

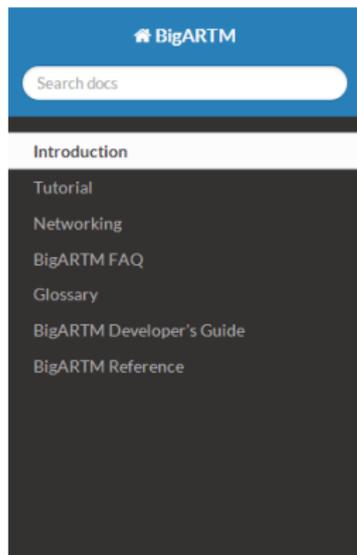
Библиотека BigARTM

Александр Фрей, Мурат Апишев

30 сентября 2014

BigARTM (<http://bigartm.org>)

- Комбинирование многих критериев при построении тематической модели.
- Параллельная распределённая обработка больших коллекций.



The screenshot shows the BigARTM website's navigation menu. At the top, there is a blue header with the BigARTM logo and a search bar labeled "Search docs". Below the header is a dark grey sidebar with a list of navigation items: "Introduction", "Tutorial", "Networking", "BigARTM FAQ", "Glossary", "BigARTM Developer's Guide", and "BigARTM Reference". The "Introduction" item is highlighted in white.

[Docs](#) » [Introduction](#)

[Edit on GitHub](#)

Introduction

Warning

Please note that this is a beta version of the BigARTM library which is still undergoing final testing before its official release. Should you encounter any bugs, lack of functionality or other problems with our library, please let us know immediately. Your help in this regard is greatly appreciated.

This is the documentation for the BigARTM library. BigARTM is a tool to infer [topic models](#), based on a novel technique called [Additive Regularization of Topic Models](#). This technique effectively builds multi-objective models by adding the weighted sums of regularizers to the optimization criterion. BigARTM is known to combine well very different objectives, including sparsening, smoothing, topics decorrelation and many others.

Ключевые особенности

- 1 **Онлайн-алгоритм** — библиотека не хранит в памяти всю коллекцию, загружая её частями по мере необходимости;
- 2 **Параллельность** — многопоточная обработка на ноде;
- 3 **Распределённость** — библиотека спроектирована для работы на кластере¹;
- 4 **Кроссплатформенность** — поддержка Windows и Linux²;
- 5 **Интерфейс** — на Python, добавятся на C++ и Java;
- 6 **Много моделей** — одновременное обучение нескольких моделей по одним данным.

¹Функциональность реализована, но нуждается в доработке.

²Протестировано на Ubuntu и Fedora.

Online Batch PLSA-EM

- 1: инициализировать ϕ_{wt} для всех $w \in W$ и $t \in T$;
- 2: $n_{wt} := 0, n_t := 0$ для всех $w \in W$ и $t \in T$;
- 3: **для всех** пакетов $D_j, j = 1, \dots, J$ **выполнять**
- 4: $\tilde{n}_{wt} := 0, \tilde{n}_t := 0$ для всех $w \in W$ и $t \in T$;
- 5: **для всех** $d \in D_j$ **выполнять**
- 6: инициализировать θ_{td} для всех $t \in T$;
- 7: **повторять**
- 8: $p_{tdw} := \frac{\phi_{wt}\theta_{td}}{\sum_s \phi_{ws}\theta_{sd}}$ для всех $w \in d, t \in T$
- 9: $\theta_{td} := \frac{1}{n_d} \sum_{w \in d} n_{dw} p_{tdw}$ для всех $t \in T$
- 10: **пока** θ_d не сойдётся;
- 11: $\tilde{n}_{wt}, \tilde{n}_t += n_{dw} p_{tdw}$ для всех $w \in W$ и $t \in T$;
- 12: $n_{wt} := \rho_j n_{wt} + \tilde{n}_{wt}; n_t := \rho_j n_t + \tilde{n}_t$ для всех $w \in W, t \in T$
- 13: $\phi_{wt} := n_{wt}/n_t$ для всех $w \in W, t \in T$

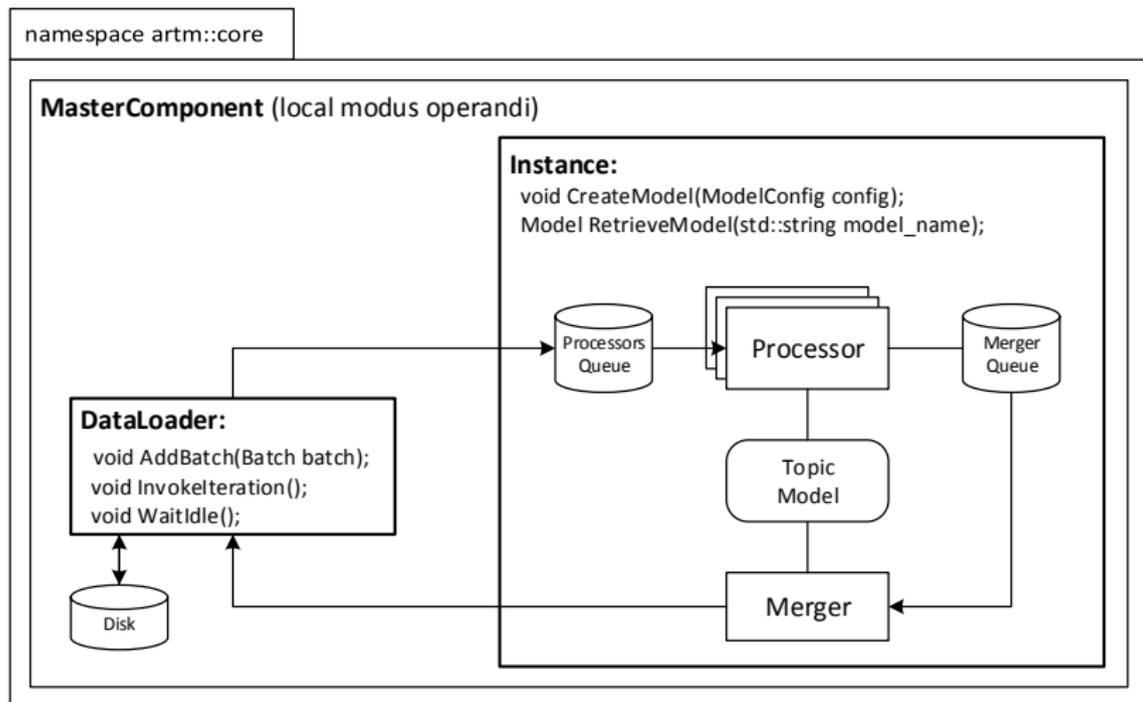
Online Batch PLSA-EM

Описанный онлайн-алгоритм обладает рядом преимуществ перед обычным:

- Не нужно хранить трехмерную матрицу p_{tdw} ;
- Можно отказаться от хранения матрицы Θ — её значения считаются по ходу работы и «забываются» после использования;
- Распределения ϕ и θ сходятся достаточно согласованно, процесс легко регулируется;
- Данный алгоритм допускает эффективную параллельную реализацию.

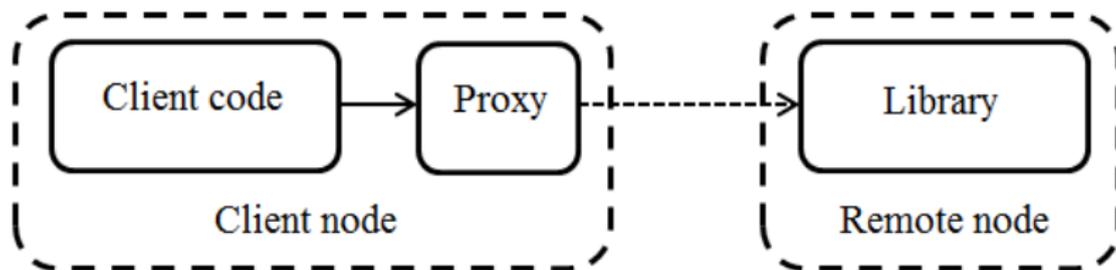
Сейчас библиотека работает с коллекцией в виде «мешка слов». В будущем планируется поддерживать обработку данных и в исходном текстовом виде.

Многопоточная обработка



Многопоточная обработка

Сейчас библиотека в первую очередь предназначена для использования на одной мощной машине. Такой компьютер можно арендовать в различных сервисах (Amazon EC2, Windows Azure и т.п.). Для подобной ситуации в BigARTM предусмотрен проху-режим. Он позволяет производить вычисления на удалённой машине, запуская клиентский код со своей. Установка библиотеки на чистую машину с Ubuntu и распаковка тестовых данных занимают 10-20 минут³.



³Компьютер с 8 ядрами и 15 Гб оперативной памяти

Распределённая обработка

- MapReduce/Hadoop — не подходит для данной задачи;
- MPI — не используется в данный момент, однако в будущем может стать основой распределённой реализации;
- Механизм RPC (библиотека grpc) — используется сейчас по нескольким причинам:
 - 1 grpc является надстройкой над ZeroMQ, которая в ряде проектов успешно заменила MPI;
 - 2 grpc предназначена для работы с Google Protocol Buffers;
 - 3 grpc автоматически управляет потоками (Threads) на вызывающей и удаленной стороне.

Архитектура распределённой обработки аналогична параллельной. Текущая распределённая реализация является работоспособной, но не протестированной с точки зрения эффективности. В будущем её ожидают большие изменения.

Механизм регуляризации

BigARTM реализует идею аддитивной регуляризации тематических моделей. Регуляризаторы подключаются в виде плагинов. Каждый плагин — это класс, который реализует метод регуляризации Φ или Θ .

- Регуляризатор Φ производит поправку во время обновления матрицы.
- Регуляризатор матрицы Θ вызывается на каждом проходе каждого документа.

Механизм регуляризации

Регуляризаторы в BigARTM:

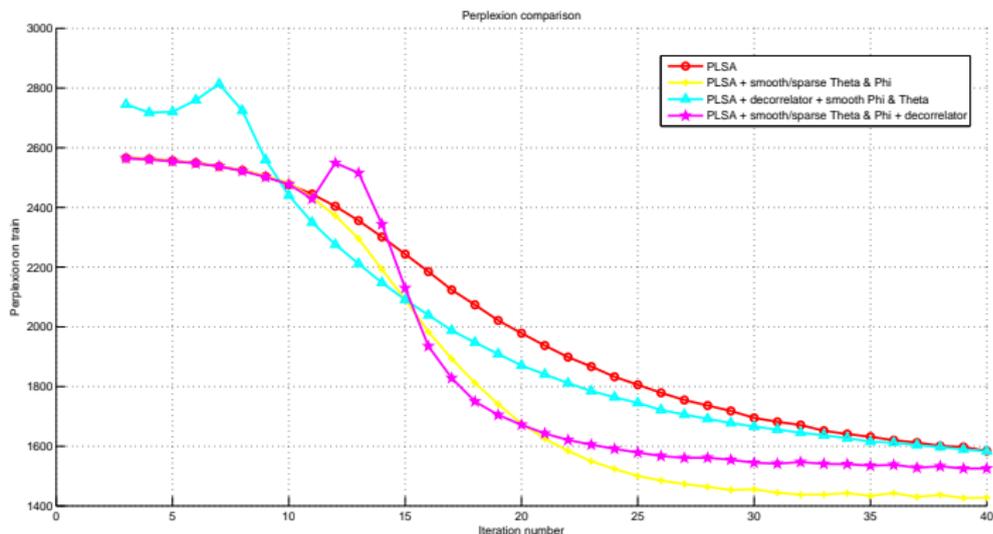
- Регуляризатор Дирихле
- Регуляризатор сглаживания/разреживания
- Декоррелятор тем
- Мультиязычный регуляризатор
- Регуляризатор для частичного обучения
- Регуляризатор сглаживания динамических тем
- Регуляризатор максимизации когерентности

Регуляризатор имеет разнообразные параметры. Для хранения объёмных данных создан механизм статических словарей.

Пример: регуляризатор сглаживания/разреживания, данными для которого являются счётчики всей коллекции.

Функционалы качества

Функционал качества — это так же класс-плагин, оценивающий параметры либо Φ , либо Θ . На картинке графики убывания перплексий обычной и регуляризованных моделей:



Функционалы качества

Функционалы качества в BigARTM:

- Перплексия
- Разреженности матриц Φ и Θ
- Объём, чистота и контрастность ядер тем
- Наиболее вероятные слова в теме
- Когерентность тем

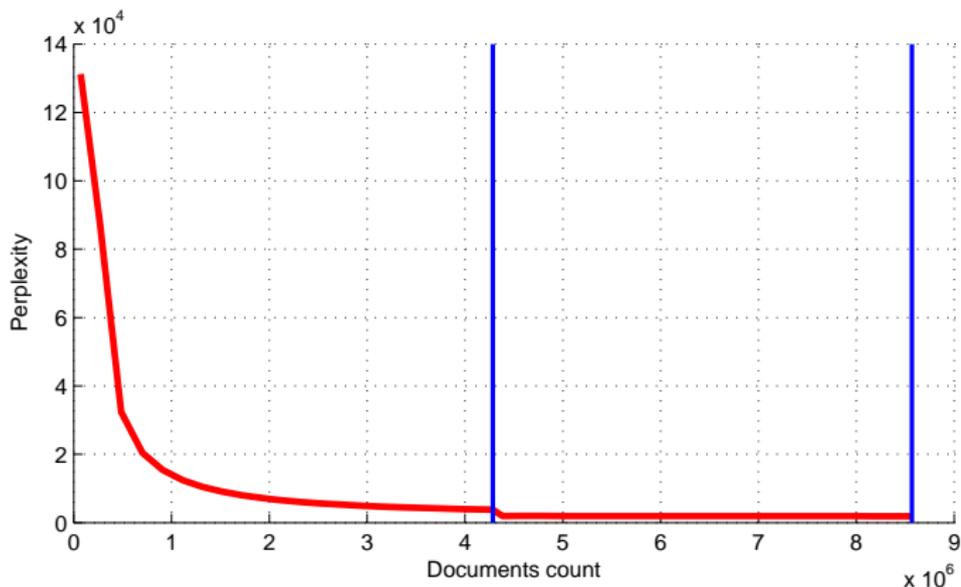
Как и регуляризаторы, функционалы получают параметры в виде словарей. Пример: перплексия, где проблема нулевой вероятности под логарифмом решается по униграмной модели коллекции (где и нужен словарь с информацией о коллекции). Результаты работы функционалов помещаются в заданные классы. Основным требованием к функционалу является легковесность (несколько Кб) его результата.

Мультимодальные тематические модели

Библиотека позволяет строить тематические модели с любым числом матриц Φ . Поэтому она может решать задачи классификации и категоризации документов, строить мультязычные модели. Мультимодальные модели позволяют эффективно использовать имеющиеся в документе мета-данные (авторы, дата издания, теги и т.п.).

Эксперименты на Pubmed

Библиотека была запущена на коллекции Pubmed, выявлялось 100 тем. Использовалась машина с 8 ядрами и 15 Гб оперативной памяти. Два прохода по коллекции заняли 1300 секунд. Перплексия сошла, что можно наблюдать на графике:



Стороннее ПО

Список стороннего ПО, необходимого для разработки и/или использования BigARTM ⁴ :

- **Boost**
- **Google Protocol Buffers** (создание всех структур данных и организация их пересылки)
- **rpcz** (RPC библиотека для Protocol Buffers на базе ZeroMQ)
- **ZeroMQ** (низкоуровневая организация сетевых взаимодействий)
- **Python** (интерпретатор для выполнения пользовательских скриптов)
- **glog** (библиотека логгирования для C++ кода ядра)
- **googletest** (библиотека для unit-тестирования кода ядра)
- **CMake** (кроссплатформенная build-система для сборки библиотеки)

⁴Все лицензии для данных программных продуктов описаны в файле LICENSE в корне BigARTM

Ссылки и документация

<http://bigartm.org/>

— официальный сайт проекта BigARTM с документацией.

<https://github.com/bigartm/bigartm>

— открытый репозиторий проекта BigARTM.

<https://github.com/bigartm/bigartm/releases>

— релизы библиотеки. Процесс установки под Linux описан в документации, для установки по Windows достаточно скачать релиз и установить Python 2.7.

Ссылки на статьи

Ниже приведён список публикаций, использованных при проектировании BigARTM:

- 1 David Newman, Arthur Asuncion, Padhraic Smyth and Max Welling — Distributed Algorithms for Topic Models, 2009.
- 2 A. J. Smola and S. Narayanamurthy — An architecture for parallel topic models, 2010.
- 3 Arthur Asuncion, Padhraic Smyth, Max Welling — Asynchronous Distributed Estimation of Topic Models for Document Analysis, 2010.
- 4 Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen and Edward Y.Chang — PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications, 2009.
- 5 Ke Zhai, Jordan Boyd-Graber, Nima Asadi, Mohamad Alkhouja — Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce, 2012.
- 6 Liu, Z., Zhang, Y., Chang, E. Y., and Sun, M. — PLDA+: Parallel Latent Dirichlet Allocation with Data Placement and Pipeline Processing, 2011.