《中文信息处理》课程项目

13300200019 吴耀波

0 摘要

当用户执行在线探索性数据分析时，他们通常需要了解当前的数据以决定下一步操作。 概率主题模型可以帮助我们发现大型文档集合中的底层主题结构，而无需阅读每个文档。 本文提出了一种用于在文档子集中构造主题结构的方法。 在全局预计算的帮助下，该算法对于在线处理足够高效，同时在主题建模中保持可靠的性能。

1 介绍

随着互联网时代的数据爆炸，用户很难从大量数据中提取有用的信息，以帮助他们做出决策。阅读文本数据对用户来说事非常耗时的，因此帮助用户理解文档集合所讨论的工具变得非常有用。词云是一种广泛使用的方法来完成此任务，图1a示出了文档集的词云演示，从中可以容易地得出它是关于宇宙物理的集合。

然而，当集合由来自不同域的文档组成时，由词云生成的表示可能有点混乱。图1b说明了这样的情况，所呈现的词似乎从字典中随机选择（但实际上不是），你几乎不能理解集合谈论什么。

(a) word cloud

(b) word cloud for documents from different domains

(c) topic model representation

Fig. 1: Representations generated by various methods.

概率主题模型[1]是用于总结大文本语料库内容的技术。这些技术将主题模型化为词语的分布，并且将每个文档表示为这些主题的组合。提取的主题总结语料库并将精制信息传递给用户。对于大多数主题，词的分布往往是稀疏的，因此主题的良好表示是具有高概率的一组词。图1c示出了在先前文档集合上训练的主题模型的输出。

虽然主题模型提供了描述文档集的解决方案，但它不适用于在线分析处理。对于中等大小的子集（例如2000个文档，300,000个tokens），从头开始训练主题模型可能需要几分钟，这对于在线用户是不能容忍的。

在本文中，在在线处理的情况下研究子集主题建模的问题，并提出一个可行的解决方案。

2 Latent Dirichlet Allocation

本节首先回顾概率主题模型，重点是潜在狄利克雷分配（LDA）。 作为生成模型，LDA模型可以分为两部分，即概率生成过程和推理算法。

2.1 概率生成过程

LDA将收集M个文档为K个潜在主题$\Phi = \phi_{1:K}$，其中每个主题ϕ_k是T个词的词典上的多项分布。 LDA的生成处理如下：

① For each of the K topics, draw φ_k from a Dirichlet prior,$\varphi_k \sim Dirchlet(\beta)$

② For each of the M documents, draw topic proportion $\theta_m \sim Dirchlet(\alpha)$

③ For each word i in document m,

1.Draw topic assignm $z_{mi} \sim Multinomial(\theta_m)$

2.Draw word token $w_{mi} \sim Multinomial(\phi_{z_{mi}})$

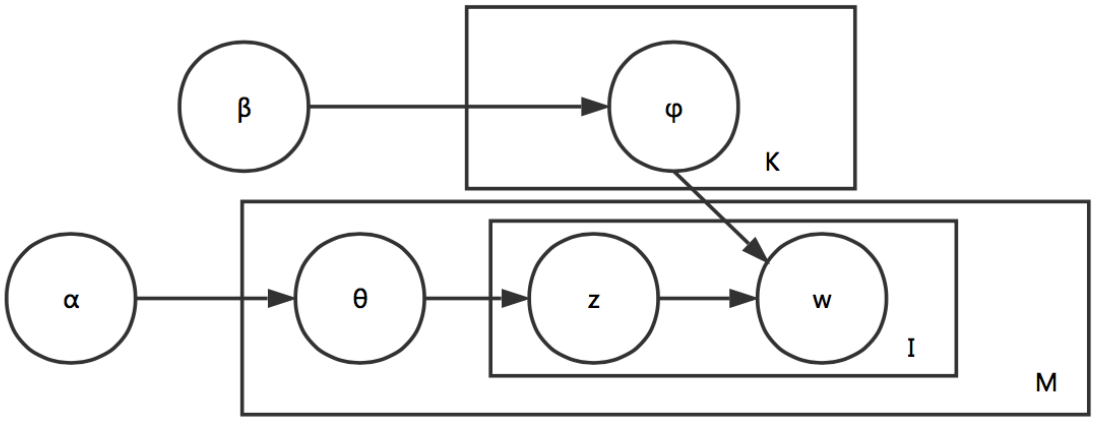


Fig. 2: Graphical model for LDA

Fig. 2: Graphical model for LDA

2.2 推理算法

给定生成模型和一组文档W_{mi}，LDA的核心是近似潜在随机变量的后验分布。 一种比较流行的推理方法是基于马尔科夫链蒙特卡罗（MCMC）的吉布斯采样[2]：

① For each word w_{mi} in each document, randomly assign a topic number z_{mi}

② Re-scan the corpus and use gibbs sampling to re-sample each word's topic z_{mi}

③ Repeat step2 until convergence

④ With flagged words, calculate the estimation of Φ and θ

Griffiths [2]给出了确定主题tokens的公式（等式1），在步骤2中使用的算法和在步骤4中使用的用于推断Φ和θ的算法（等式2，等式3）：

$$p(z_{mi} = k|\vec{z}_{\neg i}, \vec{w}) \propto \frac{n_{mk,\neg i} + \alpha_k}{\sum_{k=1}^{K}(n_{mk,\neg i} + \alpha_k)} \times \frac{n_{kt,\neg i} + \beta_t}{\sum_{t=1}^{T}(n_{kt,\neg i} + \beta_t)}$$

$$\hat{\theta}_{mk} = \frac{n_{mk} + \alpha_k}{\sum_{k=1}^{K}(n_{mk} + \alpha_k)}$$

$$\hat{\phi}_{kt} = \frac{n_{kt} + \beta_t}{\sum_{t=1}^{T}(n_{kt} + \beta_t)}$$

2.3 变量描述

为了方便阅读，下表列出来报告和代码中常用的变量描述。

Notation	Description
M	Number of documents in corpus
T	Number of distinct word in vocabulary
K	Number of topics
α, β	Prior parameters for dirichlet distribution
w_{mi}	ith word in document m
z_{mi}	Topic assigned to word w_{mi}
ϕ_k	Distribution over vocabulary of topic k
θ_m	Distribution over topics of document m
n_{mk}	Count of topic k assigned in document m
n_{kt}	Count of word t assigned to topic k
$subM$	Number of documents in subset
$subK$	Number of topics in subset

3 文档子集的主题探索

3.1 Naive 方法

如介绍所述，训练LDA模型是非常耗时的。 对于在线分析，对于某个子集从头开始运行完整的LDA模型不能满足时间要求。 Naive方法是选择目标子集中具有最高权重的语料库级别主题。 对于$subM$篇文档{w1，w2，...，wsubM}的子集，选择最主要的主题作为表示。我们以该方法作为baseline。 算法1给出了细节：

Algorithm 1 : baseline1 for topic modeling in subsets
1: Weight the topic distributions over documents $\{\vec{\theta}_1^{sub}, \vec{\theta}_2^{sub}, \ldots, \vec{\theta}_{subM}^{sub}\}$ by document length, get the topic distribution $\vec{\theta}^{sub}$ over target subset.
2: Based on $\vec{\theta}^{sub}$, choose $subK$ topics $\{\vec{\varphi}_1^{sub}, \vec{\varphi}_2^{sub}, \ldots, \vec{\varphi}_{subM}^{sub}\}$ with highest weight.
3: **return** $\{\vec{\varphi}_1^{sub}, \vec{\varphi}_2^{sub}, \ldots, \vec{\varphi}_{subM}^{sub}\}$ as the topics of target subset.

3.2 提出算法

受到Naive方法的缺点的启发，我设计了一种改进的方法来演示子集特定信息，其还涵盖所有文档并允许每个文档包含多个主题。

除了文档的主题比例$\vec{\theta}$，我们可以进一步保存所有在模型训练中分配的主题标记词。在被标记为词典的主题k的目标子集中的词的分布，表示为φ_k'，可以被视为次要主题。一方面，从目标子集中提取φ_k'中的优势词，从而保持子集特定信息。另一方面，这些词语从良好预训练的语料库级别主题中选择，这保证了主题相干性。

有相当多的次要主题，我们需要将这些主题合并成$subK$个“较大”的主题，这$subK$个主题应该是主题一致的。在主题合并中可以使用一些统计距离度量（K-L距离，Hellinger距离等）。完整的算法工作如下，

Algorithm 2: OLES algorithm for topic modeling in subsets
1: With word tokens of $subM$ documents in target subset $\{\vec{w}_1^{sub}, \vec{w}_2^{sub}, \ldots, \vec{w}_{subM}^{sub}\}$ and corresponding topic tokens $\{\vec{z}_1^{sub}, \vec{z}_2^{sub}, \ldots, \vec{z}_{subM}^{sub}\}$, count the number of word t flagged as topic k in target subset $N_{kt}^{(sub)}$ to get K minor topics $\{\vec{\varphi}_1^{'}, \vec{\varphi}_2^{'}, \ldots, \vec{\varphi}_K^{'}\}$.
2: Choose $subK$ topics from K minor topics $\{\vec{\varphi}_1^{'}, \vec{\varphi}_2^{'}, \ldots, \vec{\varphi}_K^{'}\}$ as the initial centers of $subK$ clusters.
3: Assign each of K minor topics to the cluster whose center is closet to the minor topic.
4: Merge minor topics in each cluster as this cluster's topic $\vec{\varphi}_{subk}^{sub}$, and take the merged topic as new center of this cluster.
5: Repeat step3-4 until convergence.
6: **return** $\{\vec{\varphi}_1^{sub}, \vec{\varphi}_2^{sub}, \ldots, \vec{\varphi}_{subK}^{sub}\}$ as the topics of target subset.

4　实验结果

4.1 Perplexity

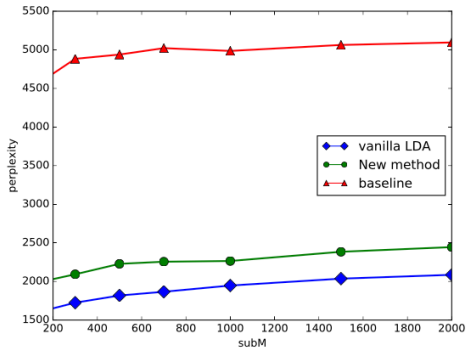
我们对两个数据集，Enron Email和一百万篇维基百科文章进行实验，在这里我使用$perplexity$来评估模型拟合的性能。 Perplexity是主题建模中广泛使用的度量，对于$subM$文档大小的测试集，perplexity是：

$$perplexity(D) = exp\{-\frac{\sum_{m=1}^{subM} \log \vec{w}_d}{\sum_{m=1}^{subM} N_m}\}$$

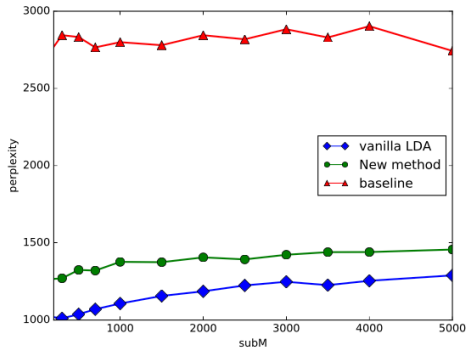
考虑到子集一般较小，如果将随机拆分的文档转化为训练集和测试集，两个集合的底层主题可能有很大的区别。为了防止这种情况，我们将每个文档随机分为两部分，80%的词被选为训练集中的文档，20%在测试集中。使用这种方法，测试集和训练集的基本主题没有很大差异。

图3a和图3b显示了不同方法对不同$subM$（子集中的文档数）的perplexity，图中的每个点是在相应条件下50个随机选择的子集的平均perplexity值。

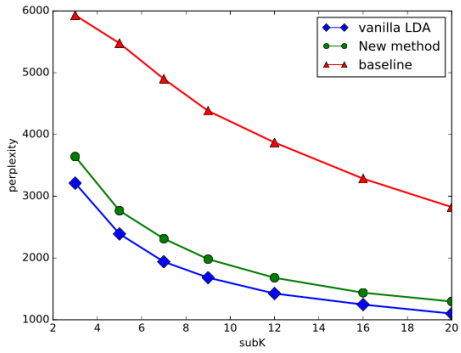
图3c和图3d显示不同方法对不同$subK$（子集中的主题数量）的perplexity。考虑到我们的任务是减少用户在集合中浏览的时间，在这种情况下，$subK$的值不应该太大。



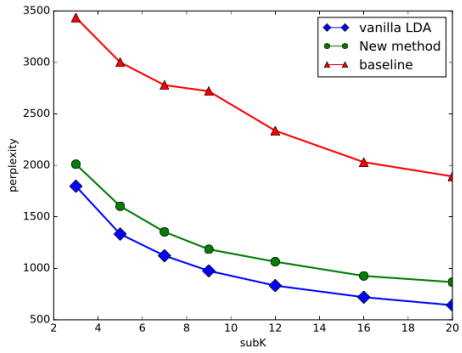
(a) Average perplexity of 50 runs on one million wikipedia articles (1000,000 docs, 200,000,000 tokens, 30,000 vocabulary size, 300 corpus-level pre-trained topics, subK=7)



(b) Average perplexity of 50 runs on Enron-Email dataset (500,000 docs, 80,000,000 tokens, 30,000 vocabulary size, 200 corpus-level pre-trained topics, subK=7)



(c) Average perplexity of 50 runs on one million wikipedia articles, with subM=1000



(d) Average perplexity of 50 runs on Enron-Email dataset, with subM=1000

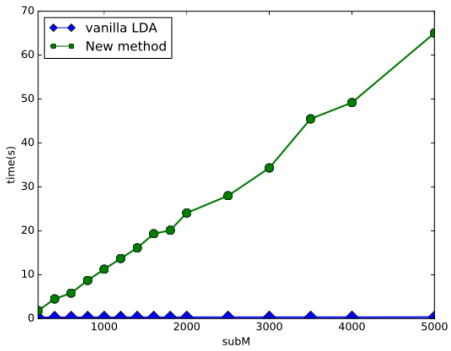
Fig. 3: Comparison of the perplexity results of five methods for subset modeling: re-training a LDA model in subset from scratch (vanilla LDA), New method, baseline.

4.2 Efficiency

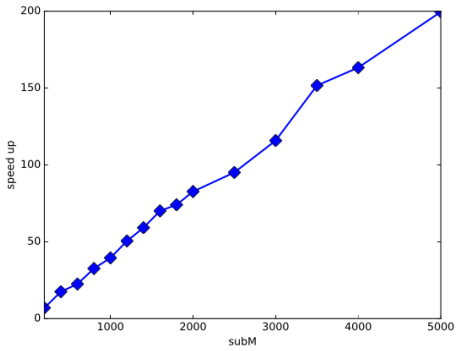
我在python / cython中实现了新算法，并使用gensim的LDA实现[3]作为基准。 我在维基百科数据集上测试新算法的效率与标准LDA在不同$subK$和$subM$的表现。 实验在2.1GHz Intel Xeon处理器和16GB RAM的Linux服务器上进行。 图5显示了实验结果。

对于来自具有300个预训练主题的语料库的中等大小（2,000个文档，30万个tokens，7个主题）子集，新算法可以在小于200毫秒内构建子集的主题结构，这满足在线探索性分析的要求。

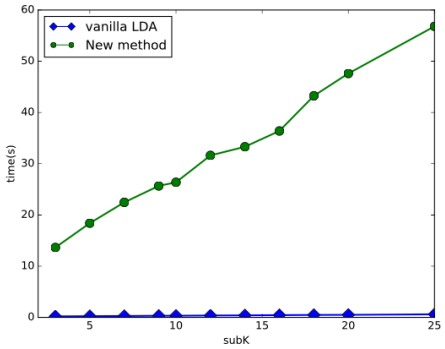
提出的算法中计算最密集的部分是K点的聚类过程，这要求比LDA少得多的训练时间。 在本节中，展示了效率比较的实验。



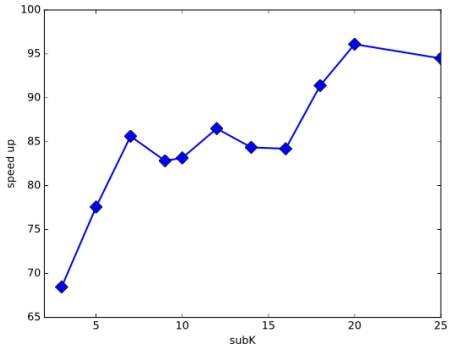
(a) Average time cost of 50 runs on wikipedia dataset, subK=7



(b) Efficiency speed up compared to standard LDA, subK=7



(c) Average time cost of 50 runs on wikipedia dataset, subM=2000



(d) Efficiency speed up compared to standard LDA, subK=7

Fig. 4: Efficiency promotion compared to standard LDA

5 参考文献

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machineLearning research 3(Jan), 993–1022 (2003)
2. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National academy of Sciences 101(suppl 1), 5228–5235 (2004)
3. Rehurek, R., Sojka, P.: Software Framework for Topic Modelling with LargeCorpora. In: Proceedings of the LREC 2010 Workshop on New Chal-lenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010),http://is.muni.cz/publication/884893/en