Hierarchic Topic Models Visualization

Research project report

Author: Dmitriy S. Fedoriaka

Group: 374, Moscow Institute of Physics and Technology

Supervisor: V.V. Strijov

Date: 13.09.2016 – 27.11.2016

TABLE OF CONTENT

1. Introduction....2
- 1.1. Project description....2
  - 1.1.1. Abstract....2
  - 1.1.2. Motivation....2
  - 1.1.3. Literature....2
- 1.2. System architecture....2
  - 1.2.1. Problem statement filename....2
  - 1.2.2. A1 diagram....3
  - 1.2.3. A2 diagram....3
- 1.3. Project structure....4
- 1.4. Basic data structures....4

2. Module interfaces....5
- 2.1. HTTP Server....5
- 2.2. Data processing....5

3. UNIT TESTS....6

4. System launch and testing....7
- 4.1. System launch....7
- 4.2. System testing....7
- 4.3. System performance....7

5. Project review....8

6. The project is located ON:....9

1. INTRODUCTION

1.1. Project description

1.1.1. Abstract

Hierarchic topic models are good tool for representing big amount of text documents. However, displaying such models on screen is difficult problem itself.

This project implements interactive visualization of hierarchic topic models with inserted polygons. It uses topic modelling algorithms provided by BigARTM library.

1.1.2. Motivation

A. Project goal (main purpose of research and development, the expected result)

The goal is to develop fast and flexible system, which will detect topic structure of large document set and render its visualization.

B. Project application (where and how the project result will be applied)

Making convenient navigation for websites and other knowledge databases.

C. Historical data description (timing, formats and structures)

Data is set of preprocessed text documents of set in one of two specified formats: UCI or Vowpal Wabbit. Those formats are described on http://docs.bigartm.org/en/stable/formats.html. Also formats are described in documentation

D. Quality criteria (target functions, error functions, how the project quality is measured)

Quality criteria of visualized topic model is improvement ratio, which shows how fast user can perform search in dataset using visualization, compared to some basic system (such as search by key words).

E. Requirements for the project implementation (must be satisfied for successful project end)

There must be a service, which builds quality visualizations for any well-defined set of documents.

F. Feasibility of the project (how to prove the project feasibility, list of possible risks and complications)

Basic algorithms of topic modelling are implemented in BigARTM, at least one variant of visualization can be provided with FoamTree. For other actions, such as extracting tree structure from topic model there are simple algorithms.

G. Methods and solutions (project roadmap, methods and algorithms)

All code is written with Python.

HTTP server is implemented with bottle.py

External modules:

- BigARTM (http://bigartm.org) – Topic Modeling.
- FoamTree (https://carrotsearch.com/foamtree-overview) –Visualization.

Main algorithm behind topic modelling is EM-algorithm, enhanced with regularizers.

For visualization model of inserted polygons is used (see article with mathematical problem statement).

1.1.3. Literature

- K. Vorontsov, A.Potapenko – Tutorial on Probabilistic Topic Modeling: Additive Regularization for Stochastic Matrix Factorization Basic bibliography records and a short explanation of their role in the project, 2014.
- N. A. Chirkova, K. V. Vorontsov – Additive Regularization for Hierarchical Multi-modal Topic Modeling, 2016.

1.2. System architecture

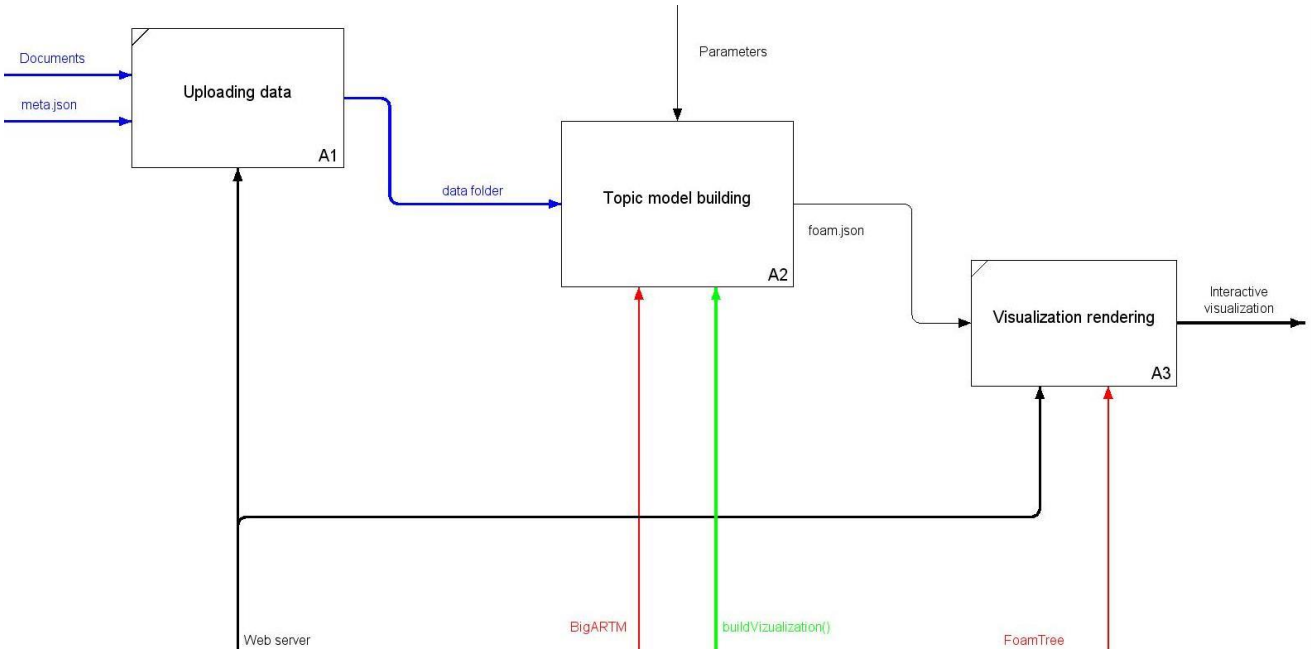
1.2.1. Problem statement filename

The mathematical problem statement can be found at the address:

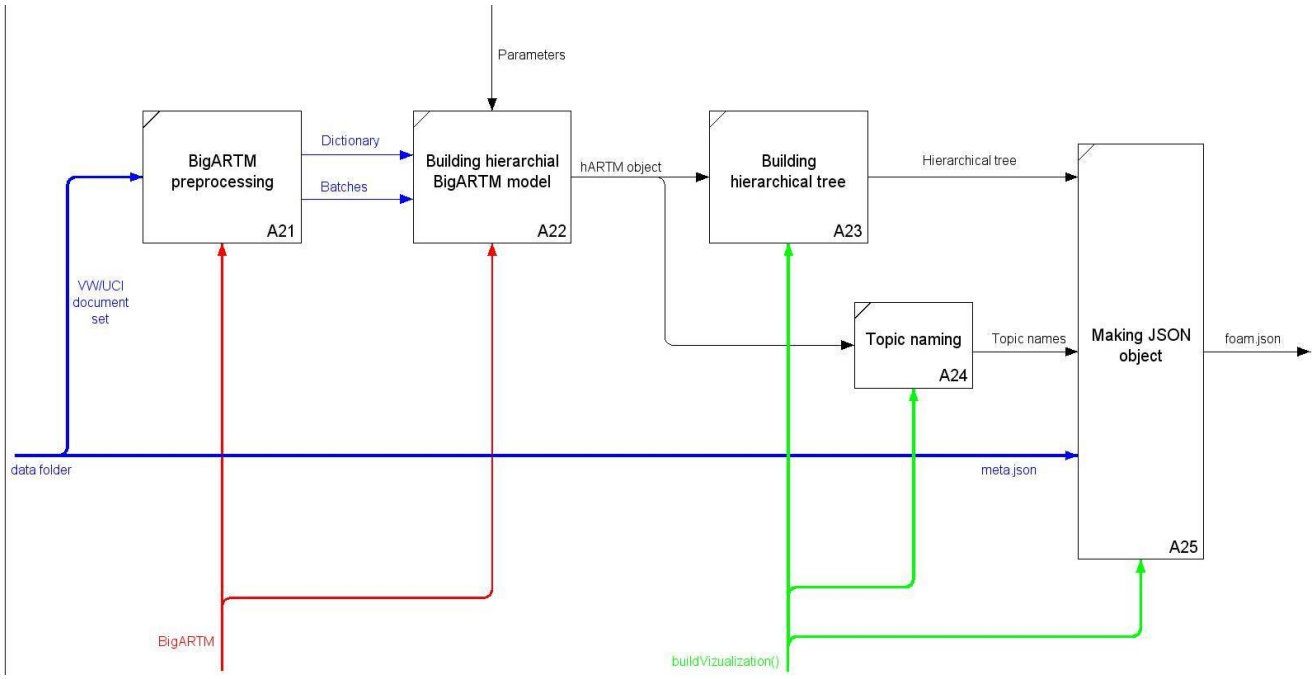
http://svn.code.sf.net/p/mlalgorithms/code/Group374/Fedoryaka2016TopicModelVisualization/doc/Fedoryaka2016TopicModelVisualization.pdf

1.2.2. A1 diagram

System architecture is described with the IDEF0 format.



1.2.3. A2 diagram



1.3. Project structure

System files are located in folders:

- doc – the project documentation;
- FancyTopicVisualizer – implementation of project:
    - ftv.py – core algorithm;
    - ArtmVisual.py – auxiliary code, dealing with visualization;
    - ftvserver.py – web server;
    - test.py – unit tests;
    - data – folder with datasets;
    - visual – folder with stored visualizations;
    - visual_examples – folder with sample visualizations;
    - template – templates of web pages for web server (including online documentation);
    - static – static files for web server, including FoamTree scripts;

1.4. Basic data structures

meta.json. Optional JSON file which provides additional information about data set (i.e. document titles, links to their location on the Internet and snippets).

ARTM batches. Set of files, which contain information about data set.

ARTM model. Python class which contains all information about document set and provides methods for fitting topic model.

TopicView. Python class which represents node of hierarchical tree.]

foamtree.json. JSON file, which describes hierarchical tree and is used by FoamTree for visualization rendering.

2. MODULE INTERFACES

2.1. HTTP Server

User interface of the system is implemented using HTTP server with paths as follows:

- / – home page with basic information.
- /upload – upload new dataset.
- /start – build new visualization.
- /visuals – list of visualizations.
- /visuals_examples – examples.
- /docs – documentation.

2.2. Data processing

Main function for data processing is function buildVisualization() in module ftv.py.

Input: single argument **params**, which is dictionary and contains all parameters:

- params[“dataset”] – name of dataset (string).
- params[“levelCount”] – the number of hierarchy levels (integer).
- params[“topicNum”] – numbers of topics on each level (list).
- params[“num_passes”] – the number of iterations of EM-algorithm (integer).
- params[“files”][“foamFile”] – path to output file.
- params[“files”][“logFile”] – path to log file.

Files with dataset should be put in /data/<params[“dataset”]> folder.

Function returns nothing, but it writes answer in JSON format to file <params[“files”][“foamFile”]>.

3. UNIT TESTS

Script **unit.py** provides one unit test for function buildVisualization().

It generates small dataset, where topics are disjoint by words, so topic structure is univocal. Then it builds topic model with buildVisualization() function and checks it.

4. SYSTEM LAUNCH AND TESTING

4.1. System launch

To launch the system, one should do following:

- Install python.
- Install BigARTM using following instructions http://docs.bigartm.org/en/stable/installation/windows.html#configure-bigartm-python-api.
- Install bottle.py web framework.
- Download files from http://svn.code.sf.net/p/mlalgorithms/code/Group374/Fedoryaka2016TopicModelVisualization/FancyTopicVisualizer.
- Run python script **ftvserver.py**.

Service will be available at http://localhost.

4.2. System testing

To test the system with unit tests, launch script **test.py**.

To test visualizations, go “Build new visualization” page and build visualization for any f prepared datasets.

4.3. System performance

System is able to measure duration of processing requests. System was tested on several datasets with following results.

Dataset	Number of documents	Number of levels	Number of topics on each level	Number of iterations	Elapsed time, seconds
kos	3430	1	15	10	5
		2	15; 50	20	28
enron	39861	1	10	20	65
		2	10; 40	20	153
nips	1500	1	10	20	10
		2	10;40	20	32

5. PROJECT REVIEW

6. THE PROJECT IS LOCATED ON:

http://svn.code.sf.net/p/mlalgorithms/code/Group374/Fedoryaka2016TopicModelVisualization/