# An Enhanced Docker Container Image Tagging Approach for Docker Hub

## 1. 引论

目前，容器已发展成为当代软件的主要载体之一。Docker作为目前容器技术的标准，为应用程序提供了进程级别的虚拟化能力。通过Docker，用户可以打包自己的应用，生成独立完整的Docker Image。随着Docker的广泛使用，DockerHub作为一个容器仓库发布了大量的容器镜像，并且数量急剧增加。用户可以通过DockerHub依靠镜像提供者给出的能够反映镜像功能、操作系统和架构的标签，简单地搜索所需的容器镜像。

现阶段，在DockerHub等社区进行容器镜像的搜索时，通常是直接使用镜像名称或标签关键词进行搜索，导致在关键词模糊的情况下出现众多的待选项，需要再次进行大量的人为筛选，而这体现出其自带标签在搜索过程中所起到的筛选作用是非常有限的，这是由于许多标签所描述的信息常常与容器镜像的功能和内容没有严密的联系，呈现出过于抽象和宏观的特点，使得用户无法精准地获取想要的结果。因而，如果具有更精确的容器镜像标签，对于用户而言，能够有效地提升结果搜索的效率，同时基于此开展容器镜像的相关推荐，提升软件的发现和展开效率，而对于容器管理平台的维护人员而言，精确的标签是对容器镜像进行区分管理的有效的支持手段。

为了得到精确描述容器镜像的标签，最关键的问题在于如何挖掘出容器镜像的特点进而生成标签而不是仅仅依靠于镜像提供者提供的标签。2018年，Tianyin Xu和Darko Marinov在ICSE-NIER通过挖掘容器仓库获得容器配置和管理的信息[1]。自2018年以来，一些与容器安全、进化和漏洞分析相关的工作不断产生。然而，仍然很少有研究考虑如何从广泛的容器镜像中挖掘信息生成相应的标签准确描述容器的功能。现在，已有团队针对容器推荐挖掘容器的信息对容器镜像进行推荐。Kang Yin团队通过对容器的Dockerfile进行分析，建立了抽象语法树来分析标签的语法关系，进而向用户推荐容器[2-3]。目前的方法仅仅针对Dockerfile进行了分析，而没有考虑到大量容器镜像自带的短文本描述和长文本描述。同时目前的研究大多通过挖掘DockerHub容器仓库获取信息忽略了其他开源仓库和社区（GitHub、Stackoverflow）的信息。

鉴于Dockerfile文本的同质性特点导致很难从中提取到有效标签的问题，本文将长文本描述和短文本描述作为标签抽取的主要数据源，率先选择引入其他相关社区的信息进行知识融合与补充。其次，使用自然语言处理（NLP）方法进行标签的获取。同时，进行标签的扩展，便于将比较专业的概念提取出更加普适的概念，增加用户搜索的便利性。

考虑到现有研究的局限性，本文利用NLP对容器图像进行智能分析，并建立了一种最小化人为因素影响的方法，该方法包括以下四个贡献：

1. 从开源仓库和社区中挖掘了大量的短文本和长文本描述。数据主要从DockerHub中挖掘，通过GitHub和Stackoverflow来补充。
2. 我们提出了一种基于TF-IDF[4-5]和层次聚类的方法用于短文本描述生成标签，以及基于LDA[6]的主题文字生成方法用于长文本描述生成标签。
3. 我们使用了Hierarchical Conceptual Tagging 方法和Word2Vec和WordNet[7] 知识库用于实现纵向和横向标签扩展。
4. 我们将我们的方法与Wang等人提出的EnTagRec++[8]进行了比较。实验结果表明，我们的方法可以实现高召回率，尤其是在样本集较小的情况下。我们的方法在Recall@5方面也比EnTagRec++提高了12.6%，在Recall@10方面提高了10.9%。

本文的其余部分安排如下。第2节给出了本文工作的动机。第3节调查了标签提取相关研究的工作。第4节概述了所提出的方法，并解释了我们方法的细节和实施。第5节展示了我们的实验设置和研究问题及实验结果。第6节说明了本文的有效性威胁。最后，第7节做出结论并提到未来的工作。

## 2. Motivation

目前用户对DockerHub等容器仓库的搜索途径仍然依靠于容器提供者所提供的标签，难以准确并具体地描述容器的功能和特点。然而，容器仓库包含了许多有关容器功能和应用场景的长短文本描述及在其他开源社区例如GitHub、StackOverflow等也有容器的相关信息。因此，本文拟通过长短文本描述和开源社区中为容器镜像生成更多的标签进而准确地描述容器镜像的功能及特点。

现阶段DockerHub中的容器镜像提供的标签仅仅是一些通用的描述，例如容器镜像的总体用途、适用的体系结构和操作系统。例如，当我们需要搜索一个非关系型数据库的镜像，我们使用nosql database进行搜索，如图1所示，位于搜索排名前2的都是关系型数据库。因此，DockerHub中自带的标签难以给用户提供搜索机制，而DockerHub也仅在左侧边栏提供了标签的类型复选框用于搜索的筛选。可见，难以为用户提供有效的搜索功能。

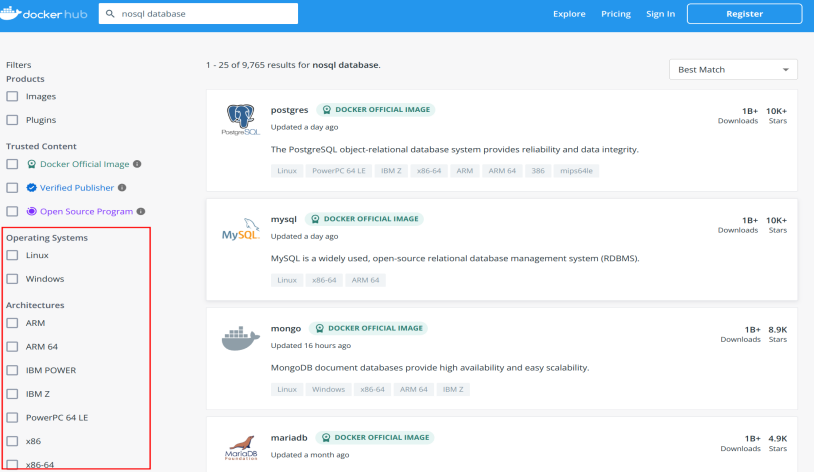


图1 DockerHub搜索示意图

DockerHub中的容器镜像一般提供容器镜像短描述和长描述，都包含高维语义知识。如图2所示，简短的描述通常会简要地给出容器的功能和非功能特性。而长描述通常会提供更多关于容器特性的内容，包括版本变化、许可证等。文本描述的优点是信息明显便于用户快速理解，值得挖掘。同时，其他开源社区例如GitHub也会提供容器镜像相关功能的长文本描述，StackOverflow也会提供关于容器镜像功能的回答。由于开源社区的开放性，目前能够公开访问的容器镜像超过了900万。可见，大量的数据并没有利用到现有的容器标签提取中。故本文拟通过多维度的数据及来源挖掘出容器镜像的特点生成有效标签。

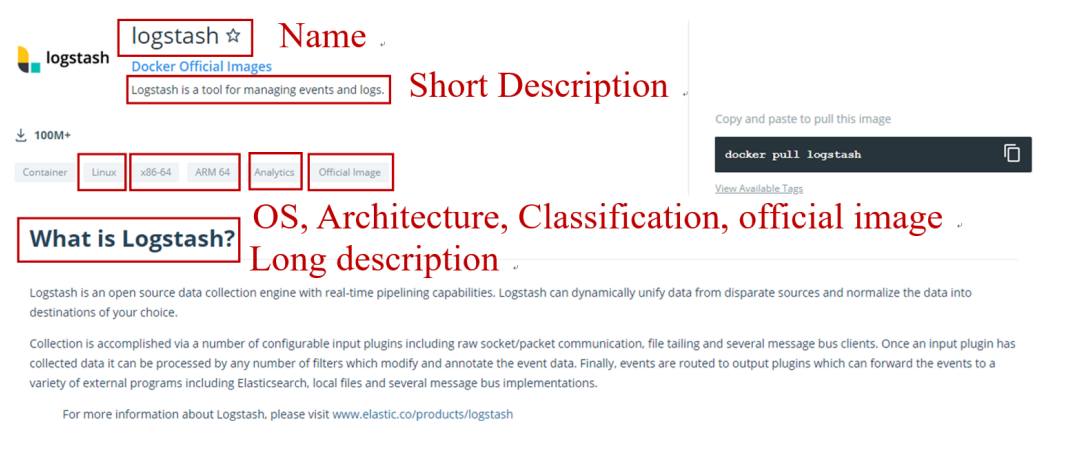


图2 短文本描述和长文本描述示意图

## 3. 相关工作

目前，分析容器数据挖掘的研究主要集中在提高DevOps的效率和容器故障分析方面。Xu，T等人通过挖掘容器存储库可以促进软件配置和软件测试，有助于开发人员了解应该如何配置软件系统[1]。Hassan等人通过分析项目代码和以前版本的Dockerfile，提出了一种帮助Dockerfile更新的新方法[9]。Zhang等人进行了实证研究，分析了Dockerfile演化轨迹对Docker的容器化质量和延迟的影响[10]。

关于基于容器数据的标签生成，从2019年开始受到研究关注，但目前研究成果较少。W Chen团队在容器标签生成做了一系列的工作。首先提出了一种容器图像存储库的标签推荐方法，通过分析Dockerfile文件应用L-LDA（Labeled Latent Dirichlet Allocation）来训练模型，生成标签[11]。再此研究的基础上，提出STAR，一种用于Docker Repositories的Specialized Tagging Approach，它使用Labeled Latent Dirichlet Allocation 算法来推荐标签[12]。后又研究了基于半监督学习的标签推荐方法，SemiTagRec[13]。可见，目前的研究均基于单一的Dockerfile数据，考虑的数据源较少。

由此可见，目前的研究局限于通过结构化的Dockerfile数据来生成描述容器的标签，没有考虑其他非结构化的数据，而容器镜像存在大量可利用的文本描述信息。因此，本文拟加入文本描述信息对现有方法进行补充，并考虑标签的扩展，为用户提供更有效的搜索机制以及基于标签的推荐活动。

## 4. 主体方法

### 4.1 整体过程

针对现有的研究问题，我们提出了三层的标签生成系统。其中第一层是数据处理层，第二层是标签生成层，第三层是标签扩展层，其整体的架构图如图3所示。

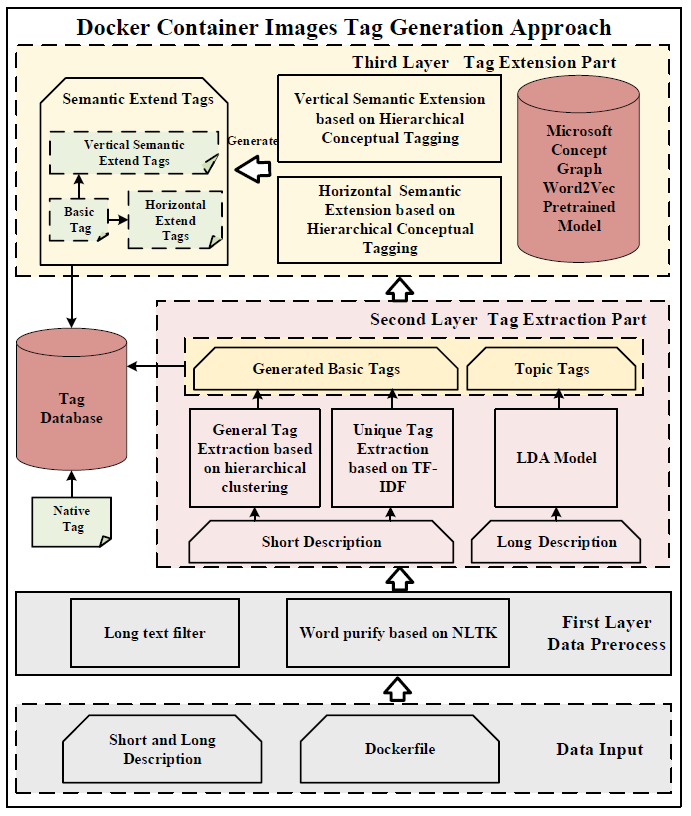


图3 整体方法架构图

数据处理层通过从DockerHub、GitHub和StackOverflow等社区中抽取容器镜像的短文本描述和长文本描述建立初始的输入数据集，并对长文本描述作数据处理，删除其中的超链接和图标等无用信息，确保数据的准确性。

标签生成层通过挖掘的数据源生成标签，对于短文本描述，标签生成层使用TF-IDF和层次聚类的方法来生成标签。对于长文本描述，则应用LDA主题模型。

标签扩展层将第二层生成的标签作为输入，通过分层概念标记方法用于在垂直语义视图中生成标记。同时，使用基于WordNet和Word2Vec的方法用于在水平语义视图中生成标签。

### 4.2 数据形式、提取与处理

如图4所示，本文基于scrapy爬虫框架软件通过DockerHub、GitHub以及StackOverflow等数据源获取了三类数据：一类是不需要进行数据处理的较简单的短文本描述。一类是需要进行数据处理的长文本描述，最后一类是一些用户对容器镜像功能的评价，需要处理为短文本描述和长文本描述。将三类信息保存到数据库中，用于后续的数据处理。

短文本描述其本身具有高准确性的特点，不需要对其进行过多的数据处理。而对于长文本描述而言，通常以MarkDown的形式存在，由于MarkDown中会存在较多的链接、图标以及图片等无用信息，故需要对长文本描述进行简单地数据处理，将链接、图标和图片等部分内容通过正则匹配删除，保留有用的信息。对于用户对功能的描述，删除主观评论的部分，根据长度分类到短文本描述和长文本描述中。



图4 数据处理过程

### 4.3 短文本标签抽取

由于短文本描述自身就有短内容和高准确性的特点，从短文本描述抽取标签的关键在于如何提高抽取标签的有效性。本文采用了TF-IDF和层次聚类的方法来生成标签。TF-IDF是一种常用的关键词提取算法，在通用领域的短文本信息提取中表现良好，本文提取的短文本描述与其他通用领域的短文本信息的特点几乎一致，都是类似对某功能进行描述。因此，本文采用TF-IDF算法提取关键标签。

此外，唯一标签在TF-IDF算法中得分较高，会导致生成的标签缺乏通用性。例如，诸如“数据库”、“服务器”之类的词存在于许多容器的描述中，这将极大地影响IDF分数。与唯一标签相比，通用标签更能体现容器镜像的功能特征，这在筛选中具有重要意义。因此，本文设计了一种基于层次聚类的通用标签提取方法来获取通用标签。

首先，本文首先计算TF-IDF得分。TF-IDF分数的计算如下所示，第一部分是TF分数，而第二部分是IDF分数。TF-IDF分数是通过将TF和IDF值相乘来计算的。本文使用TF-IDF的值的阈值来过滤低于阈值的单词而保留标签。





然后，本文基于层次聚类的通用标签提取方法来获取通用标签。伪代码如算法1所示。基本思想是使用层次聚类的方法将文档分类到簇中，然后选择合适的词作为簇的标签。最小余弦相似度用于计算簇之间的距离，其计算如下所示。Sim\_threshold的常数用于决定是否合并两个簇。

|  |
| --- |
| **Algorithm 1** General Tag Extraction based on hierarchical clustering |
| **Input:** documents(short description of each containers);  **Output:** general tags of each containers;  1: Use one-hot model encode each document  *di{w*1*,w*2*, · · · },wi ∈ {*0*,* 1*}*;  2: Sort *di* by words IDF score in descending order;  3: Initialize each document *di* as a cluster *ci*, store *ci*  in List *c{c*1*, c*2*, · · · }*;  4: Initialize constant W\_threshold, Sim\_threshold,  skip\_k, D\_threshold, P\_threshold;  5: Set K as;  6: **while** K *≥* W\_threshold **do**  7: *c*1*, c*2 = min(Similarity(*ci, cj* ));  8: **if** Similarity(*c*1*, c*2) *≥* Sim\_threshold **then**  9: merge(*c*1*, c*2);  10: **end if**  11: **end while**  12: K = K - skip\_k;  13: **for** c in clusters **do**  14: Initialize array w\_fre, set each element with 0;  15: **for** word in c **do**  16: wfre[w] = Sum(w in (*di* in c)) / c.length;  17: **if** w\_fre[w] >P\_threshold **and** c.length  >D\_threshold **then**  18: output w as tag for all di in c;  19: **end if**  20: **end for**  21: **end for** |



而在聚类之前，必须考虑的是应该使用多少单词来计算文档之间的相似度。本文使用表示要比较的词总和，即使用每个聚类的前个词计算聚类相似度。

由于词序和值对相似度有很大影响。对于文档中频繁出现的通用标签，应以文档集中词频作为衡量文档词序的指标。因此，通过one-hot模型对每个文档进行编码后，我们按照单词IDF得分降序对进行排序。

会影响相似度值，这意味着相似度值减小而增大。本文增加了常数skip\_k和W\_threshold，每次循环后都会减小skip\_k。如果低于W\_threshold，则结束聚类。

聚类后，为了找到每个簇的标签，本文首先初始化一个长度与词数相同的数组w\_fre来表示该簇的每个词频，然后计算词频并实现。考虑到标签的通用性，P\_threshold和D\_threshold用于决定一个词是否是一个标签。如果集群文档中的词频低于P\_threshold或集群长度低于D\_threshold，则该词不能是标签。简而言之，D\_threshold反映了聚类的一般性，P\_threshold体现了词的一般性。至此，基于层次聚类的通用标签提取算法的流程结束。

### 4.4 长文本标签提取

长文本描述包含了容器的特征和功能的高维信息，其描述的异构特性的主要问题可以利用其Markdown格式在一定程度上得到解决。我们使用其核心的功能信息来生成标签，而DockerHub长文本描述的格式让我们有机会过滤掉那些我们不需要的文本。由于其主要特征是从包含准确和简短信息的短文本描述中提取的，因此本文使用LDA主题模型对长文本描述进行分析，然后找到容器的潜在标签。LDA生成主题和主题词的可能性分布，通过寻找主题，我们将主题词作为容器的潜在标签。这里我们不生成容器中的特定单词。相反主题词可以成为未来挖掘容器的潜在标签或标签知识库。

### 4.5 纵向标签扩展

通过对短文本描述和长文本描述进行数据挖掘和自然语言处理生成的标签只能从静态的角度描述容器镜像，为了在动态上扩展标签，本文通过分析输入词关系找到概念词，采用Jiang Haiyun提出的分层概念标注方法。

分层概念标记方法使用贝叶斯玫瑰树(BRT)对输入词进行建模。然后采用层次聚类方法调整树的节点并生成标签。概念词生成的基本思想是通过Microsoft Concept Graph及其API，找到与词袋中尽可能多的词相关的概念词。

要解释这种方法，首先我们需要回顾一下BRT的基础知识。如图5所示，和 是都有孩子的示例树。他们可以加入一棵新树，可以吸，他们可以倒塌成一棵树。这里，词袋中的每个词都被视为封装在树中的一个叶子节点。然后它们将使用BRT的调整合并成一棵树。

用于衡量新树的适用性。代表调整的具体操作（Join、Absorb、Collapse）。如果它们的最大，则应将更多的两棵树与操作组合。

通过如下的公式计算：



表示新树的单词，计算如下。



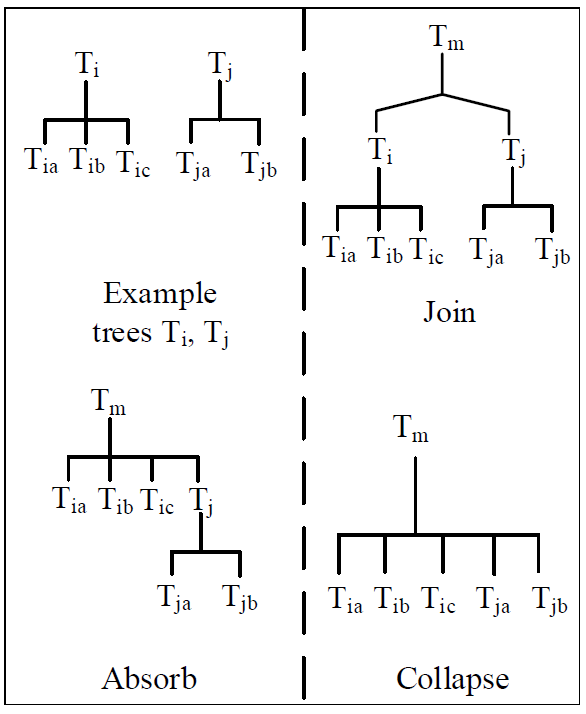


图5 贝叶斯玫瑰树

是权重因子。计算如下



符号代表概念词，是概念词列表。表示概念的先验概率，我们使用均匀分布对其进行建模。计算如下。



这里，表示是单词的概念词的概率，可以利用Microsoft Concept Graph API来访问。到这里为止，我们可以测量树之间的每一个操作，从而解决聚类过程的距离。随着合并的过程，概念词被生成并成为标签。

### 4.6 横向标签扩展

标签之间除了进行纵向的扩展，还可以进行横向的扩展，挖掘标签之间的联系。本文使用WordNet的同义词集生成用于水平语义扩展的标签。在WordNet中，每个词可能与许多同义词集相关，而每个同义词集都有唯一的定义并包含多个词。例如，单词“Java”具有三个同义词集，如图6所示。然而，前两个同义词集与信息技术领域无关。因此，在输入基本标签后，我们通过Word2Vec分析基本标签同义词定义来找到合适的同义词。

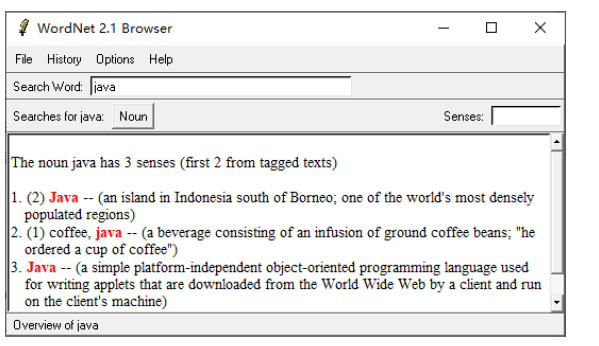


图6 WordNet示例

因此本文进一步通过Word2Vec分析基本标签同义词定义来找到合适的同义词。并使用BOW(Bag Of Words)对每个同义词集定义进行建模，然后计算定义和字段词之间的平均余弦距离 (Cset)以选择合适的同义词集。然后提取输入标签的同义词作为扩展标签。

## 5. 实验

### 5.1 实验设计

### 5.2 评估指标

### 5.3 实验数据

### 5.4 实验分析

## 6. 有效性威胁

几个威胁可能潜在地影响本文工作的有效性，讨论如下。

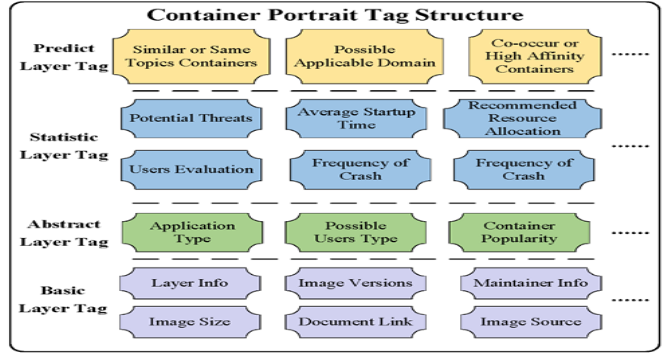
构建有效性的威胁来自手动检查精确度和召回率的计算。计算结果可能会受到主观感觉或实验者期望的影响。我们通过公开允许公众检查的统计记录来减轻这种威胁，我们学院的其他研究人员也参与了评估过程。同时，我们一直在计划建立一个标签不同等级的评估系统，我们希望它可以让更多的用户参与进来，以进一步减轻这种威胁。

收集数据的威胁来源于收集的数据不一定足够的充分及数据的有效性。本文已经从常见的开源社区中提取了较多的短文本描述和长文本描述来降低这种威胁带来的不利影响。同时，本文通过对收集的数据做预处理也减轻了这种威胁。

## 7. 结论和未来工作

在本文中，我们提出了一种新的容器镜像标签的生成方法，包括四个部分：基于NLP的短文本描述和长文本描述的标签生成以及基于推荐的横向和纵向的标签扩展。我们测试了近10万个样本。实验结果表明，该方法在Recall@5和Recall@10 中的值分别达到了0.791和0.842。与EnTagRec++相比，我们的方法可以实现高召回率，尤其是在样本集较小的情况下。我们的方法在Recall@5方面也比EnTagRec++提高了12.6%，在Recall@10方面提高了10.9%。我们的方法将NLP和基于推荐的方法进行了合理的结合，充分发挥了两者在标签生成方面的优势，并且在之前的研究中只使用Dockerfile作为数据输入，我们添加了文字描述弥补了Dockerfile缺乏语义信息的缺点。

未来，我们拟建立标签体系，将这些生成的标签分类到容器画像分层层次的不同层，如图X所示，研究不同层标签之间的语义关系以及标签分层在标签生成中的应用。同时，分析标签之间的关系，期望能够发现标签之间的语义关系，建立标签之间的知识图谱，从而可以推断出更接近容器特征的新标签，从而更好地应用到更多的应用中，并进一步研究容器推荐问题。



图X 容器标签结构层次示意图

## 参考文献

[1] Xu T, Marinov D. Mining container image repositories for software configuration and beyond[C]//Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results. 2018: 49-52.

[2] Yin K, Chen W, Zhou J, et al. STAR: a specialized tagging approach for Docker repositories[C]//2018 25th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2018: 426-435.

[3] Yin K, Zhou J, Chen W, et al. D-Tagger: A tag recommendation approach for Docker repositories[C]//Proceedings of the Tenth Asia-Pacific Symposium on Internetware. 2018: 1-10.

[4] Luhn H P. A statistical approach to mechanized encoding and searching of literary information[J]. IBM Journal of research and development, 1957, 1(4): 309-317.

[5] Jones K S. A statistical interpretation of term specificity and its application in retrieval[J]. Journal of documentation, 1972.

[6] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation: the Journal of machine Learning research, v. 3[J]. 2003.

[7] Miller G. Wordnet: a lexical database for english communications of the acm 38 (11) 3941[J]. Niemela, I, 1995.

[8] Wang S, Lo D, Vasilescu B, et al. EnTagRec++: An enhanced tag recommendation system for software information sites[J]. Empirical Software Engineering, 2018, 23(2): 800-832.

[9] Hassan F, Rodriguez R, Wang X. Rudsea: recommending updates of dockerfiles via software environment analysis[C]//Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018: 796-801.

[10] Jiang H, Zhang C, Yang D, et al. Hierarchical Conceptual Labeling[C]//International Conference on Database Systems for Advanced Applications. Springer, Cham, 2019: 230-234.

[11] Yin K, Zhou J, Chen W, et al. D-Tagger: A tag recommendation approach for Docker repositories[C]//Proceedings of the Tenth Asia-Pacific Symposium on Internetware. 2018: 1-10.

[12] Yin K, Chen W, Zhou J, et al. STAR: a specialized tagging approach for Docker repositories[C]//2018 25th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2018: 426-435.

[13] Zhou J, Chen W, Wu G, et al. SemiTagRec: a semi-supervised learning based tag recommendation approach for Docker repositories[C]//International Conference on Software and Systems Reuse. Springer, Cham, 2019: 132-148.