Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

Национальный исследовательский университет

«Высшая школа экономики»

Факультет информатики, математики и компьютерных наук

**Программа подготовки магистров по направлению**

**01.04.02 Прикладная математика и информатика**

**ПРОЕКТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОМУ СЕМИНАРУ**

На тему «Поиск и автоматический анализ научных публикаций.»

**Руководитель проекта:** Крылов Владимир Владимирович

**Список участников проекта:**

Ларюшина Юлия

Шашкин Павел

Желонкин Дмитрий

Соколов Артем

Кузнецов Владимир

Куренков Евгений

Мозохин Дмитрий

Груздев Алексей

Вороная Ксения

Нижний Новгород, 2016

**Оглавление:**

1 Сбор данных для исходной задачи, создание модели поиска тематически близких документов ……………………………………………………... 3

2 Исследование различных методов построения summary и выявление наиболее качественного саммаризатора……………………………...…. 6

3 Построение summary с использованием модуля Gensim и извлечение keywords из статьи....................................................................................... 10

4. Исследование возможностей по созданию тематической модели на основе научных работ……………………………………………………..…..…. 14

**1 Сбор данных для исходной задачи, создание модели поиска тематически близких документов**

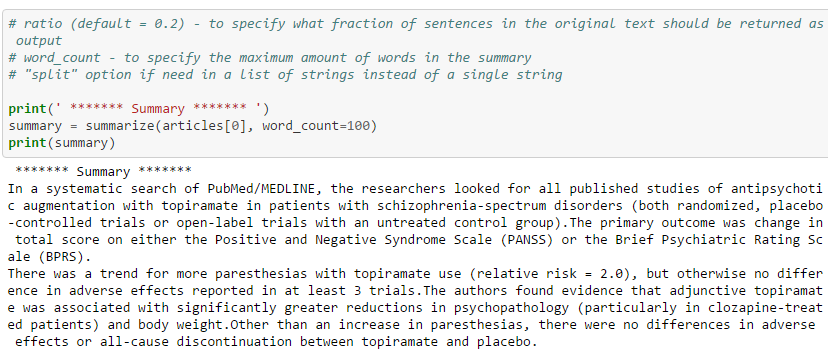
|  |  |
| --- | --- |
| **Задача группы** | Сбор данных для задачи, создание модели для поиска тематически близких документов, создание интерфейса для поиска тематических близких документов к заданному, визуализация и оценка полученных результатов. |
| **Исполнители** | ◦ Ларюшина Юлия  ◦ Шашкин Павел  ◦ Желонкин Дмитрий |
| **Используемые библиотеки и программные средства, данные** | ◦ Язык R, python2  ◦ Пакеты R: magrittr, dplyr, quanteda, tidyr, stm, lubridate, stmCorrViz, LDAvis, aRxiv, pdftools, parallel, rplos  ◦ Библиотеки python: nltk, gensim, numpy, pandas, scipy  ◦ Библиотека для мультимодального тематического моделирования bigARTM  ◦ Источники данных:  <https://www.plos.org/>  <https://arxiv.org/> |
| **Решенные подзадачи** | **1. Сбор документов для обучения моделей из открытых источников.**  Для тренировки модели были собраны данные по медицинской тематике из открытых источников.   * Для сбора данных из источника arxiv создан скрипт:   <https://github.com/artezio-kseniav/text_summarization/blob/master/data_collection/corpora_getter.R>  Данные (статьи + метаданные) могут быть собраны в формате pdf и преобразованы в текст.   * Для сбора данных из источника plos создан скрипт:   <https://github.com/artezio-kseniav/text_summarization/blob/master/data_collection/search_plos_example.R>  Данные изначально представлены в формате xml и разбиты на необходимые структурные единицы (article, body, abstract + metadata).  **2. Обработка корпуса для построения модели.**  **3. Построение мультимодальной тематической модели с использованием bigARTM.**  **4. Создание интерфейса для взаимодействия с мультимодальной тематической моделью (поиска близких документов).**  Подробная инструкция по использованию и работе доступна по ссылке: <https://github.com/artezio-kseniav/text_summarization/tree/master/python-topic-modeling>  **5. Построение структурированной тематической модели для поиска близких документов.**  Для обработки корпуса и построения модели был создан функционал: <https://github.com/artezio-kseniav/text_summarization/blob/master/topics/plos_stm.R>   1. Для начала средствами библиотек dplyr, tidyr и lubridate данные, собранные с plos, приводятся к виду id + дата\_публикации + секция\_документа + текст. 2. Средствами quanteda осуществляется построение матрицы частот для тренировки модели. 3. Производится обучение модели stm, в которой распределение тем зависит не только от секции документа, но и от даты публикации. 4. Полученные темы (topics) визуализируются средствами LDAvis (производится многомерное шкалирование векторов статей) и stmCorrViz (производится иерархическая кластеризация матрицы корреляции тем).   **6. Визуализация полученных результатов для формирования представления о собранном корпусе, экспертной оценки выделенных тем и формирования представления о предметной области:**  <https://github.com/artezio-kseniav/text_summarization/blob/master/topics/corrviz.html>  <https://github.com/artezio-kseniav/text_summarization/blob/master/topics/simil.html> |
| **Нерешенные подзадачи** | 1. Создание функционала для взаимодействия с документом в формате pdf  2. Создание функционала для обновления внутренней базы статей  3. Объединение модуля с модулями других подгрупп |
| **Связь с задачами других групп** | Созданная модель осуществляет сбор поискового корпуса, его обработку и поиск близких документов с точки зрения распределения тем в нём и в документах корпуса. После идентификации близких документов предполагается передача данных на вход модулю, осуществляющему формирование аннотаций (summary) для конечного пользователя. |

**2 Исследование различных методов построения summary и выявление наиболее качественного саммаризатора**

|  |  |
| --- | --- |
| **Задача группы** | • Исследовать различные методы построения summary  • Собрать тестовый набор данных  • Выявить наиболее качественный саммаризатор |
| **Исполнители** | • Груздев Алексей  • Мозохин Дмитрий |
| **Используемые библиотеки и программные средства, данные** | • python3  • Библиотеки python: nltk, sumy, seaborn  • Источники данных:  <https://www.plos.org/> |
| **Решенные подзадачи** | **1. Сбор документов для обучения моделей из открытых источников.**  **2. Наиболее яркий представитель методов построения summary основанных на** [Deep Learning - Sequence-to-Sequence with Attention Model for Text Summarization by Google](https://github.com/tensorflow/models/tree/master/textsum). Более подробно она описана в [Google Research Blog'е](https://research.googleblog.com/2016/08/text-summarization-with-tensorflow.html). Данная модель была натренирована на [GigaWord](https://catalog.ldc.upenn.edu/LDC2012T21) dataset'e, который является закрытым для доступа из вне - даже для ведения исследований в университете, без предварительной договоренности между университетом и владельцами датасета. Процесс тренировки модели с использованием Deep Learning библиотеки Tensorflow выложен в общий доступ, но в данной модели отсутствуют веса, посколько они были получены на закрытом датасете, поэтому без получения данного датасета, невозможно использовать Sequence-to-Sequence модель для построения саммаризатора. Поэтому далее были рассмотрены более классические методы построения summary:   1. TextRank – метод, основанный на представлении исходного текста в качестве графа, где вершине соответствует некоторый токен(например, предложение). Весам ребер приписывается значение семантической близости смежных вершин и после чего применяется Google Page Rank алгоритм, с помощью которого определяется набор "наиболее значимых" токенов, выступающих в качестве summary. 2. LSA - то метод обработки информации на естественном языке, анализирующий взаимосвязь между коллекцией документов и терминами в них встречающимися, сопоставляющий некоторые факторы (тематики) всем документам и терминам.   В основе метода латентно-семантического анализа лежат принципы факторного анализа, в частности выявление латентных связей изучаемых явлений или объектов. При классификации/кластеризации документов этот метод используется для извлечения контекстно-зависимых значений лексических единиц при помощи статистической обработки больших корпусов текстов.   1. Kullback–Leiblersummarizator - метод основанный на эвристическом добавлении предложений к имеющемуся саммари; в качестве критерия добавления/оценки текущего качества summary используется мера расстояние Кульбака — Лейблера которая считается для summary и исходного текста. Таким образом, суть метода состоит в нахождении набора предложений, которые наилучшим образом аппроксимирует исходный тест по КЛ-расстоянию. 2. LexRank - метод, схожий с TextRank, но конструктивной особенностью которого является его применимость к целому набору исходных текстов с одной темой.   **3. Проведен сравнительный анализ указанных методов на основе собранных текстов.**  В качестве метрики использовались значения ROUGE-1, ROUGE-2, ROUGE-3, усредненные для всей коллекции документов.  **Заметим, что большее значение метрики соответствует лучшему качеству построенного summary.**    Из графиков видно, что для данного набора текстов и методов, при ограничении длины summary в 15 предложений, метод LSA в среднем формирует более качественные краткие изложения исходных текстов. |
| **Нерешенные подзадачи** | 1. Улучшение TextRank метода путем изменения метрики для подсчета семантической близости токенов (см. <https://arxiv.org/pdf/1602.03606v1.pdf>)  2. Использование Sequence-to-Sequence модели и других DL-based моделей  3. Объединение модуля с модулями других подгрупп |
| **Source code** | <https://github.com/artezio-kseniav/text_summarization/tree/master/summary_v1> |
| **Вывод** | Для построения качественного саммаризатора необходимо использование DL-based моделей, обучение которых требует больших вычислительных мощностей и временных ресурсов. Однако, применение наивных методов позволяет построить модели, которые обладают необходимым качеством, чтобы выступать в качестве baseline-решения. |

**3 Построение summary с использованием модуля Gensim и извлечение keywords из статьи**

|  |  |
| --- | --- |
| **Задача группы** | * Построить качественное summary с использованием модуля Gensim * Извлечь keywords из статьи |
| **Исполнители** | * Кузнецов Владимир * Вороная Ксения * Куренков Евгений |
| **Используемые библиотеки и программные средства, данные** | * python3 * Anaconda - the leading open data science platform powered by Python. * Jupyter Notebook - a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. * Библиотеки python: genism, nltk, lxml * Источники данных:   Extract articles from on of the trusted websites  <http://www.psychiatrictimes.com/> |
| **Решенные подзадачи** | 1. Произвели парсинг одного из предложенных вебсайтов с целью извлечения топ-10 самых последних статей по шизофрении:  <http://www.psychiatrictimes.com/>  Добавили эти статьи в небольшой корпус.  2. Произвели предобработку текста для извлечения keywords, а именно удалили стоп-слова и сделали лемматизацию.  3. Извлечение keywords  Приводим пример для статьи:  Adjunctive Topiramate in People With Schizophrenia  The authors are: Brian Miller, MD, PhD, MPH  The date of publication is: September 29, 2016  C:\Users\Ksenia\Desktop\shiso.png  4. Построение summary  (пример для той же статьи см. под таблицей)  Gensim использует алгоритм TextRank с метрикой BM25.  TextRank каждому предложению присваивает метрику называемую прочностью соединения, которая ставится в соответствие количеству слов в предложении. Прочность соединения вычисляется по BM25 алгоритму и находится по следующей формуле |
|  | Где Q — предложение, состоящее из слов q\_1..q\_n.  D — документ (в нашем случае текст статьи).  f(q\_i, D) — частота слова q\_i в документе  |D| - длина документа (количество слов в нем).  Avgdl — средняя длина документа.  k\_1 и b — свободные коэффициенты (приблизительно равны 2.0 и 0.75 соотвественно).  IDF(q\_i) — обратная документная частота для слова q\_i.  Таким образом, имеем граф, узлы которого соответствуют прочности соединения для каждого предложения и чем выше данная оценка, тем более ценным считается предложение.  В функции gensim.summarize() присутствует параметр ratio, который устанавливает степень сжатия статьи. Чем выше данные параметр, тем более подробное изложение получается и тем больше времени необходимо для выполнения.  Также в программе присутствует функция keywords для подсчета и вывода наиболее популярных слов в файле.  На вход программе передается документ, каждая строчка которого содержит тело статьи, для которой необходимо сделать саммари. На выходе получается два файла: summary — каждая строчка которого содержит краткое изложение статьи и keywords - каждая строчка которого содержит наиболее значимые слова в документе.  Для оценки качества саммаризации используется метрика ROUGE-n (Recall-Oriented Understudy for Gisting Evaluation, http://anthology.aclweb.org/W/W04/W04-1013.pdf), где n- это количество последовательных слов, используемых для оценки. Данная возможность не была имплементирована в данной программе и планируется к реализации в дальнейшей разработке. |
| **Нерешенные подзадачи** | 1. Улучшение процесса извлечения key-words  2. Объединение модуля с модулями других подгрупп |
| **Source code** | <https://github.com/artezio-kseniav/text_summarization/tree/master/summary_version2>  <https://github.com/artezio-kseniav/text_summarization/tree/master/text_summarization> |

****

**4. Исследование возможностей по созданию тематической модели на основе научных работ**

**Данное исследование провел Артем Соколов.**

Изначально было запланировано что поиск и анализ публикаций будет основываться на тематическом моделировании. Это сравнительно новое направление в машинном обучении подразумевает что поиск будет выполняться на основе анализа документа вцелом, т.е. с учетом текста, ссылок, авторов. В нашей группе мы разделились так что 4 человека отвечали непосредственно за исследование возможностей по созданию тематической модели на основе научных работ. 2 человека искали и исследовали возможности языка R и два человека делали то же самое для языка python с целью дальнейшего сравнения.

Я принимал участие в работе второй группы. Мы нашли очень перспективную библиотеку для создания и работе с тематическими моделями BigARTM, написанной на языке c++, но имеющей, также, API для языка python (http://bigartm.org/). Коллегами была найдена и выложена большая выборка научных статей на медицинские темы. Мы обработали перевели эти документы в формат vowpal wabbit, поддерживаемый BigARTM и далее, разделили наши усилия.

Коллега занимался исследованием возможностей модели в направлении мультимодальности (т.е. учитывания не только текста, но и ссылок и авторов для нахождения релевантных), а я занимался построением иерархической тематической модели. Такая модель помимо матриц документы-темы и темы-слова содержит матрицу темы-подтемы для каждых соседних уровней иерархии. ([www.machinelearning.ru/](http://www.machinelearning.ru/)). Как я выяснил, BigARTM иммет интерфейс для построения таких моделей, но не в основном, стабильном бранче. Я написал код на питоне для того что бы построить трёхуровневую модель на основе текстов документов. Подбирались параметры и регуляризаторы для того что бы сделать матрицы разреженными (см. <http://bigartm.org/> для дополнительной информации о принципах работы BigARTM). Количество тем и подтем подбиралось экспериментально. Какой-то абсолютной числовой характеристики, по которой можно было бы судить о чистоте тем найдено не было. Я приведу здесь примеры для двух уровней, при этом я уверен что, полученные результаты еще возможно значительно улучшить.



Таблица тем Уровень 1



Таблица части тем Уровень 2

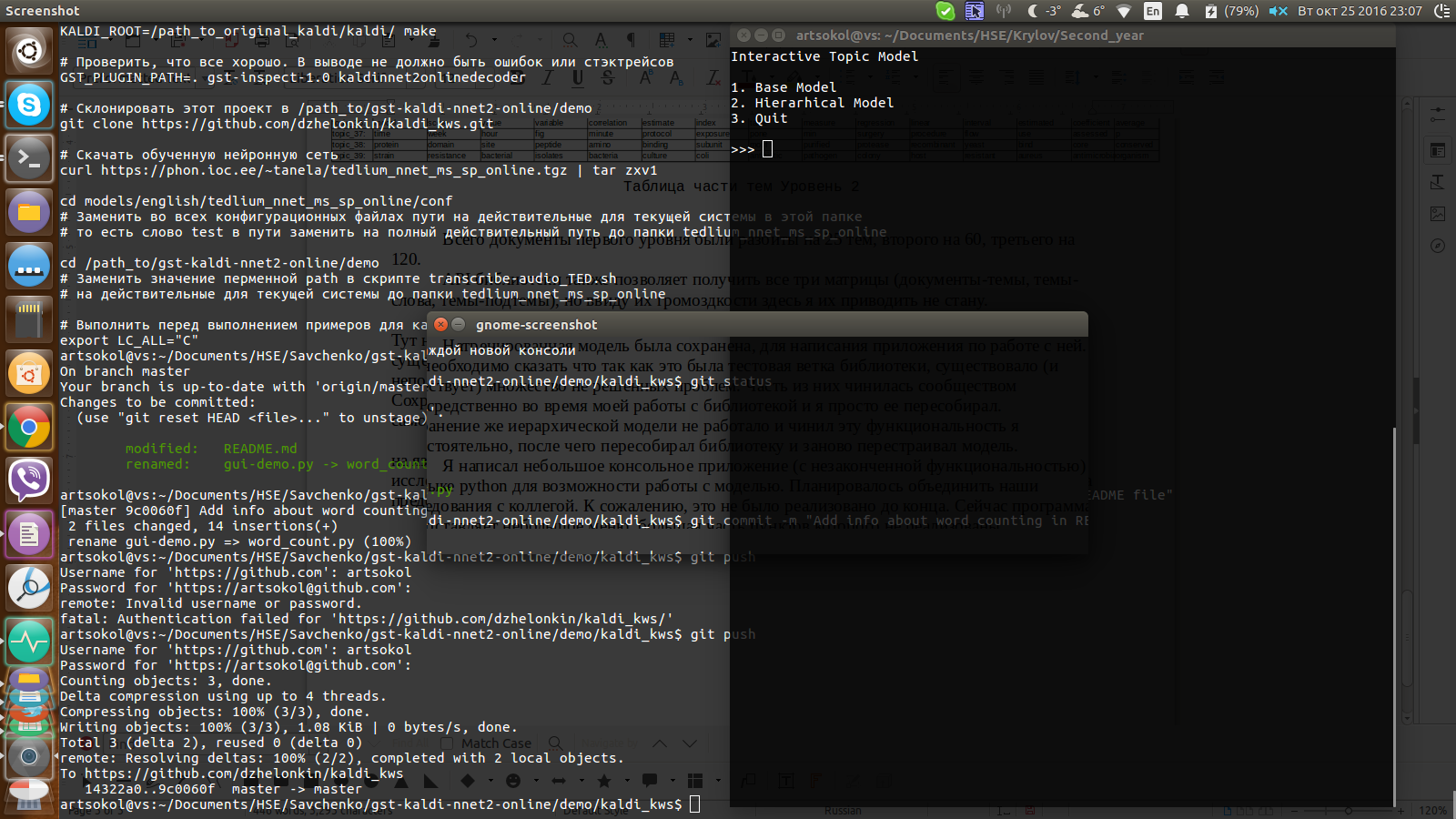
Всего документы первого уровня были разбиты на 25 тем, второго на 60, третьего на 120.

API библиотеки также позволяет получить все три матрицы (документы-темы, темы-слова, темы-подтемы), но ввиду их громоздкости здесь я их приводить не стану.

Натренированная модель была сохранена, для написания приложения по работе с ней.

Тут необходимо сказать что так как это была тестовая ветка библиотеки, существовало (и существует) множество не решенных проблем. Часть из них чинилась сообществом непосредственно во время моей работы с библиотекой и я просто ее пересобирал. Сохранение же иерархической модели не работало и чинил эту функциональность я самостоятельно, после чего пересобирал библиотеку и заново перестраивал модель.

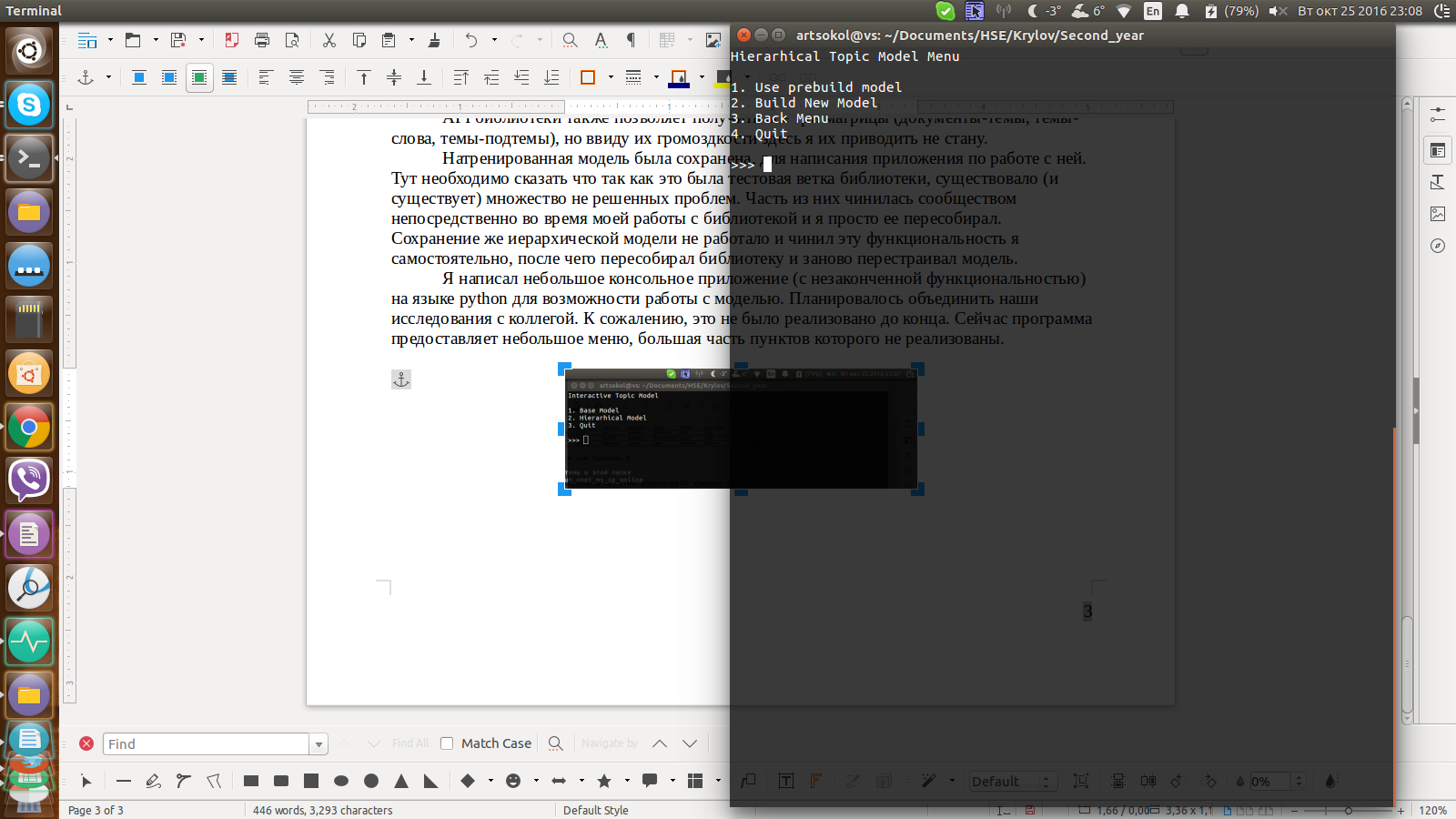
Я написал небольшое консольное приложение (с незаконченной функциональностью) на языке python и с BigARTM API идля возможности работы с моделью. Планировалось объединить наши исследования с коллегой. К сожалению, это не было реализовано до конца. Сейчас программа предоставляет небольшое меню, большая часть пунктов которого не реализованы.



Верхний уровень меню

**Base model** — уровни меню для работы с обычной, не иерархической моделью(не реализована).

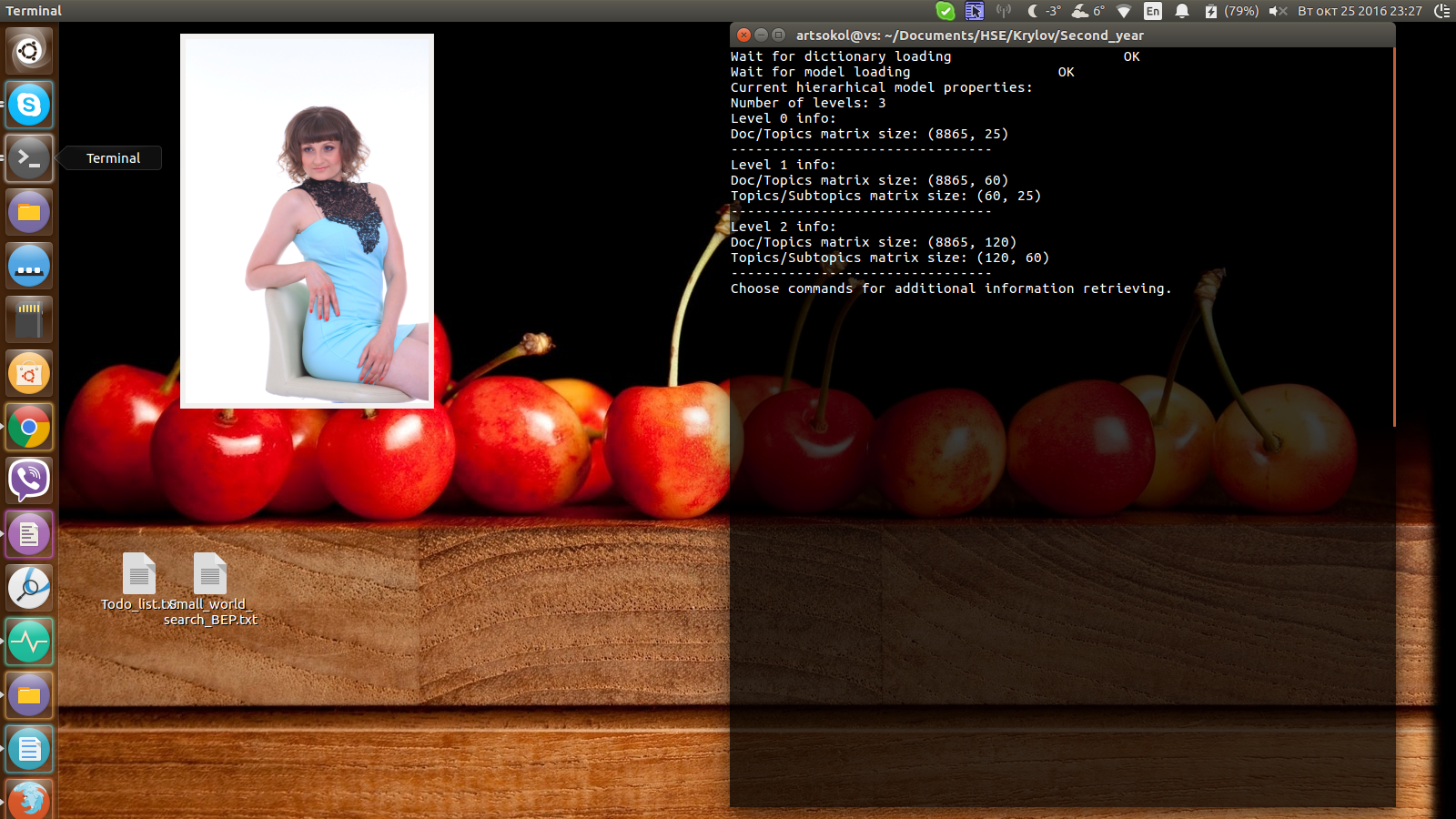
**Hierarhical Model** — работа с иерархической моделью.



Второй уровень меню

**Use prebuild model** — загрузка построенной модели с возможностью дальнейшего получения информации.

**Build New Model** — задание параметров для построения и сохранения новой модели.



Краткие данные о загруженной модели

Следующим этапом планировалось предоставить терминал пользователю для запросов информации, матриц, коэффициентов модели в целом и отдельных уровней.