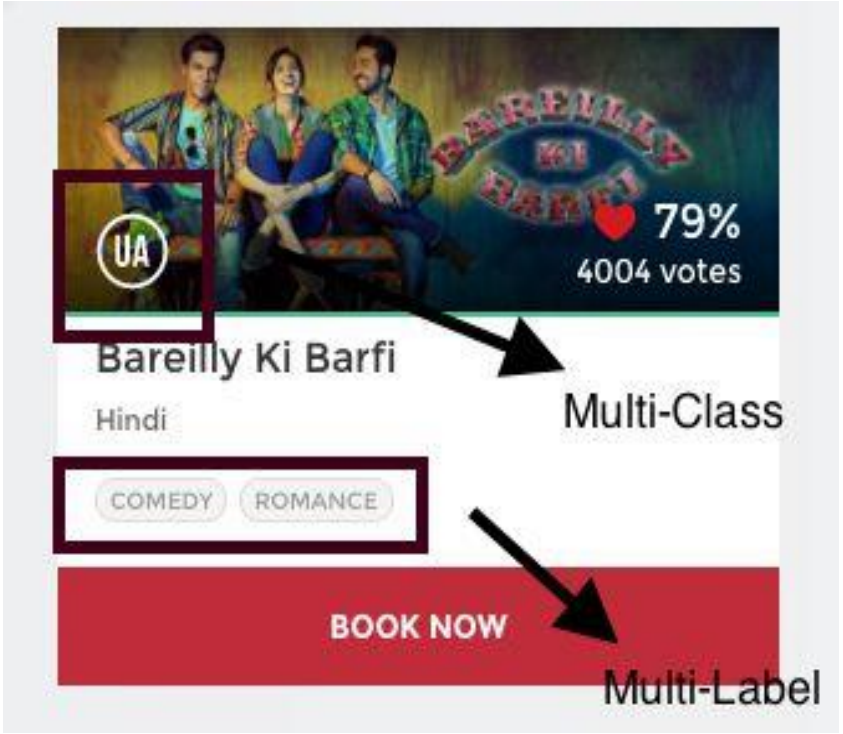
背景介绍

- 什么是多标签分类?
  - 问题一：下图中包含房子吗?
  - 问题二：与下图相关的东西（标签）有什么?

基本概念

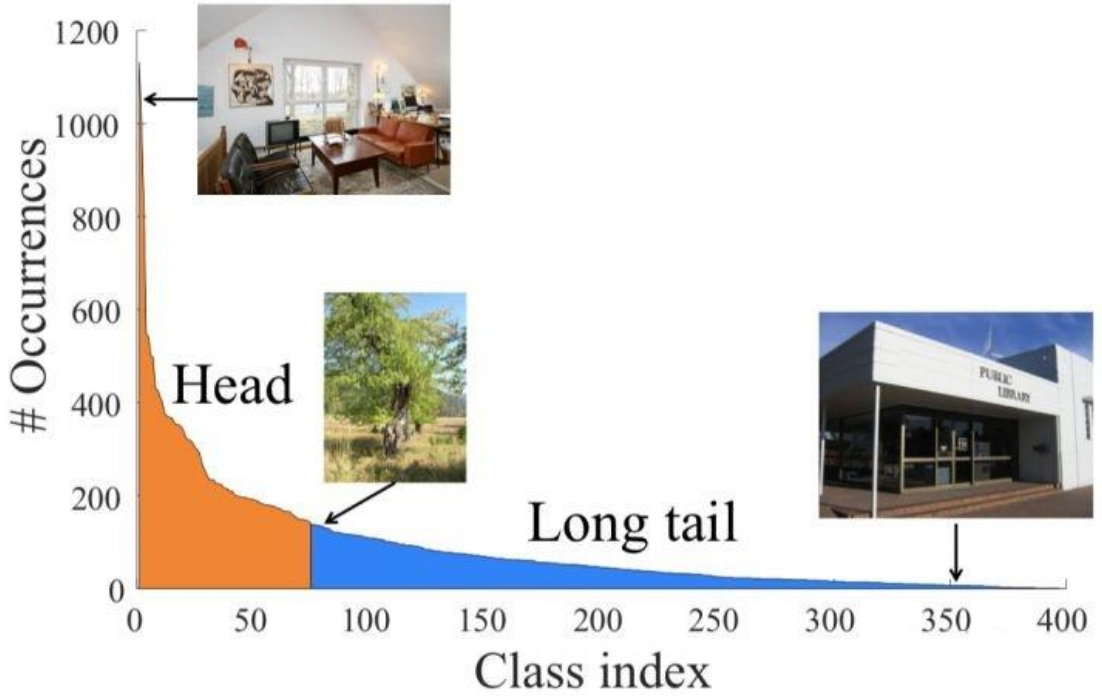
基本概念　多标签与多分类

- 多标签（Multi-Label）
  - 给每个样本一系列的目标标签，各个标签不互斥
  - 如一部电影可以同时被分为动作片和犯罪片，一则新闻可以同时属于政治和法律。
- 多分类（Multi-Class）
  - 也称多元分类，指在分类任务中有2个以上类别
  - 各标签互斥，每个样本有且仅有一个标签



基本概念　长尾标签

- Long Tail Distribution
  - 又叫 Power-law distribution（幂律分布）
  - 由于标签数目庞大，每个标签出现的频率呈长尾分布
  - 如 SUN-397 图片分类数据集中，“室内”、“树” 的样本较多，而一些标记，如 “公共图书馆” 则只有 1 个



基本概念　低秩矩阵

- Low-rank
  - 矩阵的秩
    - 矩阵中极大无关组的向量个数
  - 图像中的秩
    - 图像中包含信息的丰富程度
  - 低秩标签矩阵
    - 标签空间有大量相似标签
- 低秩标签假设
  - 可以对标签空间进行压缩
  - 丢失长尾分布信息

- P@k
  - 前 k 个结果的准确度, P 指Precision

$$\mathrm{P@}k := \frac{1}{k} \sum_{l \in \mathrm{rank}_k(\hat{\mathbf{y}})} \mathbf{y}_l$$

- CG（Cumulative Gain）
  - 累计效益，所有文档都有一个对应的相关度 ，相关度越高表示文章与查询越相关
  - 假设一共检索得到 T 个文档，CG 计算的就是这 T 个文档的相关度之和

- DCG@k（Discounted Cumulative Gain）
  - 将所有的结果累加起来，给排在后面的结果加上折扣系数
  - 排序位置越靠后，折扣系数越小
  - 不同查询返回的文档个数一般不同，无法直接使用 DCG 进行比较

$$\mathrm{DCG@}k := \sum_{l \in \mathrm{rank}_k(\hat{\mathbf{y}})} \frac{\mathbf{y}_l}{\log(l+1)}$$

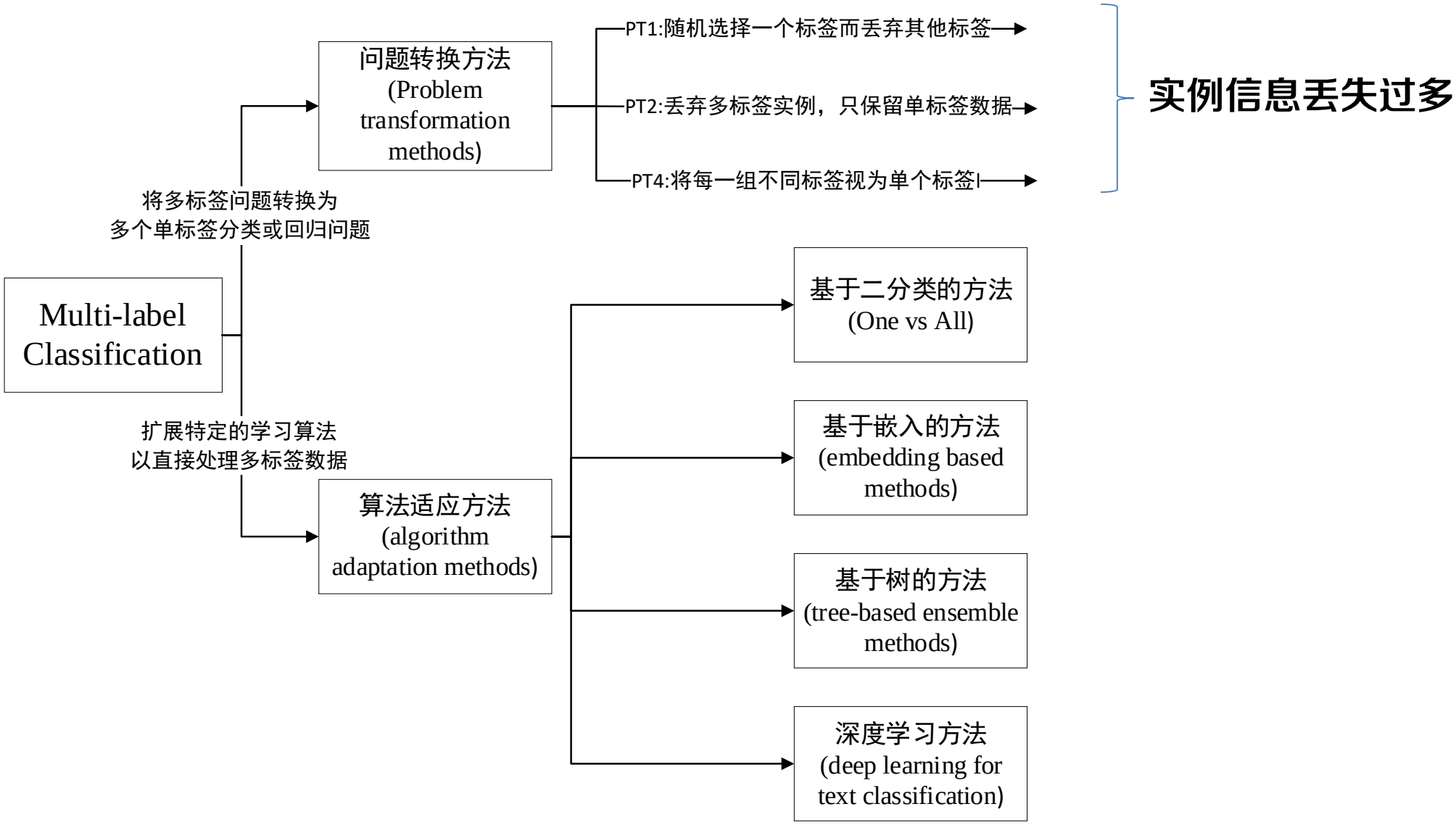
- nDCG@k
  - 归一化后的 DCG 值，可用于多个查询的情况

$$\mathrm{nDCG@}k := \frac{\mathrm{DCG@}k}{\sum_{l=1}^{\min(k, \|\mathbf{y}\|_0)} \frac{1}{\log(l+1)}}$$

大规模多标签分类方法

解决方法

解决方法　思维导图


Multi-label Classification
将多标签问题转换为
多个单标签分类或回归问题
问题转换方法
(Problem transformation methods)
PT1:随机选择一个标签而丢弃其他标签
PT2:丢弃多标签实例，只保留单标签数据
PT4:将每一组不同标签视为单个标签l
实例信息丢失过多
扩展特定的学习算法
以直接处理多标签数据
算法适应方法
(algorithm adaptation methods)
基于二分类的方法
(One vs All)
基于嵌入的方法
(embedding based methods)
基于树的方法
(tree-based ensemble methods)
深度学习方法
(deep learning for text classification)

解决方法　问题转换

- 问题转换方法
  - 为每个样本随机选择一个标签，丢弃其他标签

Ex.	Sports	Religion	Science	Politics
1	X			
2				X
3	X			
4		X		

  - 丢弃多标签实例，只保留单标签实例数据

Ex.	Sports	Religion	Science	Politics
3	X			

  - 将每组不同标签视为单个标签

Ex.	Sports	(Sports ∧ Politics)	(Science ∧ Politics)	(Science ∧ Religion)
1		X		
2			X	
3	X			
4				X

Ex.	Sports	Religion	Science	Politics
1	X			X
2			X	X
3	X			
4		X	X	

原始数据样例

解决方法　算法适应

- 算法适应方法
  - One-vs-All方法
    - 根据每个标签学习一个分类器，从而与其他标签区分开
    - 预测精度高，但训练成本高、模型规模大、可扩展性差
    - PD-Sparse、PPDSparse、Dismec
  - 基于嵌入的方法
    - 使用压缩后的标签进行训练，最后对压缩后的标签解压，进行预测
    - 压缩解压过程中导致信息丢失，幂律分布导致低秩标签空间假设被打破
    - SLEEC、AnneXML

解决方法　算法适应

- 算法适应方法
  - 基于树的方法
    - 根据特征递归划分给定实例或标签空间来生成树，从而在每个叶子上产生一个简单的分类器
    - 树级联中存在错误传播问题
    - FastXML、PLT、PfastreXML、Parabel、Bonsai
  - 基于深度学习方法
    - 由于正样本数量少，数据稀疏，因此对长尾标签问题处理效果不好
    - CNN-Kim、XML-CNN、FastText

算法原理

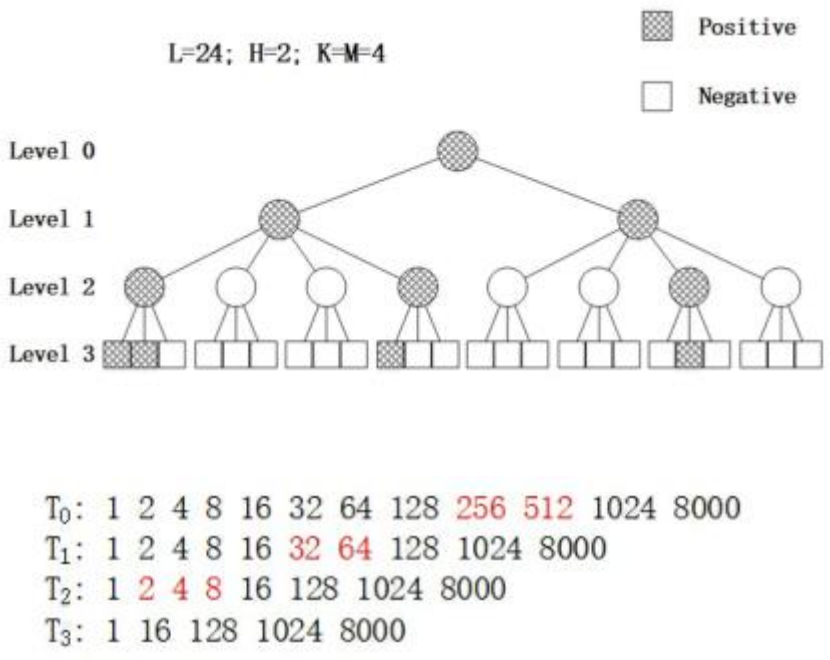
算法原理

T	构建准确率高且计算开销小的大规模多标签分类模型
I	原始标记化文本
P	1.构建浅而宽的标签树（PLT） 2.对PLT的每一级，使用BiLSTM和多标签注意力机制训练一个注意力感知深度模型
O	带标签的文本信息

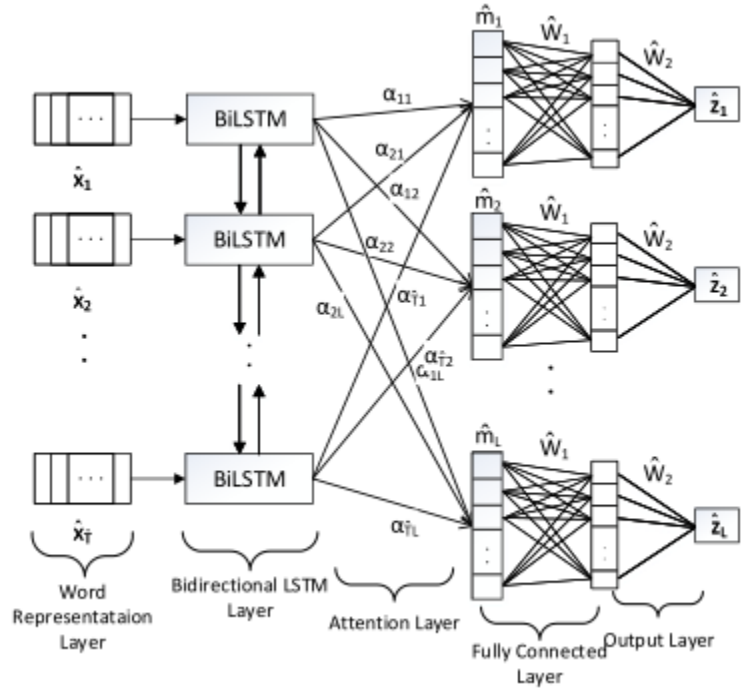
P	无法捕捉标签潜在含义，尾部标签分类性能差
C	标签实例及信息稀疏情况下的多标签分类
D	尾部标签数据稀疏
L	CORR2019

算法原理

- AttentionXML算法原理图



PLT building process



Attention-aware deep model

算法原理　标签树

- 标签树（PLT）
  - PLT是一颗叶节点是L的树，每个叶子对应着原来的一个标签
  - 标签树构建过程
    - 使用标签下文本的BOW特征求和获得该标签的特征
    - 通过k-means（k=2）将标签递归划分为两个更小的簇，直到每个节点下标签的数量小于M

- 标签树（PLT）
  - 标签树压缩过程
    - 选择第h次的祖宗节点或者根节点作为 S_h
    - 移除 S_{h-1} 和 S_h之间的节点
    - 重置 S_h 为相应 S_{h-1} 中的父节点

Algorithm 1 Compressing into a shallow and wide PLT

Input: (a) Labels of training data $\{y_i\}_{i=1}^{N_{train}}$; (b) PLT T_0; (c) $K = 2^c$, H
Output: A compressed shallow and wide PLT T

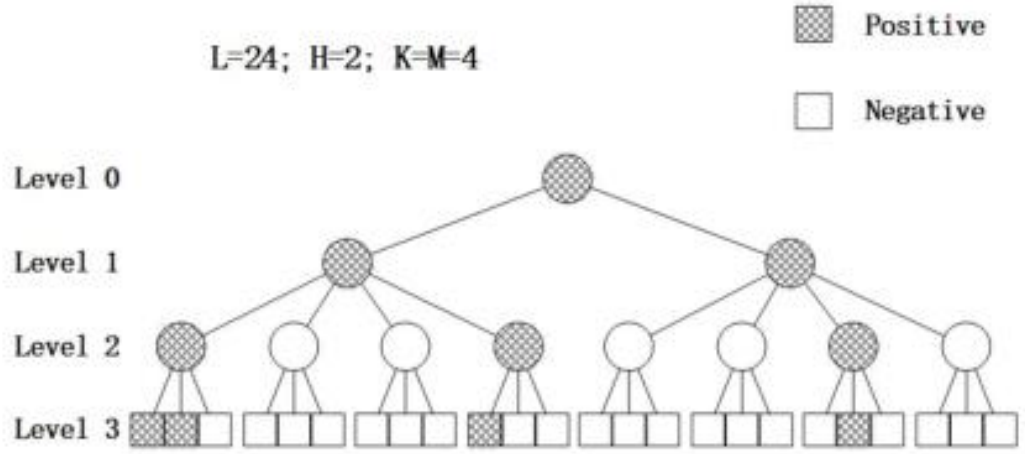
$S_0 \leftarrow$ {parent nodes of leaves}
for $h \leftarrow 1$ **to** H **do**
  if h < H **then**
    $S_h \leftarrow \{c$-th ancestor node n of nodes in $S_{h-1}\}$
  else
    $S_h \leftarrow$ { the root of T_0}
  $T_h \leftarrow T_{h-1}$
  for all nodes $n \in S_h$ **do**
    for all nodes $n' \in S_{h-1}$ and node n is the ancestor of n' in T **do**
      $Pa(n') \leftarrow n$ ▷ Let node n be parent of node n' in T_h, $Pa(n)$ means parent of n.
return T_H

T_0: 1 2 4 8 16 32 64 128 256 512 1024 8000
T_1: 1 2 4 8 16 32 64 128 1024 8000
T_2: 1 2 4 8 16 128 1024 8000
T_3: 1 16 128 1024 8000

$$K = M = 2^c = 8 \quad c = 3$$

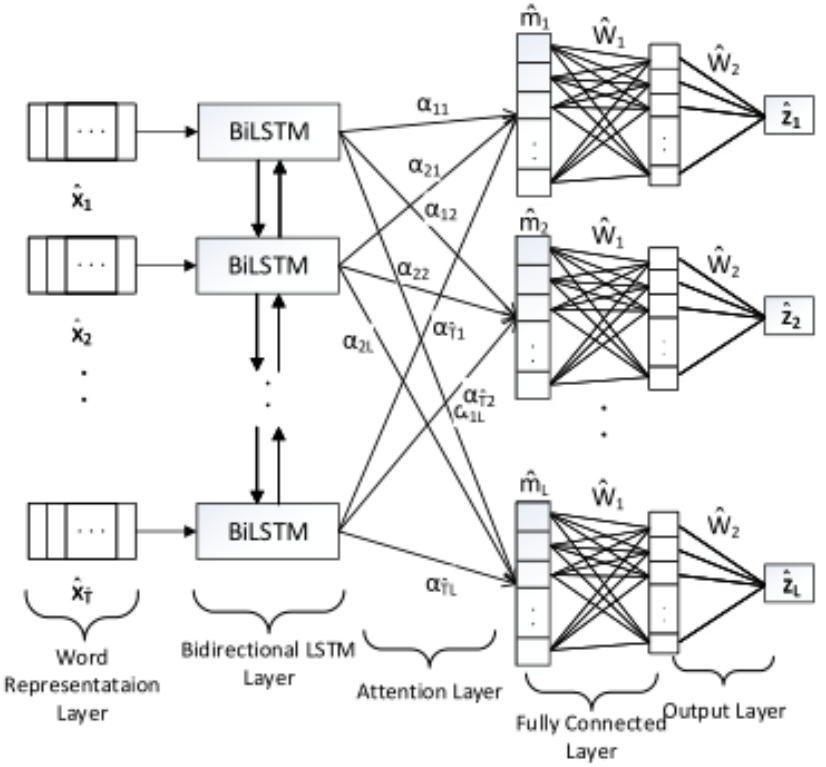
算法原理

- 分层训练AttentionXML
  - PLT中层次越深，正例越少，模型训练越困难
  - 自顶向下地为给定PLT的每级单独训练一个深度模型
  - 对于第一级节点(根的子节点)，直接训练AttentionXML($AttentionXML_1$)
  - 将$AttentionXML_{d-1}$预测的第$(d-1)$层标签的得分由高到底进行排序，选择第$(d-1)$层的top C标签作为$AttentionXML_d$训练的候选标签



算法原理

- 注意力感知深度模型
  - 单词表示层
    - 使用预训练的300维GloVe词嵌入作为初始词表示
  - BiLSTM层
    - 捕获左右两侧的上下文
    - 通过连接正向输出和反向输出获得h
  - 多标签注意层
    - 为每个标签计算上下文向量，捕获与之最相关的文本
    - 输出为 $\hat{\mathbf{m}}_j = \sum_{i=1}^{\hat{T}} \alpha_{ij} \hat{\mathbf{h}}_i$ $\alpha_{ij} = \frac{e^{\hat{\mathbf{h}}_i \hat{\mathbf{w}}_j}}{\sum_{t=1}^{\hat{T}} e^{\hat{\mathbf{h}}_t \hat{\mathbf{w}}_j}}$
  - 全连接层
    - 所有标签使用相同的参数值，强调标签之间的attention差异
  - 输出层



实验结果　数据集

- 数据集
  - 3个 large-scale 数据集（标签数范围为 4K – 30K）
    - EUR-Lex1，Wiki10-31K2，AmazonCat-13K
  - 3个extreme-scale 数据集（标签数范围为 500K – 3M）
    - Amazon-670K2，Wiki-500K2，Amazon-3M

Dataset	N_{train}	N_{test}	D	L	$\overline{L}$	$\hat{L}$	$\overline{W}_{train}$	$\overline{W}_{test}$
EUR-Lex	15,449	3,865	186,104	3,956	5.30	20.79	1248.58	1230.40
Wiki10-31K	14,146	6,616	101,938	30,938	18.64	8.52	2484.30	2425.45
AmazonCat-13K	1,186,239	306,782	203,882	13,330	5.04	448.57	246.61	245.98
Amazon-670K	490,449	153,025	135,909	670,091	5.45	3.99	247.33	241.22
Wiki-500K	1,779,881	769,421	2,381,304	501,008	4.75	16.86	808.66	808.56
Amazon-3M	1,717,899	742,507	337,067	2,812,281	36.04	22.02	104.08	104.18

N_{train}: #training instances, N_{test}: #test instances, D: #features, L: #labels, $\overline{L}$: average #labels per instance, $\hat{L}$: the average #instances per label, $\overline{W}_{train}$: the average #words per training instance and $\overline{W}_{test}$: the average #words per test instance. The partition of training and test is from the data source.

实验结果

- 对比方法
  - AnnexML，DiSMEC，PfastreXML，Parabel，Bonsai，XML-CNN
- 对比实验
  - 长尾标签上的性能
  - 标签树数量的影响
  - 标签树高度的影响
- 评价指标
  - P@k（k取1，3，5）
  - PSP@k（k取1，3，5）

实验结果

- 实验结论
  - 使用了浅而宽的标签树及注意力机制后，准确率有所提高

Table 3: Performance comparisons of AttentionXML and other competing methods over six benchmarks. The results with the stars are from **Extreme Classification Repository** directly.

Methods	P@1=N@1	P@3	P@5	Methods	P@1=N@1	P@3	P@5
EUR-Lex				Amazon-670K			
AnnexML	79.66	64.94	53.52	AnnexML	42.09	36.61	32.75
DiSMEC	83.21	70.39	58.73	DiSMEC	44.78	39.72	36.17
PfastreXML	73.14	60.16	50.54	PfastreXML*	36.84	34.23	32.09
Parabel	82.12	68.91	57.89	Parabel	44.91	39.77	35.98
XT	79.17	66.80	56.09	XT	42.54	37.93	34.63
Bonsai	82.30	69.55	58.35	Bonsai	45.58	40.39	36.60
MLC2Seq	62.77	59.06	51.32	MCL2Seq	-	-	-
XML-CNN	75.32	60.14	49.21	XML-CNN	33.41	30.00	27.42
AttentionXML-1	*85.49*	*73.08*	*61.10*	AttentionXML-1	*45.66*	*40.67*	*36.94*
AttentionXML	**87.12**	**73.99**	**61.92**	AttentionXML	**47.58**	**42.61**	**38.92**
Wiki10-31K				Wiki-500K			
AnnexML	86.46	74.28	64.20	AnnexML	64.22	43.15	32.79
DiSMEC	84.13	74.72	65.94	DiSMEC	70.21	50.57	39.68
PfastreXML*	83.57	68.61	59.10	PfastreXML	56.25	37.32	28.16
Parabel	84.19	72.46	63.37	Parabel	68.70	49.57	38.64
XT	83.66	73.28	64.51	XT	65.17	46.32	36.15
Bonsai	84.52	73.76	64.69	Bonsai	69.26	49.80	38.83
MLC2Seq	80.79	58.59	54.66	MCL2Seq	-	-	-
XML-CNN	81.41	66.23	56.11	XML-CNN	-	-	-
AttentionXML-1	*87.05*	*77.78*	*68.78*	AttentionXML-1	*75.07*	*56.49*	*44.41*
AttentionXML	**87.47**	**78.48**	**69.37**	AttentionXML	**76.95**	**58.42**	**46.14**

实验结果

- 长尾标签上的性能
  - $PSP@k$为k上的倾向性得分精度
  - 浅而宽的标签树和多标签注意力机制都显著提高了长尾标签的预测性能

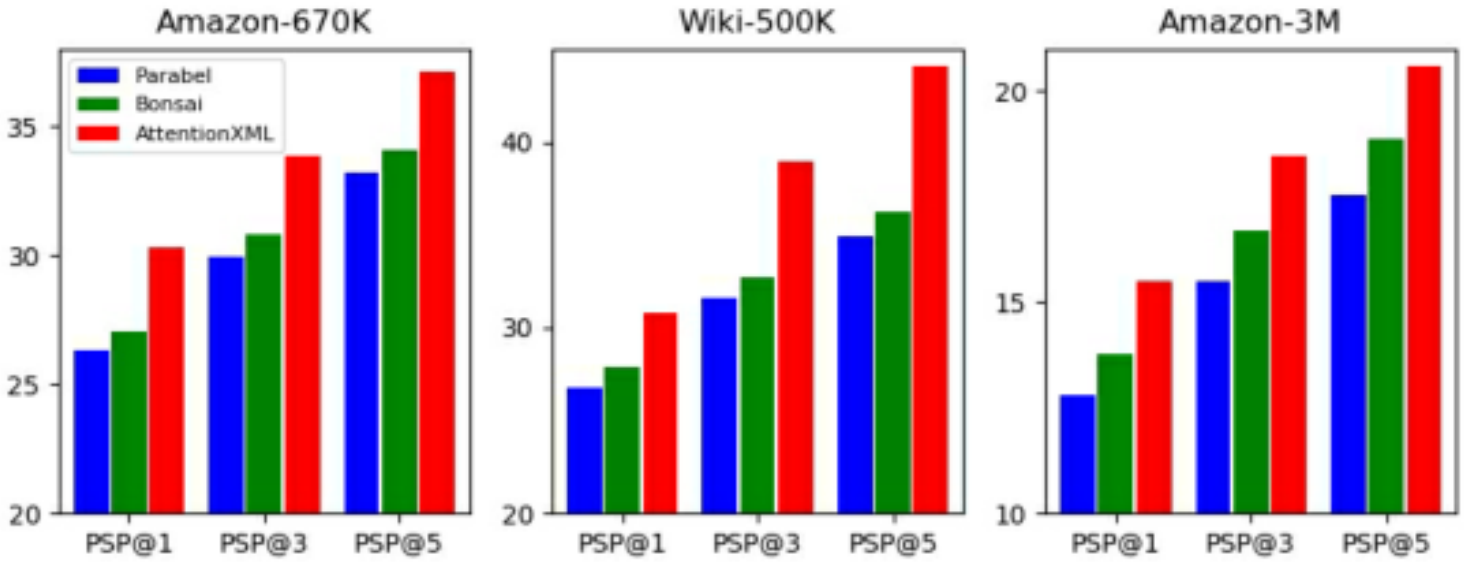


Figure 2: $PSP@k$ of label tree-based methods.

实验结果

- 标签树数量对性能的影响
  - 树的增多能够提高预测精度，但需要更多的时间来进行训练和预测

Table 5: Performance of variant number of trees in AttentionXML.

	Amazon-670K			Wiki-500K			Amazon-3M		
Trees	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
1	45.66	40.67	36.94	75.07	56.49	44.41	49.08	46.04	43.88
2	46.86	41.95	38.27	76.44	57.92	45.68	50.34	47.45	45.26
3	47.58	42.61	38.92	76.95	58.42	46.14	50.86	48.04	45.83
4	**48.03**	**43.05**	**39.32**	**77.21**	**58.72**	**46.40**	**51.66**	**48.39**	**46.23**

- 标签树高度对性能的影响
  - H越小，性能越好，但所需时间成本更大

Table 8: Performance comparisons of different H for AttentionXML on Amazon-670K.

Methods	P@1=N@1	P@3	P@5	N@3	N@5	PSP@1	PSP@3	PSP@5
Amazon-670K, $K = 8, C = 160$								
$H = 2$	**45.74**	**40.92**	**37.12**	**43.26**	**41.53**	**29.32**	**32.50**	**35.18**
$H = 3$	45.66	40.67	36.94	43.04	41.35	29.30	32.36	35.12
$H = 4$	45.29	40.47	36.73	42.83	41.13	28.88	32.08	34.79

应用总结

应用总结

- 算法的应用领域
  - 音频分类
  - 图像分类
  - 生物信息
    - 医学诊断
    - 基因功能
  - 文本分类
    - 新闻标注
    - 网页标签
    - 商品分类
  - …

参考文献

[1] You, R., et al., AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. 2019.
[2] Y . Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. V arma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, pages 993 - 1002. International World Wide Web Conferences Steering Committee, 2018

道德经

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！