Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____20__г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., доцент

_____
«____»_____2018 г.

Разработка и исследование алгоритма поиска документов по запросу на основе тематической модели

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–010400.2018.105.ПЗ ВКР

Руководитель работы, доцент кафедры прикладной математики и программирования, к.т.н.
_____/Т.Ю. Оленчикова
«____»_____2018 г.

Автор работы
Студент группы ММиКН-414
_____/С.Ю. Тепляков
«____»_____2018 г.

Нормоконтролер, доцент кафедры прикладной математики и программирования, к.т.н.
_____/Т.Ю. Оленчикова
«____»_____2018 г.

Челябинск 2018

АННОТАЦИЯ

Тепляков С.Ю. Разработка и исследование алгоритма поиска документов запросу на основе тематической модели.– Челябинск: ЮУрГУ, ЕТ-414, 46 с., 13 ил., библиогр. список – 18 наим., 2 прил.

В работе исследуются методы построения тематических моделей и реализация разведочного информационного поиска с их использованием. Решается задача повышения релевантности поисковой выдачи за счет учета предпочтений пользователя. В качестве решения задачи предлагаются два метода – коррекция вектора тем запроса и предсказание релевантности с помощью логистической регрессионной модели.

Написана программа для построения тематической модели на основе подхода аддитивной регуляризации, разработан программный модуль, реализующий построенные алгоритмы поиска и учета предпочтений пользователя. На примере конкретных запросов проведено экспериментальное исследование описанных методов.

Полученные алгоритмы значительно повышают релевантность поиска, а реализующий их программный модуль может быть подключен к другим задачам и проектам.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В условиях все большего увеличения объемов информации и ускорения темпов ее прироста, все более актуальной становится задача поиска новых способов обработки, анализа и хранений данных. Одним из возможных решений выступает тематическое моделирование – современный статистический аппарат анализа текстов, активно развивающийся с конца 90-х годов.

Задача тематического моделирования заключается в построении для некоторой коллекции документов тематической модели, выявляющей тематику коллекции. Каждая тема в такой модели представляется в виде дискретного распределения вероятностей на множестве слов, а каждый отдельный документ – дискретным распределением вероятностей на множестве тем.

Данная задача имеет некоторое сходство с задачей кластеризации документов. Однако при кластеризации каждый документ целиком соотносится с одним конкретным кластером, в то время как в тематической модели осуществляется мягкая кластеризация (альтернативно, нечеткая кластеризация), допускающая принадлежность документа нескольким кластерам-темам. Такой подход позволяет обойти проблему многозначности и синонимии слов.

К числу конкретных приложений тематического моделирования можно отнести: поиск, классификацию и кластеризацию документов, определение текущих трендов в потоке новостей и различных областях знаний, выявление спама, многоязычный информационный поиск [1].

Возможны приложения и за рамками текстовых в сфере анализа видео и изображений, рекомендательных системах, электрокардиографии и биоинформатике.

Отдельно выделяется применение к задаче реализации системы «разведочного поиска». В стандартных поисковых системах поиск осуществляется по короткому запросу из набора ключевых слов, что предполагает наличие у пользователя достаточного конкретного представления о том, что он желает найти. Разведочный поиск же в качестве запроса может принимать отрывок текста, документ целиком или коллекцию документов, а в качестве результата должен выдавать наиболее близкие по тематике и актуальные на сегодняшний день данные.

Предполагается, что такой поиск мог бы быть особенно полезен для научной, исследовательской и инженерной сфер, где часто возникает необходимость получить наиболее релевантную информацию по какой-либо заданной теме.

Одной из проблем, с которой приходится сталкиваться при реализации подобной системы поиска является персонализация поисковой выдачи под нужды пользователя. В общем случае единственным рычагом контроля над результатом поиска выступает только сам документ-запрос, однако далеко не во всех случаях такой документ идеально соответствует тематикам, которые пользователь хотел бы получить в результате поиска. Как следствие, важную роль обретает наличие у пользователя способов коррекции выдачи и ее подстройка под некие дополнительные критерии.

Задача тематического моделирования в общем случае является некорректно поставленной и обладает бесконечным множеством решений. Наиболее популярные алгоритмы построения тематических моделей PLSA (Probabilistic latent semantic analysis) [2] и LDA (Latent Dirichlet Allocation) [3] на выходе выдают одно из этих решений. Аддитивная регуляризация тематических моделей ARTM позволяет при помощи регуляризаторов внести в процесс построения дополнительные критерии и посредством их максимизации сделать выбор решения более обоснованным.

Работа посвящена разработке программного модуля, реализующего алгоритмы поиска документов по тематической модели и улучшения релевантности поисковой выдачи посредством учета предпочтений пользователя.

Первый раздел посвящен тематическому моделированию. Описаны общие принципы и методы построения тематических моделей, сделан обзор существующих пакетов ПО, реализующих эти методы. На основе проведенного обзора принято решение использовать подход ARTM и реализующую его библиотеку BigARTM. Также была сформулирована постановка задачи для данной работы.

Во втором разделе приводятся разработанные алгоритмы предобработки исходных данных, поиска документов по тематической модели и два метода учета предпочтений пользователя – коррекция тем в запросе и регрессионное прогнозирование релевантности документов.

Третий раздел посвящен разработке архитектуры программного модуля и его описанию, приводятся диаграмма компонентов и диаграмма классов.

В четвертом разделе описывается методика экспериментального исследования эффективности разработанных алгоритмов и приводятся его результаты.

1. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

1.1 Математическая модель тематического моделирования

1.1.1 Терминология и базовые предположения

Исходными данными для тематического моделирования является множество (коллекция) текстовых документов *D* и множество (словарь) всех употребляемых в низ термов *W*. Термами могут выступать слова, нормальные формы слов, или определенные термины и словосочетания (граммы) – это зависит от того, как и какими методами осуществилась предварительная обработка исходных текстов. Каждый документ *d* из множества всех документов представляется вектором слов (термов) ( ) из словаря *W*, где *n* – это длина документа. Каждый термин может повторяться в документе несколько раз.

Под термином «тема» подразумевается множество признаков, принадлежащих одной и той же предметной области. При этом каждый документ может принадлежать более чем одной теме, а каждый признак, в свою очередь, может принадлежать произвольному числу тем.

При решении задачи построения тематической модели выдвигается ряд гипотез.

1. Гипотезой о существовании тем называется предположение о том, что существует некоторое конечное множество тем T, и каждое вхождение в документ d терма w связанно с некоторой заранее неизвестной темой , принадлежащей T. Коллекция документов тогда представляет собой последовательность троек (w_i, d_i, t_i), i = 1, … n из дискретного распределения p(w, d, t) на конечном множестве (W × D × T);

Термы w_i и документы d_i являются наблюдаемыми переменными, а темы t_i являются латентными (скрытыми) переменными.

2. Гипотеза мешка слов, также известная как предположение о независимости, гласит, что тематику документа можно определить, зная лишь то, какие термы принадлежат каким документам, не имея информации о порядке расположения термов в документах. В других словах, даже после произвольной перестановки термов в документе, его тематику по-прежнему можно выявить, даже если такое представление текста теряет смысл для человека. В формальной интерпретации получается, что каждый отдельный документ рассматривается как выборка случайно и независимо порождаемых терминов.

Данное предположение позволяет перейти к такому представлению документов, где каждый d является подмножеством W, в котором каждому элементу поставлено в соответствие некоторое число n_{dw}, обозначающее число вхождений термина w в документ.

Порядок документов в коллекции также не имеет значения (гипотеза мешка документов);

3. Согласно гипотезе разреженности, каждое слово w и каждый текст d должны быть связаны с небольшим количеством тем, то есть значительная часть вероятностей p*(t|d)* и *p(w|t)* будут либо обращены в нуль, либо очень близки к нулю. Разреженность является обязательным условием для больших коллекций, так как алгоритмы, в которых отсутствует необходимости хранить нулевые значения, работают намного быстрее и позволяют экономить память.

Однако недостаток такого обнуления заключатся в том, что при и перплексия модели уходит в бесконечность, что существенно затрудняет оценку качества модели.

Если документ принадлежит большому множеству различных тем, его следует разделить на некоторое число отдельных фрагментов, где в каждом фрагменте можно выделить меньшее число тем.

Если то же самое верно для какого-либо терма, то можно говорить о принадлежности этого терма к числу общеупотребимых. В таком случае данный терм стоит исключить из выборки документов, так как общеупотребимые термы зачастую понижают интерпретируемость модели;

4. Гипотеза условной независимости задается распределением и предполагает, что появление термов из темы t в документе d зависит только от темы и не зависит от самого документа. Согласно формуле полной вероятности и гипотезе условной независимости тематическая модель коллекции имеет следующий вид:

(1.1)

Вероятностная модель (1.1) описывает порождение коллекции по известным .

Построение тематической модели является обратной задачей: по заданной коллекции *D* необходимо найти породившие её дискретные распределения и .

В общем случае число тем |T| значительно меньше |D| и |W|. Благодаря этому задачу построения тематической модели допустимо рассматривать как задачу поиска приближённого матричного разложения заданной матрицы частот. Равенство (1.1) тогда можно переписать в матричном виде произведения , где в левой части находится заданная матрица частоты термов в документах , , $i=1,\ldots, |W|$, $j=1,\ldots, |D|$, а правая представлена произведением матрицы термов и матрицы тем

,

Все три матрицы являются стохастическими.

1.1.2 Принцип максимума правдоподобия

В математической статистике для оценки неизвестных параметров вероятностной модели по наблюдаемым данным используется принцип

максимума правдоподобия. Согласно этому принципу, выбираются такие значения параметров, которые обеспечивают наибольшую правдоподобность наблюдаемой выборки.

Для нахождения параметров модели Φ, Θ выполняется максимизация функции правдоподобия, определяемой как зависимость вероятности выборки от параметров модели:

Переход от произведения к сумме осуществляется с помощью логарифмирования, после отбрасываются слагаемые, не зависящие от параметров модели. В результате получается задача максимизации log-правдоподобия:

(1.2)

При этом устанавливаются ограничения неотрицательности
и нормировки

1.1.3 Частотные оценки условных вероятностей

В пространстве (W × D × T) вероятности, связанные с переменными *d* и *w*, можно оценивать по выборке как частоты (будем обозначать частотные оценки *p* через :

—, —, —, — (1.3)

– длина документа ;
– число вхождений терма в документ ;
– число вхождений терма *w* во все документы коллекции;
– длина коллекции.

Вероятности, выражающиеся через скрытую переменную *t*, аналогично можно оценивать как частоты, если рассматривать коллекцию документов как выборку троек (*d*,*w*,*t*):

—, —, —, — (1.4)

– число троек, связанных с темой *t*.
- число троек, в которых терм *w* документа *d* относится к теме *t*;
- число троек, в которых терм *w* связан с темой *t*;
– число троек, в которых терм *w* документа *d* относится к теме *t*;

При частотные оценки (1.3)-(1.4) стремятся к соответствующим вероятностям *p* в соответствии с законом больших чисел.

1.1.4 Перплексия в тематических моделях

Для оценки качества работы алгоритмов построения тематических моделей в общем случае наиболее часто используется перплексия:

– (1.5)

Для ее расчета обычно используется некоторая выборка контрольных данных D`, не входящих в обучающую выборку. Каждый документ d D' произвольно делится на две части d' и d''. Оценка параметров осуществляется по обучающей выборке D, параметры оценивается по d'. Перплексия же подсчитывается по d'' из контрольной выборки [4].

В тематическом моделировании численное значение перплексии интерпретируется следующим образом. Если каждый документ генерируется из множества всех термов W, то перплексия сходится к его мощности |W|.

Таким образом, перплексия является зависимой от размера словаря и распределения частоты слов в коллекции , из чего следует ряд ее недостатков [5]:

- невозможность проведения сравнения между моделями с различной n-граммностью;
- невозможность сравнения различных методов разреживания словаря;
- невозможность оценки качества удаления стоп-слов и слов, не участвующих в формировании тем.

Важной характеристикой тематической модели является способность к обобщению. Говорят, что модель переобучается, если перплексия контрольной выборки существенно превышает перплексию обучающей выборки. На данный момент не существует адекватных теоретических оценок переобучения вероятностных тематических моделей, в силу чего их качество принято оценивать и сравнивать эмпирически.

1.1.5 Существующие подходы к тематическому моделированию

1.1.5.1 Латентно-семантический анализ LSA

Также известный в области информационного поиска как латентно-семантическое индексирование (LSI), данный метод был запатентован в 1988 году американской командой инженеров-исследователей.

LSA представляет собой подход к выявлению латентных ассоциативно-семантических связей между коллекцией документов и составляющими ее термами посредством сокращения факторного пространства термы-на-документы. Метод работает с векторами слов.

Исходной информацией для LSA выступает матрица термы-на-документы, где элементами матрицы выступают веса, отображающее частоты использования термов в документах. Выявление связей между термами выполняется с помощью разложения этой матрицы по сингулярным значениям во множество ортогональных матриц, линейную комбинацию которых можно рассматривать как достаточно точное приближение к исходной матрице.

В соответствии с теоремой о сингулярном разложении [6], вещественная прямоугольная матрица может быть разложена на произведение , где U и V – ортогональные матрицы, а матрица S – диагональная. Значения на диагонали матрицы S называются сингулярными значениями матрицы A.

Если после разложения в матрице S оставить только k наибольших сингулярных значений, а в матрицах U и V соответствующие им столбцы, то линейная комбинация этих трех матриц даст наилучшее приближение матрицы A к матрице ранга k. Полученная матрица отражает структуру латентных зависимостей исходной матрицы.

Каждый термин в каждом документе, таким образом, получает представление в виде векторов в общем семантическом пространстве размерности k. Через скалярное произведение векторов можно с легкостью узнать близость между любой комбинацией термов и/или документов.

Выбор размерности k, как правило, зависит от специфики решаемой задачи и подбирается эмпирически. Чрезмерно малое k ухудшает способность модели к выявлению различий между схожими терминами и документами. Если же k слишком большое, то модель теряет мощность и приближается по характеристикам к стандартным векторным методам.

В числе возможных приложений LSA выделяют:

- сравнение двух документов между собой;
- сравнение двух термов между собой;
- сравнение терма и документа.

По сравнению с обычными статистическими методами LSA обладает отличной способностью выявлять зависимости между словами. Другим выгодным преимуществом этого подхода выступает возможность расширения модели при помощи ввода в нее новых переменных и новых связей между наблюдаемыми и скрытыми переменными, а также отсутствие необходимости повторного обучения модели при добавлении нового сегмента входных данных.

Однако большим недостатком данного метода является существенно снижение скорости вычисления при увеличении объема входных данных [7]. Кроме того, будучи основанной на нормальном распределении, вероятностная модель метода не совсем соответствует реальности.

1.1.5.2 Вероятностные тематические модели

Вероятностное тематическое моделирование представляет собой набор алгоритмов, позволяющих анализировать слова в больших наборах документов и извлекать из них темы, связи между темами и изменение этих тем во времени [8].

Документы исходной выборки представляется в виде мешка слов, порядок которых не имеет значения. Каждый документ определяется распределением составляющих его слов по множеству тем , то есть для каждой темы находится величина, характеризующая вероятность присутствия данной темы в данном документе.

Темы представляются в виде распределения на множестве словаря, то есть для каждого слова задается вероятность его принадлежности данной теме.

Вероятностные модели являются генеративными, то есть могут быть использованы для генерации документов. Но главная цель тематического моделирования заключается в извлечении тем из уже имеющегося набора документов, а не генерация. Таким образом, получаем задачу, обратную генерации: выявить наиболее вероятные скрытые структуры (темы и их распределение), с помощью которых мог быть сгенерирован исходный набор данных.

1.1.5.3 Вероятностный латентно-семантический анализ PLSA

PLSA (Probabilistic Latent Semantic Analysis) является одним из распространённых инструментов вероятностного тематического моделирования. Впервые метод был предложен Томасом Хоффманом в 1999 году в работе [2]. За основу метод берет аспектную модель (aspect model), связывающую между собой латентные (срытые) переменные тем и наблюдаемые переменные термов w_i и документов d_i.

Задача определяется как поиск и выявление этих латентный переменных. Каждый документ при этом представлен в виде числового вектора и может относиться ко множеству разных тем с некоторой вероятностью для каждой из них. В этом состоит основная особенность PLSA, отличающая ее от методов, основанных не на вероятностном моделировании.

После задания определенного количества тем в качестве желаемого параметра модели метод PLSA позволяет оценить ряд величин:

- – вероятность того, что слово будет находиться в случайно выбранном из коллекции документе ;
- – вероятность того, что документ это наиболее тесно связанный с темой документ;
- – вероятность того, что слово наиболее тесно связанно с темой .

Вероятностную модель порождения документов возможно описать следующим образом.

1. Случайным образом выбирается документ с вероятностью .
2. Случайным образом выбирается тема с вероятностью .
3. Случайным образом выбирается терм с вероятностью .В результате получаем пару ( ). Вероятностная модель тогда задается следующим образом:

Отдельно стоит отменить, что модель также можно записать двумя другими способами:

где и являются распределением тем и слов по всей коллекции документов соответственно.

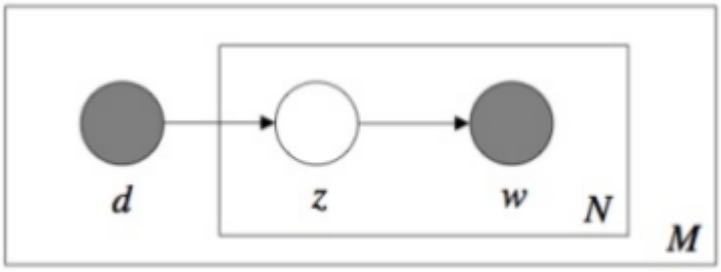


Рисунок 1.1 – Модель PLSA

Вершины графа на рисунке соответствуют переменным, а соединяющие их ребра – вероятностным взаимоотношениям между ними. Закрашенные вершины обозначают наблюдаемые переменные, не закрашенная вершина – латентная переменная тем. M является множеством всех документов, а N – множеством слов в каждом отдельном документе. Направленность ребер означает отношение условной зависимости вершин, на которые указывают ребра, от вершин, из которых ребра исходят.

PLSA обладает более гораздо более лучшим статистическим обоснованием, нежели LSA, основанный на методах линейной алгебры.

К недостаткам PLSA относят:

- склонность к переобучению при работе с большой коллекций текстов в виду линейного возрастания числа параметров и ;
- необходимость перестройки всей модели при добавлении в исходную коллекцию нового документа;
- значительное снижение скорости сходимость модели на больших коллекциях документов, так как обновляются после каждого прохода коллекции;

- алгоритм не предусматривает специальной обработки общеупотребительных слов. Как результат, снижается интерпретируемость выделенных моделью тем;
- PLSA не позволяет управлять ни разреженностью ни разреженностью ;
- необходимость хранить трехмерную матрицу .

1.1.5.4 Байесовская регуляризация

В байесовском подходе используется принцип максимума апостериорной вероятности (maximuma posteriori probability; MAP), в рамках которого предполагается, что параметры модели являются случайными, но подчиняются некоторому априорному распределению где – неслучайный гиперпараметр. Формула максимизация совместного правдоподобия в этом случае обретает следующий вид:

После логарифмирования получается модификация задачи (1.2), с логарифмом априорного распределения в роли регуляризатора:

Для снижения размерности задачи и риска переобучения в байесовском подходе может применяться принцип неполного правдоподобия, в котором гиперпараметры оптимизируются с помощью интегрирования параметров . Это возможно в силу того, что размерность вектора не зависит от объема исходной коллекции и, чаще всего, много меньше матриц

В байесовском подходе оцениваются не сами параметры а их апостериорное распределение . Но в практических приложения в этом нет необходимости, так как полученное распределение нужно только для того, чтобы затем вернуться к точечным оценкам математического ожидания.

Несмотря на доминирование байесовского подхода в литературе по тематическому моделированию, он представляет скорее академический интерес, нежели практический метод.

Объясняется это тем, что в байесовском выводе попытки комбинирования моделей и добавления дополнительных критериев влекут за собой необходимость повторного проведения всех математических выкладок и в каждом случае требуют новой программной реализации, что критически сказывается на сроках и стоимости таких разработок.

Байесовские модели также чрезвычайно сложны для понимания, вывода и сравнения, отсутствует возможность их комбинирования и взаимной замены. При этом модели, опирающиеся на байесовский вывод, часто не обладают

достаточными лингвистическими обоснованиями и подвержены проблемам неединственности и неустойчивости.

В силу этих причин в практических приложениях выбор метода построения тематической модели, как правило, происходит в пользу других, более простых моделей.

1.1.5.5 Латентное размещение Дирихле

LDA (Latent Dirichlet Allocation) – генеративная вероятностная модель, предложенная Дэвидом Блеем, Эндрю ЫНом и Майклом Джорданом в 2003 году для решения проблемы переобучения PLSA, которая предсказывала вероятности распределения слов на новых заметно хуже, чем на обучающей выборке. На сегодняшний день LDA считается доминирующим методом в области вероятностного тематического моделирования и имеет огромное множество модифицированных вариантов, специализированных на решение более конкретных задач.

Процесс порождения документов в данной модели происходит независимо для каждого документа и схож с генеративным процессом из PLSA (рисунок 1.2):

1) случайным образом для документа выбирается его распределение по темам ;

2) для каждого слова в документе:

a) случайным образом выбирается тема из распределения , полученного на 1-м шаге;

b) случайным образом выбирается слово из распределения слов в выбранной теме .

Исходные данные состоят из набора D документов, каждый из которых содержит слов. Слова являются единственными наблюдаемыми переменными в модели, все остальные переменные являются скрытыми. Переменная характеризует значение темы с шага 2a по отношению слову , а задает распределение тем для каждого документа .

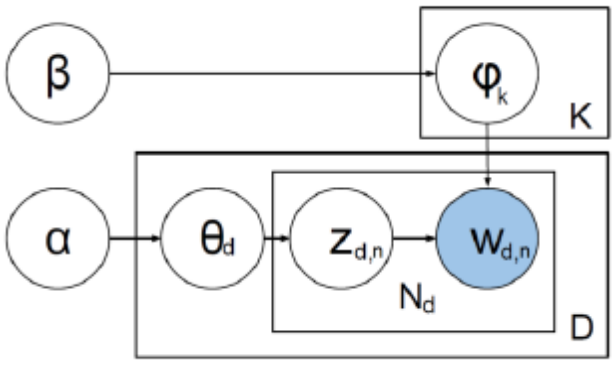


Рисунок 1.2 – Модель LDA

K выступает фиксированным входным параметром модели и равняется количеству тем. Оптимальное K обычно подбирают эмпирическим способом, но

существуют и подходы к автоматическому определению оптимального количества тем [9]. Переменная задает распределение слов в теме k.

LDA тоже основана на разложении (1.1), но с дополнительным предположением, что столбы матриц и являются случайными векторами, порожденными распределением Дирихле с параметрами и соответственно:

_____

_____

где – гамма-функция, а векторы и названются гиперпараметрами.

Распределение Дирихле способно порождать как плотные, так и разреженные векторы распределений. С уменьшением растет разреженность w компонента в порождаемых векторах .

В процессе порождения данных из распределения Дирихле сперва генерируются вектора , задающие темы. После чего из распределений генерируются слова, формирующие тематические части документов . В результате получаем описание кластерных структур на коллекции текстов, где задают кластерные центры, а распределения описывают точки этих кластеров.

Главными отличием LDA от PLSA являются оценки условных вероятностей (1.4):

- В PLSA – несмещенные оценки максимума правдоподобия:

___ ___

- В LDA – сглаженные байесовские оценки:

_____ _____

Недостатки распределения Дирихле:

- отсутствие убедительных лингвистических обоснований;
- байесовский вывод требует интегрирования по пространству параметров модели, которое элементарно только в простейшем варианте LDA;
- попытки построения многоцелевых комбинированных моделей приводят к громоздким математическим выкладкам;
- практические исследования показывают, что на больших коллекциях правдоподобие LDA и PLSA отличается незначительно [10].

Практическая распространенность метода в области задач тематического моделирования объясняется скорее математическим удобством и популярностью байесовского обучения. Байесовский вывод в данной значительно упрощен

благодаря сопряженности распределения Дирихле к мульти-номинальному распределению.

1.1.5.6 EM-алгоритм

Для решения оптимизационной задачи (1.2) в PLSA применяют итерационный алгоритм, в котором каждая итерация состоит их двух шагов – E (expectation) и M (maximization) [11].

Перед первой итерацией осуществляется выбор начального приближения параметров и . В простейшем случае их можно задать нормированными случайными векторами из равномерного распределения.

На E-шаге с помощью текущих и и формулы Байоса вычисляются условные вероятности всех тем для каждого терма в каждом документе :

На M-шаге по условным вероятностям тем вычисляется новое приближение параметров , при помощи величины

Суммирование по документам и по термам дает оценки , , , посредством которых, в соответствии с (1.4), можно получить частотные оценки условных вероятностей , :

—, , . (1.6)

—, , (1.7)

1.1.6 Аддитивная регуляризация тематических моделей

ARTM (AdditiveRegularizationofTopicModels) – это подход к построению вероятностных тематических моделей, охватывающий PLSA, LDA и многие байесовские модели. ARTM позиционируют как альтернату байесовскому обучению тематических моделей [12].

Задачу называют корректно поставленной по Адамару в случае, если ее решение существует, единственно и устойчиво.

Однако стохастическое матричное разложение не единственно и определенно с точностью до невырожденного преобразования
(где матрицы и тоже стохастические), что означает наличие у него, в общем случае, бесконечного множества возможных решений. Таким образом, задача построения тематической модели является некорректно поставленной.

Для решения этой проблемы применяют регуляризацию, представляющую собой общий подход для решения обратных задач с некорректной постановкой

[13]. В рамках данного подхода к основному критерию правдоподобия (1.2) добавляется дополнительный критерий, называемый регуляризатором:

Переменная здесь выполняет роль коэффициента регуляризации и напрямую задает параметр, определяющий, насколько важен данный регулятор, и насколько сильным должно быть его влияние на конечную модель.

Как результат, при решении задачи максимизации правдоподобия будет также осуществляется и максимизация нового критерия, что позволяет учитывать специфику решаемой задачи и получать более релевантные модели.

На практике в задачах обработки текстов зачастую используются сразу несколько регуляризаторов, каждый из которых накладывает свое ограничите на решение.

Подход ARTM обладает множеством преимуществ над типичным байесовским обучением и графическими моделями, в рамках которых, главным образом, до сих пор развивалось вероятностное тематическое моделирование. В их числе можно выделить:

- относительная легкость понимания и вывода моделей, снижающая порог входа в область тематического моделирования для исследователей их смежных областей;
- очень простой математический аппарат – для добавления регуляризатора достаточно лишь добавить его производные в формулы М-шага (1.6)-(1.7);
- регулязаторам в ARTM необязательно иметь вероятностную интерпретацию или быть априорными распределениями;
- регулязатор Диририхле утрачивает свою особую роль, исчезает необходимость использовать его в каждой модели для всех тем;
- легкость построения многоцелевых комбинированиях моделей посредством суммирования регуляторов, взятых из разных моделей.

1.1.6.1 Регуляризатор сглаживания

Данный регуляризор формализует требование о том, чтобы слова общей лексики, стоп-слова и специфические редко встречающиеся слова собрались в специальные фоновые темы и не засоряли предметные темы, что положительно сказывается на их интерпретируемости.

С его помощью осуществляется минимизация дивергенции Кульбака-Лейблера между распределениями и и дискретными распределениями слов во всей коллекции и тем по всей коллекции соответсвенно:

где – коэффициенты регуляризации.

Формулы М-шага тогда приобретают вид:

1.1.6.2 Регуляризатор разреживания

Естественным является предположение, каждый документ описывается небольшим числом тем w из общего словаря коллекции W. Значение вероятностей таких тем в распределении в таком случае следует увеличивать, в то время вероятности тем, не характеризующих документ, следует занулять

Аналогично предполагается, что каждая тема определяется небольшим количеством слов, а вероятности остальных слов в распределении должны равняться нулю.

В случае с LDA осуществить зануление невозможно, поскольку распределение Дирихле не допускает нулевых значений в порождаемых им векторах

В ARTM таким результатов можно добиться добавлением сглаживающего регуляризатора. Таким образом, осуществляется максимизация дивергенции между равномерным распределением и искомыми распределениями:

где – коэффициенты регуляризации.

Формулы М-шага:

1.1.6.3 Регуляризатор сокращения незначимых тем

Формализует требование исключения из модели тем, ядро которых состоит из крайне малого количества редких слов. Регуляризатор вводит требование разреженности распределения тем во всей коллекции и максимизирует дивергенцию Кульбака-Лейблера между и равномерным распределением:

Формула М-шага принимает вид:

—

где – коэффициент регуляризации. В соответствии с этой формулой, вероятности темы t понижаются для всех документов, если число отнесенных к ней слов мало. Регулязатор способен оптимизировать количество тем, если их изначально заданное число было заведомо избыточным.

1.1.6.4 Регуляризатор декоррелирования

Формализует требование к предметным темам, согласно которому темы должны как можно сильнее отличаться друг от друга. С помощью этого регуляризотора осуществляется минимизация суммы ковариаций между распределениями и для всех пар тем *t,s:*

Формулы М-шага:

где – коэффициент регуляризации. Декоррелирование способствует разрежеванию тем и выделению в них ядер, состоящих слов с доминирующей величиной вероятности . Регулязатор также обладает способностью группировать стоп-слова в отдельные фоновые темы [14].

1.2 Существующие пакеты ПО для тематического моделирования

1.2.1 Gensim

Gensim представляет собой пакет инструментов с открытым исходным кодом для обработки коллекций текстов, реализованный в виде подключаемого модуля для языка программирования Python [15]. Проект берет свое начало с 2008 года, тогда он являлся небольшим набором скриптов для поиска похожих статей, разработанным для нужд Чешской Библиотеки Цифровой математики.

Gensim специально заточен под обработку больших объемов текстовых данных с помощью потоковых данных и эффективных инкрементальных алгоритмов. Проект позиционирует себя как один из самых робастных, эффективных и простых в использовании пакетов программного обеспечения для работы с задачами семантического моделирования текстов.

Gensim включает в себя, помимо прочего, реализации алгоритмов латентного семантического анализа, латентного распределения Дирихле и иерархического процесса Дирихле (Hierarchical Dirichlet Process, HDP). Для каждого из алгоритмов также реализованы его распределенные параллельные версии.

Пакет распространяется по лицензии GNU LGPLv2.1, допускающей свободное академическое и коммерческое использование.

1.1.2 Vowpal Wabbit (VW)

Является одной из наиболее широко используемых библиотек для работы с алгоритмами машинного обучения в индустрии. На текущий момент проект разрабатывается усилиями Microsoft Research, в прошлом был разработкой группы Yahoo! Research. [16]

Отличается, прежде всего, гибкостью в выборе формата входных данных, высокой скоростью работы, хорошей масштабируемостью, поддержкой большого количества различных режимов обучения и огромным множеством поддерживаемых алгоритмов, включая метод LDA.

Главной особенностью библиотеки считается возможность онлайн-обучения, что представляет особый интерес для задач, требующих обработки больших и высокоразмерных данных.

Проект реализован на языке C++, исходные коды библиотеки также находятся в открытом доступе.

1.1.3 BigARTM

BigARTM[17] – это кроссплатформенная библиотека с открытым исходными кодом, специализированная на решении задач тематического моделирования больших коллекций текстовых документов. Разрабатывается А. Фреем (МФТИ), К. Воронцовым (ВЦ РАН) и М. Апишевым (МГУ). Ядро библиотеки написано на C++, имеются интерфейсы для работы с библиотекой на C++, Python и из командной строки.

Библиотека является программной реализацией подхода аддитивной регуляризации ARTM, благодаря чему способна к построенной практически любой из возможных тематических моделей на основе мешка слов, включая модели PLSA и LDA.

Отличается наиболее эффективной на сегодняшнюю реализацию параллельных распределенных алгоритмов вероятностного тематического моделирования. В библиотеке, помимо его классической оффлайн версии, реализован параллельный онлайновый EM-алгоритм, что позволяет обрабатывать большие коллекции документов за одну итерацию, минимизируя используемый объём оперативной памяти. Кроме того, EM-алгоритм в BigARTM обобщён для мультимодальных регуляризованных моделей.

Эксперименты на коллекции из 3.7 миллионов статей английской Википедии и словаре из 100 тысяч уникальных слов показали, что BigARTM существенно выигрывает в скорости и точности работы, по сравнению с другими известными программными пакетами.

	procs	train	inference	perplexity
BigARTM	1	35 min	72 sec	4000
Gensim.LdaModel	1	369 min	395 sec	4161
VowpalWabbit.LDA	1	73 min	120 sec	4108
BigARTM	4	9 min	20 sec	4061
Gensim.LdaMulticore	4	60 min	222 sec	4111
BigARTM	8	4.5 min	14 sec	4304
Gensim.LdaMulticore	8	57 min	224 sec	4455

Рисунок 1.3 – Сравнение BigARTM с другими программными пакетами

Столбец procs отображает количество параллельных потоков, inference – время, потребовавшееся на тематизацию 100 тысяч документов, perplexity – перплексия на тестовой выборке документов.

На текущий момент BigARTM по умолчанию включены следующие метрики качества:

- перплексия;
- разреженность;
- средняя чистота тем;
- средняя контрастность тем;
- средний размер лексического ядра тем;
- доля фоновых слов во всей коллекции.

Библиотека также адаптирована для работы с мультимодальными тематическими моделями, учитывающими метаданные документов. Метаданные могут быть представлены в виде информации об авторах, метках времени создания документа, изображений в документе, цитируемых документах, цитируемых авторов и т.д.

1.3 Постановка задачи

Целью выпускной квалификационной работы является разработка поискового алгоритма на основе тематических моделей и его исследование.

Для достижения поставленной цели требуется:

1) разработать и реализовать алгоритм предобработки исходных данных;
2) освоить методы работы с библиотекой BigARTM и построить тематическую модель на коллекции документов;
3) разработать и реализовать алгоритм поиска документов по тематической модели;
4) разработать и реализовать алгоритм учета предпочтений пользователя и корректировки результатов запроса в соответствии с ними;
5) реализовать модуль для поиска документов по тематической модели с учетом предпочтений пользователя;
6) разработать методику экспериментального исследования алгоритма поиска
7) сделать выводы на основе полученных результатов исследований.

1.4 Выводы по разделу

В первом разделе были описаны общие принципы тематического моделирования и его математические обоснования. Были рассмотрены различные методы построения тематических моделей, указаны их недостатки и достоинства. На основе проведенного обзора в качестве используемого в данной работе подхода решено использовать подход аддитивной регуляризации тематических моделей ARTM, так как благодаря его гибкому аппарату регуляризаторов становится возможной тонкая подстройка модели под конкретные нужды решаемой задачи.

Также в данном разделе приводится обзор существующих программных решений, реализующих рассмотренные алгоритмы. Все найденные пакеты ПО представляют собой библиотеки функций или их компоненты, в то время как самостоятельных приложений, которые бы осуществляли работу с тематическими моделями, обнаружить не удалось.

2 РАЗРАБОТКА АЛГОРИТМОВ ПОИСКА

2.1 Алгоритм предварительной обработки документов

Перед использованием документов для построения модели или выполнения поисковых запросов их необходимо особым образом подготовить. Схема алгоритма предобработки документов имеет следующий вид:

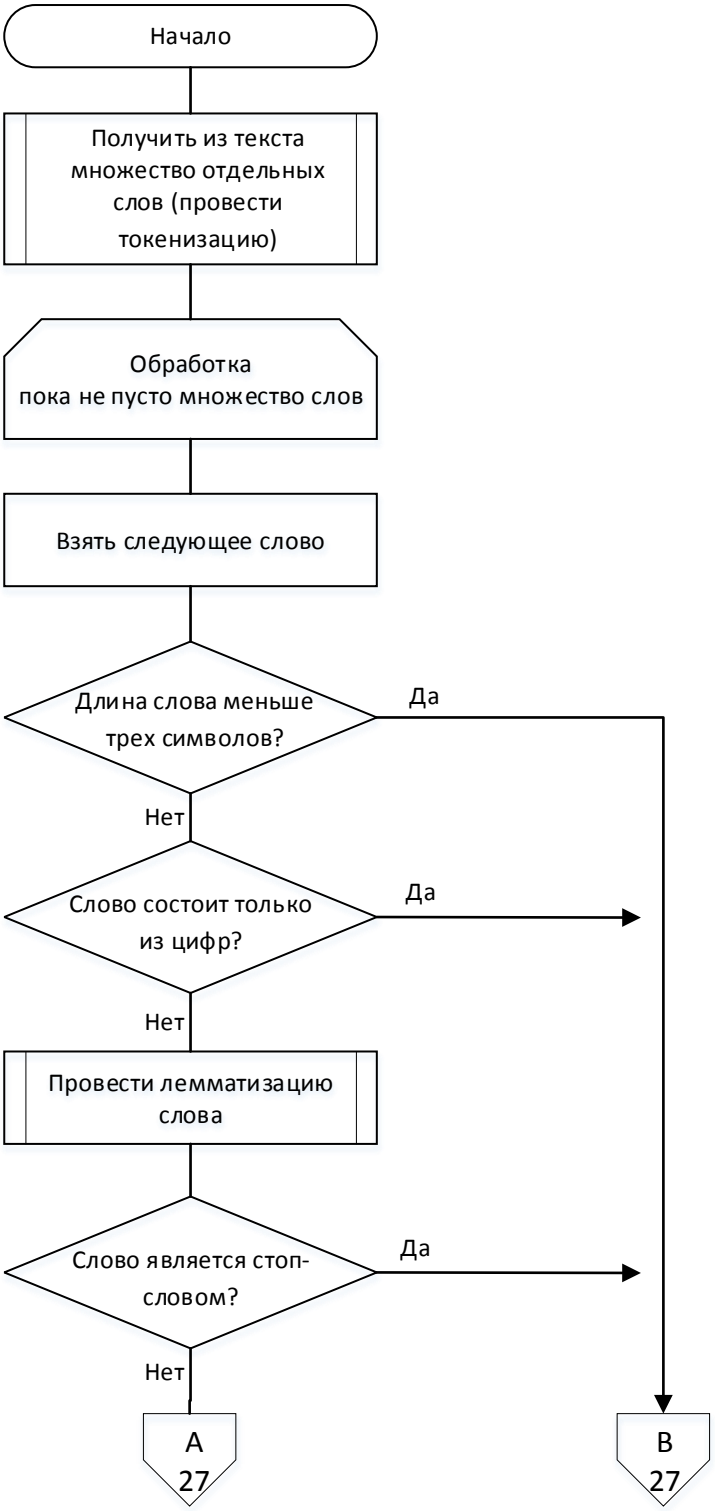


Рисунок 2.1 – Алгоритм предобработки документов (начало)

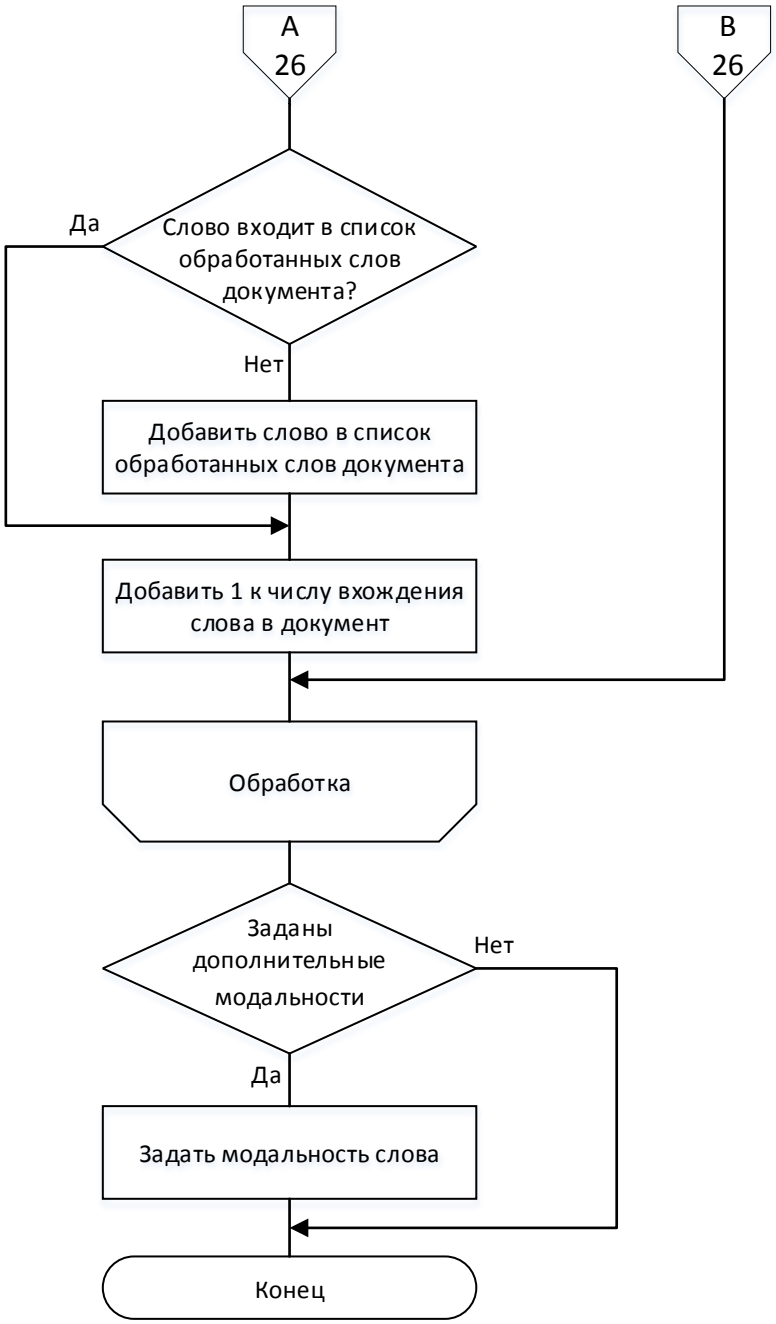


Рисунок 2.1 – продолжение

Прежде всего выполняется токенизация – разбиение целого текста на более мелкие части, токены. К токенам в общем случае относятся слова, числа и знаки пунктуации. Это необходимо для выделения слов в самостоятельные элементы, над каждым из которых можно проводить индивидуальные операции.

Все слова необходимо привести к единой форме либо путем стемминга (нахождение основы слова и удаление всех остальных его частей), либо посредством проведения лемматизации, то есть приведения слов к их нормальной (словарной) форме. Задача лемматизации требует намного больших вычислительных и временных затрат по сравнению со стеммингом, но взамен обеспечивает лучшее качество обработки, поскольку при стемминге информация

может быть утеряна. Для ряда языков, в число которых входит и русский язык, в силу их грамматических особенностей лемматизация является намного более предпочтительным вариантом предобработки.

Следующим шагом из исходных текстов следует удалить все стоп-слова, то есть слова, не имеющие конкретной тематической окраски, а потому понижающие качество тем конечной модели. К таким словам можно отнести предлоги, частицы, союзы, слова, число вхождений которых в исходный корпус чрезмерно велико, или, наоборот, мало и т.д. В общем случае слова с очень низкой частотой вхождений не способствуют формированию качественных и хорошо интерпретируемых тем, даже если их можно явно отнести к какой-либо из них, поэтому они также подвергаются удалению.

Вместе со стоп-словами из текстов также имеет смысл убрать числовые значения, то есть слова состоящие только из цифр, хотя выполнение данного шага может зависеть от особенностей исходной коллекции документов, и целей, которым должна отвечать модель.

Далее полученную в результате выполнения предыдущих шагов коллекцию необходимо привести к формату Vowpal Wabbit. Данный формат представляет собой текстовый файл, в котором документы располагаются друг за другом по одному на строку в виде мешка слов. Каждая строка начинается с идентификатора документа, затем, опционально, указывается название какой-либо модальности слов, вслед за которым перечисляются все слова, отнесенные к данной модальности и число их вхождений в данный документ. По умолчанию задается одна модальность – модальность основного текста. Дополнительные модальности – это метаданные документа, то есть информация, дополнительная по отношению к основному содержанию документа. Такими данными могут быть автор, время создания документа, категории, цитируемые документы и т.д. Предполагается, что метаданные могут помогать выявлять тематику документа, или, наоборот, могут быть предугаданы на основе тематики документа.

2.2 Алгоритм поиска и ранжирования документов по запросу

Документ-запрос сперва необходимо подвергнуть предобработке, аналогично документам исходного корпуса текстов.

Затем для документа-запроса формируется вектор-распределение по темам, предопределенным в матрице распределения тем по документам модели . Данный вектор получается в результате выполнения EM-алгоритма только для запроса. Однако в отличие от используемого при построение тематической модели подхода, матрица распределения тем по словам берется из самой модели и остается фиксированной на всем протяжении работы алгоритма. Обновлению на каждой итерации подвергается только интересующий нас вектор , и именно после его схождения работа алгоритма завершается.

По полученному вектору с помощью косинусной меры вычисляется близость запроса ко всем документам исходной коллекции. Формула косинусной меры имеет следующий вид:

здесь – это вес темы в документе коллекции и в запросе соответственно, – количество тем в модели.

Из результатов расчетов составляется рейтинг релевантности каждого документа коллекции к запросу – чем выше значение косинусной меры, тем большей тематической близостью обладает документ.

Схемы описанных выше алгоритмов приведены на рисунке 2.2 и рисунке 2.3:

Рисунок 2.2 – Общий алгоритм поиска документов по запросу

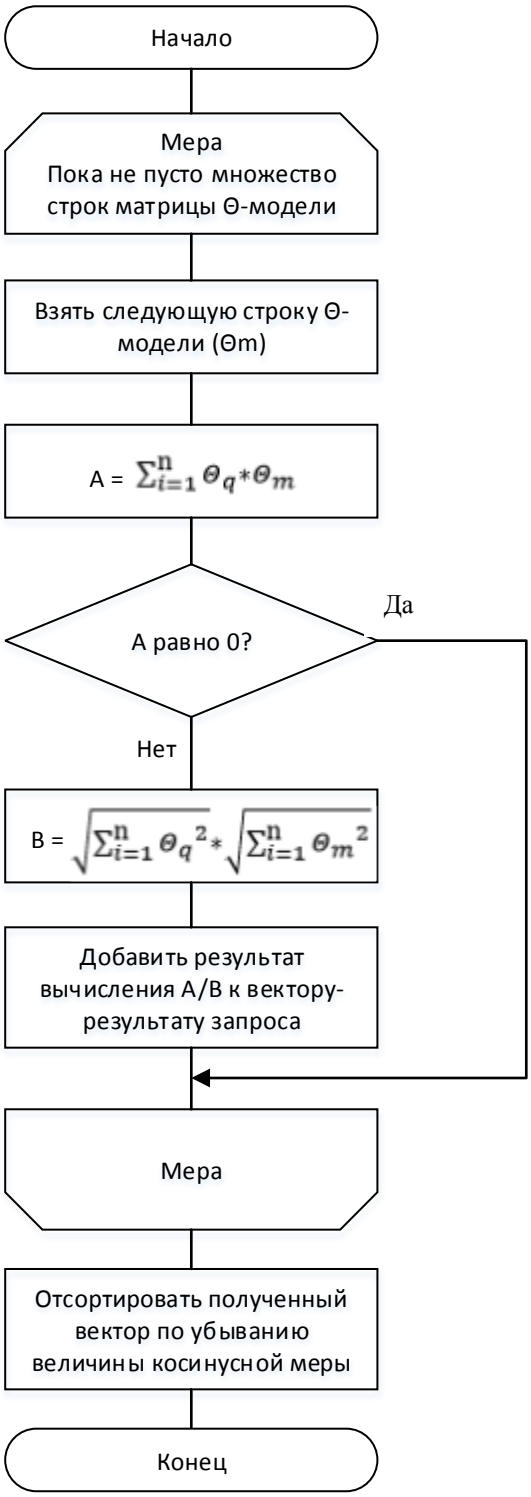


Рисунок 2.3– Алгоритм расчета косинусной меры документа-запроса

2.3 Алгоритм модификации запроса с учетом предпочтений пользователя

После выполнения запроса и вывода первых n документов с наибольшим значением косинусной близости может случиться так, что пользователь захочет уточнить поиск, задав дополнительный критерий. Такой критерий может

выражаться в более конкретной тематике, которой должны обладать результаты поиска.

Например, задав в качестве исходного запроса статью про постройку и программирование роботов своими руками, и получив на выходе связанные с роботостроением статьи, пользователь может решить, что его интересуют только статьи, посвященные созданию летающих дронов и мультикоптеров.

В таком случае у него должна быть возможность скорректировать поисковую выдачу так, чтобы максимизировать в ней количество релевантных новому критерию документов и исключить из нее документы, по-прежнему тематически близкие исходному запросу, но не соответствующие заданному критерию.

Этого можно добиться, если позволить пользователю помечать просмотренные им результаты поиска как релевантные и нерелевантные его пожеланиям. Сами пожелания при этом выступают скрытым критерием и явным образом не обозначаются. От системы требуется самостоятельно выявить их на основе предоставленного пользователем разбиения документов на релевантные/нерелевантные и скорректировать выдачу соответствующим образом.

В рамка данной работы рассматриваются два метода для достижения поставленной цели – предугадывание релевантности документов на основе логистической регрессии и модификация весов тем в исходном запросе.

2.3.1 Логистическая регрессия

В основе метода предугадывания лежит предположение о том, что размеченные пользователем документы можно использовать в качестве обучающего множества данных для построения регрессионной модели. Соответственно, все еще не просмотренные пользователем документы тогда можно рассматривать как тестовую выборку. Для каждого документа из тестового набора данных модель способна сделать предсказание о том, к какому классу и с какой вероятностью он относится. Исходными данными здесь, как и в случае косинусной меры, выступают вектора распределений тем по документам.

Для решения данной задачи было принято решение использовать логистическую регрессию, так как значением ее функции является вероятность принадлежности исходного значения к определенному классу, а не значение числовой переменной, как в случае с обычной линейной регрессией. Ее формула выглядит следующим образом:

–

где признак i-го объекта;
класс i-го объекта;
вектор весов;
обратный коэффициент силы регуляризации;
вес i-го объекта;

Количество признаков для каждого документа равно количеству его тем и может равняться нескольким сотням. Количество объектов же в обучающей выборке ограничивается размеченным пользователем множеством документов, и, в общем случае, намного меньше количества признаков. Такой расклад приводит к переобучению регрессионной модели и, как следствие, некорректным предсказаниям. Чтобы этого избежать в функцию регрессии дополнительно вводится L-2 регуляризация, представленная первым слагаемым в формуле. Данный регуляризатор позволяет снизить эффект переобучения и улучшить обобщающие способности получившейся модели.

2.3.2 Коррекция распределения тем в запросе

Извлеченные из размеченных пользователем документов данные об их тематическом распределении можно использовать для модификации вектора тем исходного запроса.

Для этого к вектору-запросу с некоторым коэффициентом прибавляются вектора документов, помеченных как релевантные, и вычитаются вектора нерелевантных. Результат приводится к нормализованному вектору. Таким образом удается добиться того, что у тематик, превалирующих в релевантных документах, повышается вес и в запросе. И, обратным образом, определяющие нерелевантные статьи тематики получают в нем пониженную значимость. Коэффициент задается следующим образом.

$$\frac{\qquad}{\qquad} \qquad \frac{\qquad}{\qquad}$$

где M – мощность большего из множеств релев./нерелв. документов,

– мощность меньшего из множеств релев./нерелв. документов,

– мощность множества нерелевантных документов,

– мощность множества релевантных документов,

–

для релевантных и нерелевантных документов соответственно.

По результирующему вектору снова выполняется расчет косинусной меры и ранжирование поисковой выдачи.

2.4 Выводы по разделу

В разделе был описан алгоритм предобработки исходных документов и приведения их в пригодный для работы с ними тематической модели формат.

Также были рассмотрены общие алгоритмы выполнения поисковых запросов по тематической модели, и два различных метода модификации поисковой выдачи на основе данных о предпочтениях пользователя.

3 РАЗРАБОТКА МОДУЛЯ ПОИСКА

3.1 Диаграмма компонентов

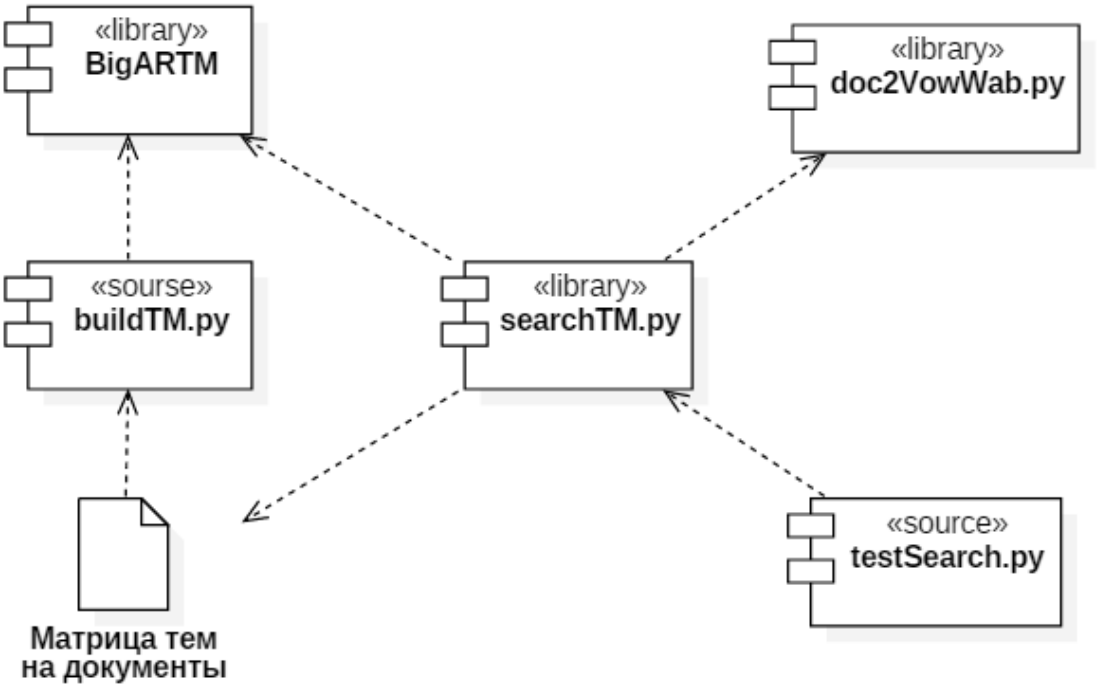


Рисунок 3.1 – Диаграмма компонентов

Поисковому модулю searchTM для работы требуется матрица распределения тем по документам, то есть матрица θ. Данную матрицу же, в свою очередь, можно получить из тематической модели, построить которую можно с помощью библиотеки BigARTM.

Готовую матрицу можно сохранить на диск, в таком случае модуль просто загрузит ее в память. Альтернативно, можно не сохранять саму матрицу, а передать в поисковой модуль объект самой модели, тогда матрица θ будет извлечена напрямую из нее.

Зависимость от библиотеки BigARTM также выражается в использовании ее метода, позволяющего получить распределение произвольного документа по темам построенной тематической модели.

Модуль doc2VowWab реализует алгоритмы предобработки исходного блока документов модели и документов запроса. Перед использованием с поисковым модулем любой текстовый документ сперва необходимо привести к определенному формату, называемому Vowpal Wabbit.

Модуль testSearch содержит в себе функции, с помощью которых проводилось тестирование реализованных методов поиска документов и учета предпочтений пользователя.

Модуль buildTM реализует построение используемой для проведения экспериментальных исследований тематической модели.

3.2 Диаграмма классов

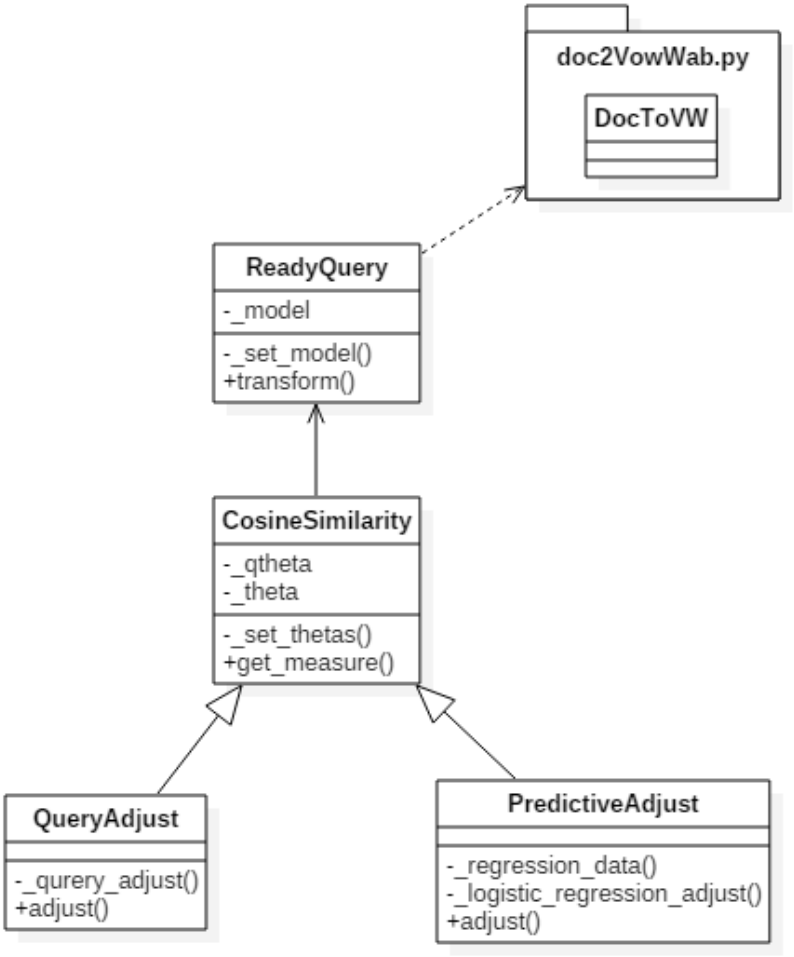


Рисунок 3.2 – Диаграмма классов модуля SearchTM

Класс CosineSimiliraty воплощает базовую поисковую логику модуля. Именно его полях содержатся матрицы распределения тем по документам тематической модели и запроса, и именно в нем реализован расчет косинусной меры их сходства.

Его логику расширяют наследуемые от него классы QueryAdjust и PredictiveAdjust, реализующие коррекцию распределения тем в векторе-запросе и предсказание релевантности документов с помощью регрессии соответственно.

Класс ReadyQuery обеспечивает взаимодействие с объектом тематической модели, если таковой был предоставлен, и отвечает за получение вектора тематического распределения запроса.

Класс DocToVW реализует предварительную обработку документов и приведение их к формату, пригодному для взаимодействия с тематическими моделями.

3.3 Описание библиотеки модуля

Разработанный пакет SearchTM для языка Python состоит из двух модулей. Первый модуль searchTM.py содержит в себе всю логику выполнения поискового

запроса по тематическим моделям. Второй модуль является скорее вспомогательным и отвечает за предобработку текстовых данных в исходных документах и приведению их к подходящему для работы основного модуля формату.

Модуль SearchTM предоставляет функции:

1) поиска тематически близких запросу документов на основе расчета косинусной меры близости их векторов распределения тем;
2) модификации поисковой выдачи с учетом предпочтений пользователя методом коррекции распределения тем в исходном запросе;
3) модификации поисковой выдачи с учетом предпочтений пользователя методом предсказания релевантности документов посредством регрессионного анализа (логистической регрессии);

Взаимодействие с библиотекой может осуществляться по различным сценариям. Предполагается, что в распоряжении пользователя данной библиотеки есть построеннаяс помощью средств библиотеки BigARTM тематическая модель (в виде объекта в оперативной памяти или сохраненная на диск), и, желательно, извлеченная из нее матрица распределения тем на документы Θ.

Если передать в библиотеку только объект модели, матица Θ будет извлечена из нее. Если передать путь до модели, сохраненной на диске, она будет считана в оперативную память. Последний вариант является наименее предпочтительным, поскольку размер модели может достигать гигабайт (ее размер зависит от размера исходной коллекции и, строго говоря, может быть сколь угодно большим), и на ее загрузку может потребоваться значительное количество времени.

Поисковым запросом может быть вектор тематического распределения некоторого документа (соответствующий тематикам имеющейся тематической модели), либо сам этот документ в необработанном виде. Во втором случае автоматически будет произведена предобработка документа и получение его вектора тем.

Наиболее общий сценарий взаимодействия с модулем может быть следующим:

1) создается объект CosineSimiliraty, в его конструктор передаются матрица Θ тематической модели и вектора документа-запроса, либо сам этот документ и тематическая модель;
2) выполняется метод get_measure(), возвращающий значение косинусной меры близости между документом запроса и каждым документом коллекции; в метод можно передать новый документ запроса или вектор , запрос тогда будет выполнен именно с ним;
3) создается объект класса QueryAdjust или PredictiveAdjust (возможно их одновременное использование), в метод adjust() которого в виде двух списков передаются имена (совпадающие с идентификаторами документов матрицы Θ) документов, оцененных пользователем как релевантные или нерелевантные к его запросу;

4) метод adjust() возвращает скорректированную под предпочтения пользователя поисковую выдачу.

Также возможно создание экземпляра класса DocToVW и вызов его методов для ручной обработки исходных документов и приведению их к формату Vowpal Wabbit, с которым работает модуль, но обычно в этом нет необходимости.

3.4 Выводы по разделу

В разделе описана программная составляющая данной работы. Приведены диаграммы классов и компонентов разрабатываемого модуля, описаны его зависимости, структура и структура его компонентов.

Также рассмотрены предоставляемые модулем функциональные возможности и способы организации взаимодействия с ним.

4 ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ

4.1 Методика эксперимента

Для проведения исследований в рамках данной работы прежде всего была построена тематическая модель, по которой и в дальнейшем и осуществляется поиск. Детальное описание процесса построения модели и используемой коллекции документов приведено в разделе 4.1.1.

Сам алгоритм поиска заключается в определении тематической близости между документом-запросом и документами, входящих в модель, и последующей коррекции результатов поиска с учетом предпочтений пользователя на основе разметки документов на релевантные или нерелевантные его запросу. Коррекция достигается путем применения двух разных подходов – предсказания релевантности документов на основе логистической регрессии и коррекции самого запроса.

Целью экспериментов ставилось определение эффективности предлагаемых методов улучшения релевантности поисковой выдачи, сравнение их друг с другом и с простой выдачей без их использования.

Для этого были предприняты следующие шаги. В качестве запроса был выбран определенный документ. Затем был выполнен поиск тематически близких ему документов на основе (косинусной меры) косинусного сходства между векторами-распределениями тем документов коллекции и документа-запроса.

Из результатов поиска были взяты первые 100 документов с наибольшим показателем сходства по косинусной мере. Все дальнейшие действия по ходу эксперимента осуществлялись только с ними.

Далее был задан некоторый дополнительный критерий релевантности. В качестве критерия для запроса по постройку робота своими руками, например, может выступать требование, чтобы в статье речь шла про дроны, беспилотники, трикоптеры, квадрокопторы и т.д. В полученной на предыдущем шаге выборке путем ручного просмотра содержания документов были найдены все отвечающие этому критерию (релевантные ему) документы.

После чего на каждом шаге алгоритма улучшения релевантности выводилось по 20 документов (из отобранных 100), которые размечались на релевантные и нет, и на основе полученного разбиения производилась корректировка следующих к выдаче 20 документов. Сравнение методов проводилось по количеству найденных ими на каждом шаге релевантных документов и по тому, насколько быстро ими были найденные все релевантные документы в отобранной сотне.

Общий алгоритм эксперимента для обоих методов коррекции можно описать следующим образом:

1) выполнить первоначальный поисковый запрос и получить значение его тематической близости ко всем документам коллекции;

2) отобрать 100 документов с наибольшим значением тематической близости по косинусной мере, последующие шаги осуществлять только с их использованием;
3) задать некий дополнительный критерий, делающий запрос более строгим;
4) в отобранных ста документах найти все, отвечающие заданному критерию (релевантные) и запомнить их;
5) вывести первые 20 документов с наибольшим косинусным сходством и разметить их на релевантные и нерелевантные;
6) выполнить новый, скорректированный на основе полученного на предыдущем шаге распределения запрос;
7) из еще не размеченных документов вывести 20 с наибольшим после корректировки показателем релевантности;
8) повторять шаги 6-7 до тех пор, пока не будут найдены все релевантные документы с шага 4.

Собранная в ходе эксперимента информация об эффективности каждого из методов коррекции приводится в разделе 4.2.

4.1.1 Построение тематической модели

Для реализации алгоритма поиска и проведения экспериментов необходима построенная по определенной коллекции документов тематическая модель. В качестве исходного корпуса документов была взята коллекция статей с интернет ресурсов Habrahabr и Geektimes за 2008 – 2015 годы. Размер коллекции составляет 135 000 документов. Тематика статей достаточно разнообразна, но большинство из них имеет связь с программированием, IT-индустрией, инженерией и роботостроением.

Количество тем тематической модели было решено установить равным 250, из которых 50 тем отведены под фоновые темы (где должна собираться общая лексика и слова, слабо влияющие на интерпретируемость тем), а остальные 200 – под темы, которые должны стать основными.

Используемая в данной работе коллекция документов предоставляет различную мета информацию о каждом документе, такую как: дата публикации, теги, хабы (аналог тегов, но с большей степенью обобщения), никнейм автора, количество комментариев и рейтинг статьи. Данную информацию также можно использовать для построения тематической модели. В используемой модели была задана дополнительная модальность хабов, где находится информация о том, к каким хабам относится каждая статья.

В качестве регуляризаторов модели были использованы регуляризаторы сглаживания/разреживания матриц и регуляризатор декоррелирования матрицы и регуляризатор улучшения когерентности тем. Регуляризация проводилась для фоновых и основных тем отдельно – на фоновых осуществлялось сглаживание, в то время как основные подвергались разрежеванию и декорелированию.

Алгоритм, по которому строилась модель, имеет следующий вид:

1) активировать регуляризатор сглаживания (раздел 1.1.3.1) на всех 250 темах, с большим коэффициентом сглаживания на фоновых темах, нежели на основных;
2) провести 15 проходов по коллекции;
3) отключить сглаживание основных тем, активировать для них регуляризатор декоррелирования (раздел 1.1.3.4);
4) провести 20 проходов по коллекции;
5) активировать регуляризатор разреживания (раздел 1.1.3.2) для основных тем и регуляризатором улучшения когерентности тем;
6) провести 35 проходов по коллекции.

Первоначальное сглаживание всех тем проводится для предотвращения зануления слов и тем с низкой частотой оценкой, но потенциально важной ролью для интерпретируемости тем и качества модели в целом.

Для оценки состояния модели использовались метрики перплексии (раздел 1.1.4, рисунок 4.1), разреженности матриц и (раздел 1.1.3.2, рисунок 4.2), контрастность и чистота тем (формулы (4.1), (4.2), рисунок 4.3). Ниже приведены графики снятых с построенной модели метрик.

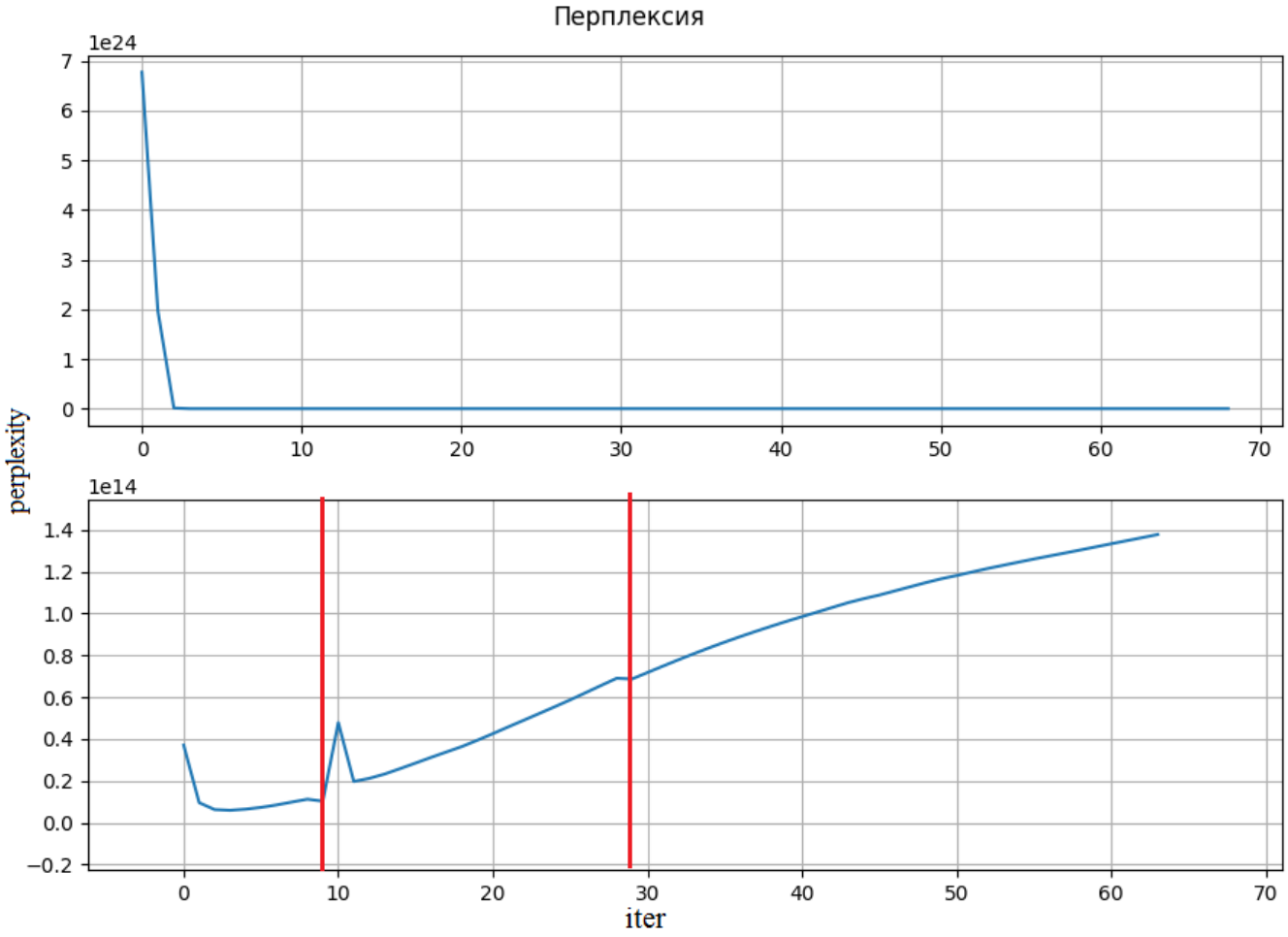
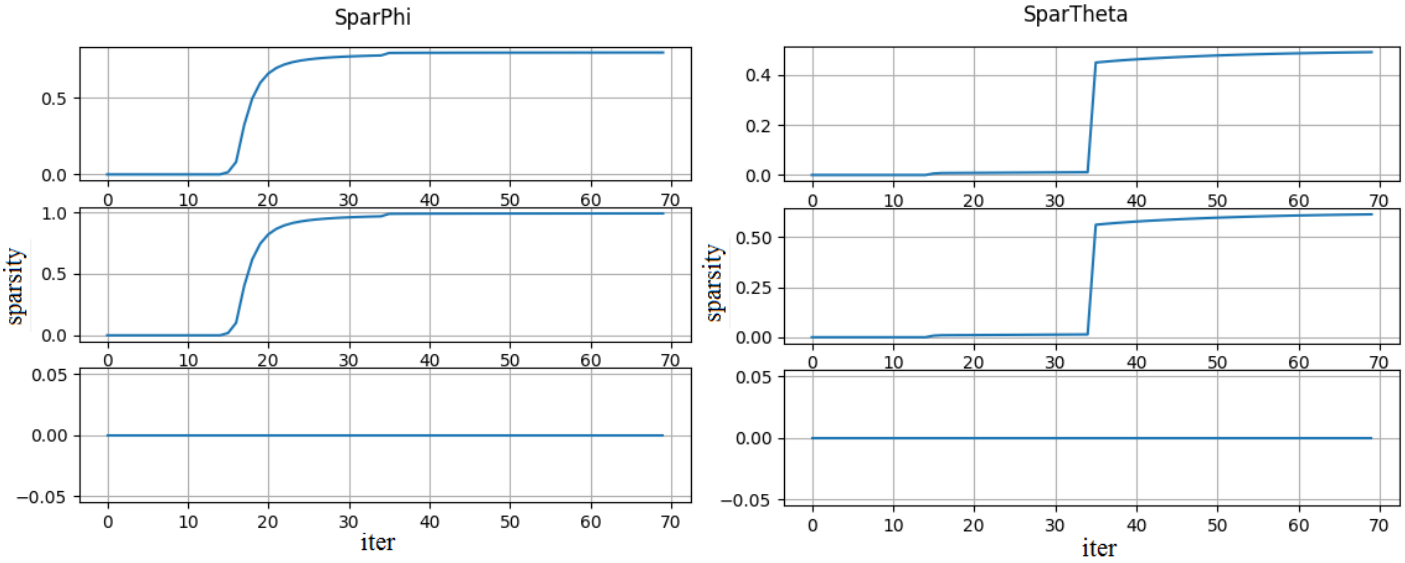


Рисунок 4.1 – Перплексия модели с начальными значениями (сверху) и без них (снизу)

Перед началом вычислений матрицы инициализируются случайными значениями, поэтому значение перплексии на нулевой и нескольких последующих итерациях очень велико. Поэтому, чтобы получить корректное графическое представление перплексии, нижний график не учитывает первые 5 итераций.

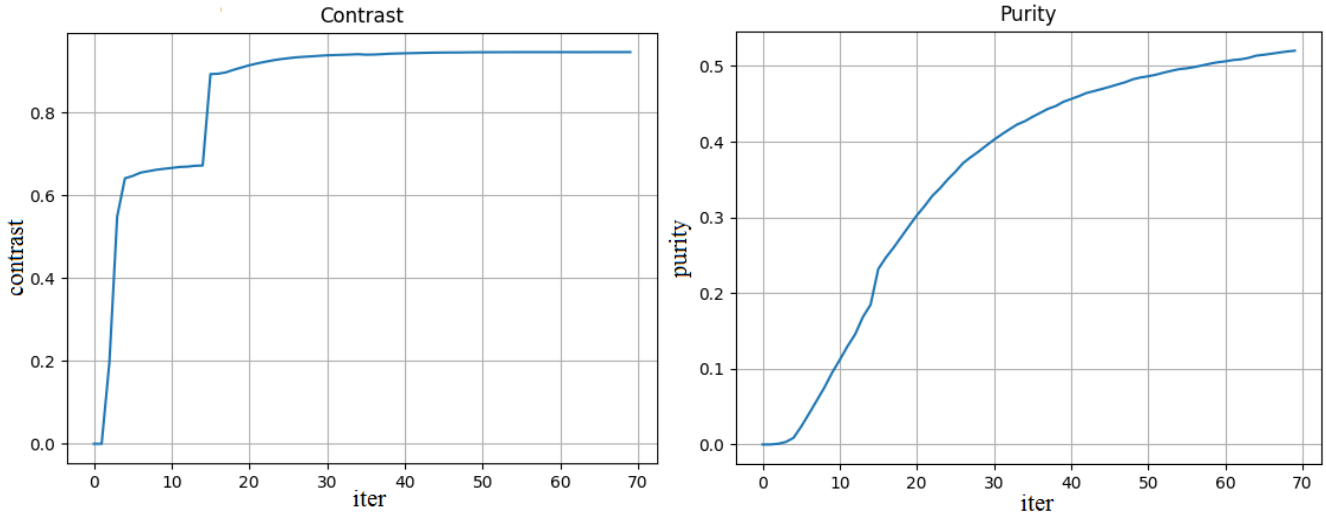
Красными вертикальными линиями обозначены итерации 15 и 35 из шагов алгоритма 2 и 6 соответственно. Именно на них производилась смена параметров регуляризации.



а) б)

Рисунок 4.2 – Разреженность матриц (а) и (б)

Верхние графики соответствуют разреженности матриц по всем 250 темам модели, средние показывают разреженность только в главных 200 темах, которые подвергались разреживанию и декорреляции, а нижние – по фоновым 50 темам, которые подвергались сглаживанию.



а) б)

Рисунок 4.3 – Средняя контрастность (а) и чистота (б) основных тем модели

Контрастность и чистота являются дополнительной метрикой интерпретируемости и различности тем. Чистоте соответствует формула:

(4.1)

.

Переменная q задает пороговое значение вероятности для слов, которые будут учитываться при расчете данной метрики.

Контрастность определяется формулой:

— (4.2)

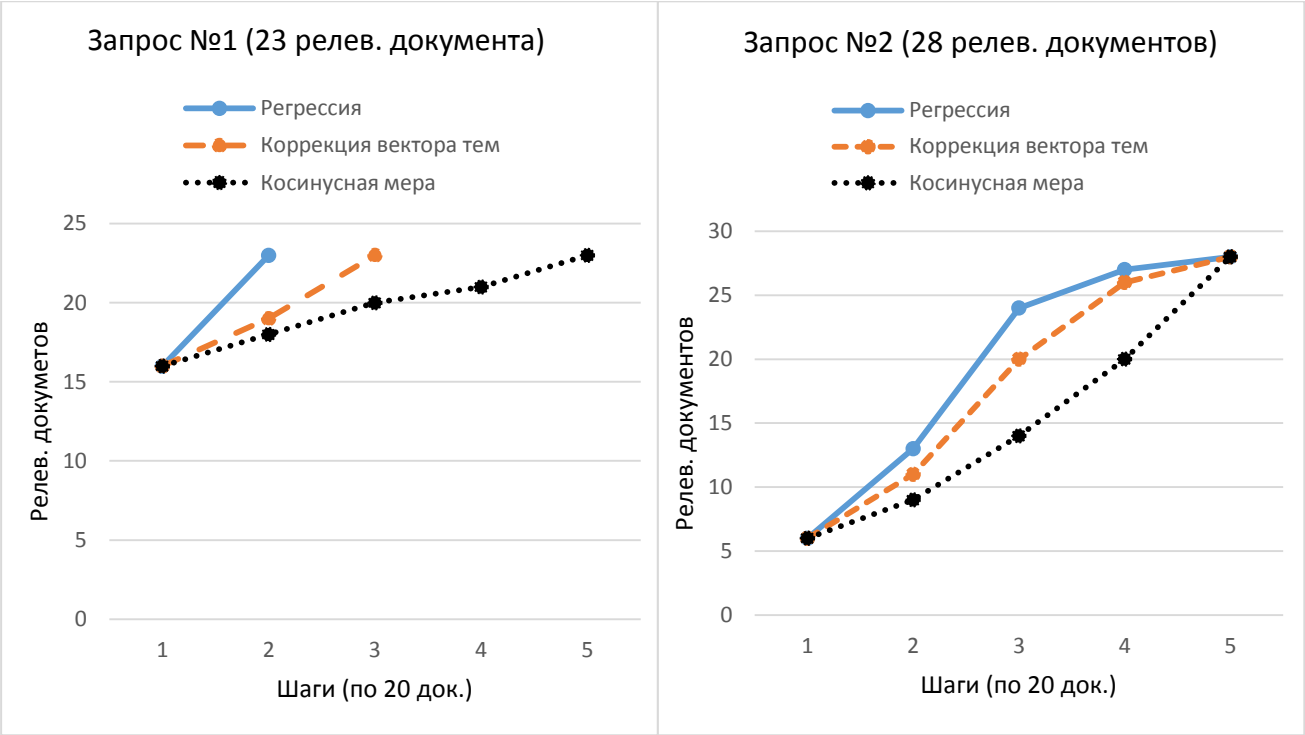
– приблизительно равно —.

Более высокие значения контрастности и частоты соответствуют более качественным темам.

4.2 Результаты экспериментов

Эксперименты проводились для разных запросов с различными тематиками и варьирующимся количеством релевантных документов и их распределением. Ниже для обоих описанных методов улучшения поиска на примере двух разных запросов приведены графики скорости нахождения всех релевантных дополнительному критерию документов из отобранных на 4 шаге алгоритма. Для перового запроса (рисунок 1а) число релевантных документов составило 23, для второго (рисунок 1б) – 28.

Графики отражают то, насколько быстро (за сколько итераций шага 8) каждый из методов справился с нахождением релевантных документов, и сколько их было найдено на каждой итерации. Пользователь заинтересован в скорейшем получении желаемых результатов – следовательно, чем более стремительно график метода приближается к верхнему значению (23 и 28 для запроса №1 и №2 соответственно), тем он более эффективен.



а) б)

Рисунок 4.4 – Общее число найденных на каждой итерации релевантных документов
(а – запрос с 23 релевантными документами; б – запрос с 28 релевантными документами)

Графики косинусной меры здесь, по существу, отражают исходное распределение релевантных дополнительному критерию документов в отобранной (шаг 2 алгоритма) сотне. Таким образом, например, для второго запроса в первых 20 документах с наибольшей косинусной мерой было 6 релевантных документов, в следующих двадцати по косинусной мере их было 3, затем 5 и т.д. Графики имеют одинаковую начальную точку в рамках одного запроса, поскольку первый шаг поиска (шаг 5 алгоритма) задает первые релевантные документы и совпадает для всех методов.

На графиках на рис. 1а видно, что после разметки первых 20 документов регрессионному методу понадобился всего один шаг, чтобы найти все остальные релевантные документы. Метод коррекции запроса справился за два шага.

Во втором запросе (рис. 1б) обоим методам потребовалось столько же итераций, сколько и простому выводу документов по косинусной мере, однако благодаря их применению повышается число релевантных документов, выводимых на ранних итерациях. Так, к третьей итерации поиска по одной косинусной мере удалось бы найти всего 14 релевантных документов из 28, в то время как регрессия и коррекция тем на том же шаге позволяют обнаружить 24 и 20 документов соответственно.

Для измерения качества работы каждого метода здесь была введена следующая метрика:

– количество шагов, потребовавшихся методу для нахождения всех релевантных документов,

– общее число релевантных документов для данного запроса,

– процент релевантных документов на данном шаге от общего числа релевантных.

Таким образом, максимальное значение, которое может принимать данная величина равна 100, что соответствует нахождению всех релевантных документов на первом шаге. Для косинусной меры средний показатель данной величины составил порядка 51%, для метода коррекции запроса – 63%, а для регрессионного метода – 77%.

Как показали эксперименты, и предсказание релевантности документов на основе регрессионной модели, и коррекция вектора-распределения тем запроса предоставляют пользователю больше контроля над результатами поиска, позволяют значительно улучшить релевантность поисковой выдачи и существенно ускоряют нахождение документов, соответствующих предпочтениям пользователя, двигая их вверх в очереди на вывод.

При этом особенно выделяется регрессионный подход, демонстрирующий наилучшие из описанных методов результаты. Применение этого метода повышает релевантность выдачи в среднем на 51% по сравнению с косинусной мерой, в время как с помощью коррекции запроса поиск удалось улучшить в среднем на 23%.

4.3 Выводы по разделу

В разделе предложена методика экспериментов, проверяющих эффективность описанных методов улучшения релевантности поисковой выдачи. Также приводится описание построенной для проведения экспериментов тематической модели.

Оба рассматриваемых метода показали значительное улучшение результатов поиска, и особенно хорошими показателями отличился метод регрессионного предсказания релевантности документов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы мной были изучены методики и принципы построения тематических моделей. В частности, такие их вариации как латентно-семантический анализ LSA, его вероятностная версия PLSA, латентное размещение Дирихле LDA и подход аддитивной регуляризации ARTM. Для дальнейшей работы был выбран подход ARTM, для которого также были рассмотрены некоторые регуляризаторы. Также разработан алгоритм подготовки исходных данных для тематических моделей.

С помощью библиотеки BigARTM была построена тематическая модель, с использованием которой разрабатывался алгоритм поиска документов и алгоритмы улучшения релевантности поисковой выдачи посредством учета предпочтений пользователя. Было разработано два алгоритма улучшения результатов поиска – метод регрессионного предсказания релевантности документов и корректировка вектора тем документа-запроса.

Разработанные алгоритмы были реализованы в виде подключаемого модуля для языка Python 3. Данный модуль можно использовать для других проектов, где требуется функционал поиска по тематическим моделям или предобработки текстовых данных для работы с ними.

Также была разработана методика для экспериментального исследования описанных алгоритмов поиска. Проведённые эксперименты продемонстрировали эффективность разработанных методов – с помощью коррекции тем запроса по введенной в рамках экспериментов метрике эффективность поиска удалось улучшить на 23%, а с использованием регрессионного предсказания релевантности – на 51%.

Таким образом, поставленные на выпускную квалификационную работу задачи были выполнены, но тематическое моделирование, как и информационный поиск с его использованием – это относительно молодая и чрезвычайно обширная сфера для дальнейших исследований и совершенствований, в которой на сегодняшний день остается множество нерешенных задач и объектов для изучения и улучшения.

По результатам проведенного исследования написана статья, которая была опубликована в выпуске №26 естественнонаучного журнала «Точная наука» [18].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Воронцов, К.В. Вероятностное тематическое моделирование: обзор моделей и регуляризационный подход / К.В. Воронцов. –Дата обновления: 16.08.2017. URL: http://machinelearning.ru/wiki/images/d/d5/Voron17survey-artm.pdf (дата обращения: 11.01.2018).

2. Hofmann, T. Probabilistic latent semantic indexing / T. Hofmann // Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. – ACM, 1999. – С. 50-57.

3. Blei, D. M. Latent dirichlet allocation / D.M. Blei, A. Y. Ng , M. I. Jordan // Journal of machine Learning research. – 2003. Т. 3. №. Jan. – С. 993-1022.

4. Воронцов, К.В. Регуляризация, робастность и разреженность вероятностных тематических моделей / К.В. Воронцов, А.А. Потапенко // Компьютерные исследования и моделирование 2012. № 4. – С. 693-706

5. Нижибицкий, Е.А. Относительная перплексия как мера качества тематических моделей / Е.А. Нижибицкий . –Дата обновления: 07.04.2014. URL: http://www.machinelearning.ru/wiki/images/0/0c/NizhibitskyLomonosovSlides14.pdf (Дата обращения: 13.01.2018).

6. Сингулярное разложение // Википедия, свободная энциклопедия.. –Дата обновления: 24.02.2016. URL: https://ru.wikipedia.org/wiki/Сингулярное_разложение (Дата обращения: 15.01.2018).

7. Thomas, L. Introduction to Latent Semantic Analysis / LandauerThomas, Peter W. Foltz, Darrell Laham. –Дата обновления: 07.04.2014. URL: http://lsa.colorado.edu/ papers/ dp1.LSAintro.pdf (Дата обращения: 15.01.2018).

8. Blei, Introduction to Probabilistic Topic Models / Blei, M. David . – Comm. ACM 55 (4). – С. 77–84.

9. Коршунов, А. Тематическое моделирование текстов на естественном языке / А. Коршунов, А. Гомзин . –Дата обновления:. URL: http://www.ispras.ru/proceedings/ docs/2012/23/isp_23_2012_215.pdf (Дата обращения: 15.01.2018).

10. Masada T. Comparing LDA with pLSI as a dimensionality reduction method in document clustering / T. Masada, S. Kiyasu, S. Miyahara // Proceedings of the 3rd International Conference on Large-scale knowledge resources: construction and applications. – LKT'08. Springer-Verlag, 2008. – С. 13-25.

11 Воронцов, К.В. Модификации EM-алгоритмов для вероятностного тематического моделирования / К.В. Воронцов, А.А. Потапенко // Машинное обучение и анализ данных. – 2013. Т.1, №6. – С.657-686.

12. Воронцов, К.В. Аддитивная регуляризация тематических моделей коллекции текстовых документов / К.В. Воронцов // Доклады РАН. – 2014. Т. 456. №3 С. 268-271

13. Тихонов, А.Н. Методы решения некорректных задач / А.Н. Тихонов, В.Я. Арсенин // М.: Наука, 1986 . – 285с.

14. Vorontsov, K. V. Tutorial on probabilistic topic modeling: Additive regularization for stochastic matrix factorization / K. V. Vorontsov , A. A. Potapenko //

AIST’2014, Analysis of Images, Social networks and Texts. — 2014. Vol. 436. C. 29–46.

15. Gensim official page URL: http://radimrehurek.com/gensim/about.html. – Дата обновления: 10.12.2017 (Дата обращения: 15.01.2018).

16. VowpalWabbit official page. URL: https://github.com/JohnLangford/vowpal_wabbit/wiki . – Дата обновления: 27.04.2017 (Дата обращения: 15.01.2018).

17. BigARTM documentation URL: http://docs.bigartm.org/en/stable/index.html. – Дата обновления: 27.07.2017 (Дата обращения: 15.01.2018).

18. Тепляков С.Ю. Повышение релевантности поисковой выдачи и учет предпочтений пользователя в разведочном поиске / С.Ю. Тепляков, Т.Ю. Оленчикова // Точная наука. – 2018. №26. – С. 107-112.

ПРИЛОЖЕНИЕ 1 ОПИСАНИЕ ПРОГРАММЫ

1. Общие сведения

Программный модуль для поиска документов по тематическим моделям по запросу в виде целого документа. Для работы модуля необходимо наличие построенной тематической модели и установленной библиотеки для тематического моделирования BigARTM. Модуль реализован на языке Python 3.

2. Функциональное назначение

Данный модуль предназначен для подключения к другим программным проектам в качестве компонента и реализации в них логики поиска по тематическим моделям.

3. Описание логической структуры

Модуль состоит из набора следующих подмодулей:

searthTM.py – модуль поиска, содержит в себе классы и методы для нахождения косинусной меры близости между документом запроса и документами тематической модели, а также модификации поисковой выдачи с целью учета предпочтений пользователя;

doc2VowWab.py – реализует алгоритм предобработки документов и приведение их формату, подходящему для их использования в построении тематических моделей и поиску по ним;

buildTM.py и testSearch.py – модули, реализованные в рамках экспериментального исследования разработанных алгоритмов поиска, отвечают за построение тематической модели и тестирование поиска соответственно.

4. Используемые технические средства

Системные требования для работы модуля в первую очередь определяются размером тематической модели, это особенно справедливо в отношении оперативной памяти. Минимально необходимо иметь компьютер с объемом оперативной памяти не менее 500Мб оперативной памяти и тактовой частотой процессора не менее 1.8 ГГц.

5. Вызов и загрузка

Для использования реализованных в модуле функций его необходимо подключить к программному проекту через соответствующую директиву подключения внешнего модуля.

6. Входные и выходные данные

На вход модулю подается построенная тематическая модель, по которой будет осуществляться поиск и запрос либо в виде необработанного документа, либо как его вектора распределения тем.

На выходе модуль формирует ранжированный по степени релевантности к заданному запросу список документов тематической модели, по которой осуществлялся поиск.

ПРИЛОЖЕНИЕ 2 ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ

Модуль searchTM.py

```
import os
import shutil
from time import time
import pandas as pd
from operator import itemgetter
from sklearn.linear_model import LogisticRegression
import numpy as np

def _timeit(method):
    def timed(*args, **kw):
        ts = time()
        result = method(*args, **kw)
        te = time()

        print('{} completed in {:.2f}s'.format(method.__name__, (te-
ts)))
        return result
    return timed

class ReadyQuery:
    def __init__(self, model=None):
        """
        :param model: topic model to get query Theta distribution
from, if None can't use transform method
        :type model: str or reference to BigARTM artm model object
        """
        if model is not None:
            self._model = self._set_model(model)
        else:
            self._model = None

    @classmethod
    def _set_model(cls, arg_model):
        try:
            import artm
        except ModuleNotFoundError as e:
            raise ModuleNotFoundError('Make sure python knows path
to BigARTM library.') from e
        if isinstance(arg_model, str) and os.path.isdir(arg_model):
            model = artm.load_artm_model(arg_model)
        if not isinstance(model, (artm.ARTM, artm.LDA)):
            raise TypeError('Use BigARTM library model')
        return model

    def transform(self, filename):
        """
            :Description: Returns file's Theta distribution which
corresponds to model's Theta matrix
```

```
            :param str filename: path to the file to get Theta
distribution of
            :return:
                * pandas.DataFrame: (data, columns, row), where:
                * columns --- the names of topics in topic model;
                * row --- id of the query document;
                * data --- content of Theta matrix.
            :Note:
                * Needs model to be set
        """
        if self._model is None:
            raise AttributeError('{} object has no model set. Set
the model in the constructor'
                                 .format(self.__class__.__name__))
        import artm
        bv_folder = os.path.join(os.getcwd(), 'query_batches')
        query_bv = artm.BatchVectorizer(data_path=filename,
                                        data_format='vowpal_wabbit',
                                        batch_size=10,
                                        target_folder=bv_folder,
                                        gather_dictionary=True)
        query_theta                                                =
self._model.transform(batch_vectorizer=query_bv)
        shutil.rmtree(bv_folder)
        return query_theta


class CosineSimilarity(ReadyQuery):
    def __init__(self, theta, qtheta=None, file=None, model=None):
        """
        :param theta: Theta matrix of the topic model
        :type theta: csv or hdf (.h5) table
        :param qtheta: Theta distribution of the query document
        :type qtheta: csv or hdf (.h5) table
        :param str file: path to an unprepared query document
        :param model: topic model to get query Theta distribution
from, if None can't get\
                      distribution for a new document
        :type model: str or reference to BigARTM artm model object
        """
        super(CosineSimilarity, self).__init__(model)
        self._theta = self._set_thetas(theta)
        if qtheta is not None:
            self._qtheta = self._set_thetas(qtheta)
        elif file is not None and model is not None:
            super(CosineSimilarity, self).__init__(model=model)
            self._qtheta = self.transform(file)
        else:
            raise ValueError('Either qtheta or file and model must
be set')
```

```
    @classmethod
    def _set_thetas(cls, theta):
        if isinstance(theta, (pd.DataFrame, pd.Series)):
            if isinstance(theta, pd.Series):
                theta = pd.DataFrame(theta)
            return theta
        elif isinstance(theta, str):
            _, file_extension = os.path.splitext(theta)
            if file_extension == '.csv':
                return pd.read_csv(theta, index_col=0)
            elif file_extension == '.h5':
                return pd.read_hdf(theta, 'table')
            else:
                raise TypeError("Theta matrix must have either hdf
(.h5) or csv format. " +
                                "Instead                          got
'{}'".format(file_extension))
        else:
            raise TypeError('theta must be either a DataFrame/Series
object or a path to it')

    @property
    def qtheta(self):
        return self._qtheta

    @property
    def theta(self):
        return self._theta

    @qtheta.setter
    def qtheta(self, qtheta):
        self._qtheta = self._set_thetas(qtheta)

    @theta.setter
    def theta(self, theta):
        self._theta = self._set_thetas(theta)

    def get_measure(self, qtheta=None):
        """
            :Description: Gets cosine measure for the specified
query document\
                          and documents in the topic model

            :param qtheta: Theta distribution for a new document,
None means use one that the object has
            :type qtheta: csv or hdf (.h5) table

            :return: list with tuples of documents ids and their
corresponding cosine measure in descending order
        """
        if qtheta is not None:
            self.qtheta = self._set_thetas(qtheta)
```

```
        mera = dict()

        mera_down1 = pow(pow(self.qtheta.values[0], 2).sum(), 1/2)
        for row in range(len(self.theta.index)):
            mera_up                                                    =
self.qtheta.values[0].dot(self.theta.values[row])
            if mera_up == 0:
                continue
            mera_down2  =  pow(pow(self.theta.values[row],  2).sum(),
1/2)
            mera[self._theta.index[row]] = mera_up / (mera_down1 *
mera_down2)

        mera = sorted(mera.items(), key=itemgetter(1))
        mera.reverse()
        return mera

class QueryAdjust(CosineSimilarity):
    def __init__(self, theta, qtheta=None, file=None, model=None):
        """
           :param theta: Theta matrix of the topic model
           :type theta: csv or hdf (.h5) table
           :param qtheta: Theta distribution of the query document
           :type qtheta: csv or hdf (.h5) table
           :param str file: path to an unprepared query document
           :param model: topic model to get query Theta distribution
from, if None can't get\
                         distribution for a new document
           :type  model:  str  or  reference  to  BigARTM  artm  model
object
        """
        super(QueryAdjust,   self).__init__(theta,   qtheta,   file,
model)

    def _qurery_adjust(self, rev, irrev, koef):
        nrev = len(rev)
        nirrev = len(irrev)
        n = nrev + nirrev
        if koef is None:
            koef = nrev/nirrev if nirrev > nrev else nirrev/nrev
        q_source       =       pd.Series(data=self.qtheta.values[0],
index=self.qtheta.columns)
        q_rev    =    pd.Series(data=np.zeros(self.qtheta.shape[1]),
index=self.qtheta.columns)
        q_irrev   =   pd.Series(data=np.zeros(self.qtheta.shape[1]),
index=self.qtheta.columns)

        for relev in rev:
            q_rev = q_rev.add(self.theta.loc[relev]).multiply(koef *
nirrev / n)
        for irrelev in irrev:
```

```
            q_irrev                                                    =
q_irrev.add(self.theta.loc[irrelev]).multiply(koef * nrev / n)

        res = q_source.add(q_rev).subtract(q_irrev)
        res = (res - res.min()) / (res.max() - res.min())
        res = res.divide(res.sum())
        return pd.DataFrame(res).transpose()

    def adjust(self, rev, irrev, qtheta=None, koef=None):
        """
           :Description: Gets cosine measure for the specified query
document\
                         with adjusted to rev and irrev topic
distribution and documents in the topic model

           :param list rev: ids of the relevant documents
           :param list rev: ids of the irrelevant documents
           :param qtheta: Theta distribution for a new document,
None means use one that the object has
           :type qtheta: csv or hdf (.h5) table

           :return: list with tuples of documents ids and their
corresponding cosine measure in descending order
        """
        if qtheta is not None:
            self.qtheta                    =                    qtheta
# does it call _set_thetas or not?
        mera   =   self.get_measure(self._qurery_adjust(rev,   irrev,
koef))
        return mera


class PredictiveAdjust(CosineSimilarity):
    def __init__(self, theta, qtheta=None, file=None, model=None):
        """
             :param theta: Theta matrix of the topic model
             :type theta: csv or hdf (.h5) table
             :param qtheta: Theta distribution of the query document
             :type qtheta: csv or hdf (.h5) table
             :param str file: path to an unprepared query document
             :param model: topic model to get query Theta
distribution from, if None can't get\
                           distribution for a new document
             :type model: str or reference to BigARTM artm model
object
        """
        super(PredictiveAdjust, self).__init__(theta, qtheta, file,
model)

    def _regression_data(self, rev, irrev, mera, confidence):
        if confidence is None:
            confidence = 5e2
```

```
        items = len(mera)
        x = np.ones((items, self.theta.shape[1]))
        y = np.zeros(items)
        weights = np.ones(items)
        n = (len(rev) + len(irrev))
        koef_rev = len(irrev) / n
        koef_irrev = len(rev) / n

        for i, item in enumerate(mera):
            x[i] = self.theta.loc[item[0]].values
            if item[0] in rev:
                y[i] = 1
                weights[i] = confidence * koef_rev
            elif item[0] in irrev:
                y[i] = 0
                weights[i] = confidence * koef_irrev
        return x, y, weights

    def _logistic_regression_adjust(self, x, y, weights):
        fitting_index = weights > 1
        logistic = LogisticRegression()
        logistic.fit(x[fitting_index],              y[fitting_index],
weights[fitting_index])
        return logistic.predict_proba(x)

    def   adjust(self,   rev,   irrev,   mera=None,   qtheta=None,
confidence=None):
        """
           :Description: Gets logistic regression model's prediction
regarding relevance of the\
                         document  in  topic  model  to  the  query
document

           :param list rev: ids of the relevant documents
           :param list rev: ids of the irrelevant documents
           :param mera: cosine measure of the document to predict
relevance of, None means\
                         predict for all document in the topic model
           :type mera: list of tuples with documents ids and their
corresponding cosine measure
           :param qtheta: Theta distribution for a new document,
None means use one that the object has
           :type qtheta: csv or hdf (.h5) table

           :return:
            *list with triple tuples with:
            *document ids;
            *corresponding cosine measure;
            *corresponding relevance prediction (from 0 to 1);
            sorted by relevance in descending order
        """
        if mera is None:
```

```
            mera = self.get_measure(qtheta)
        x, y, weights = self._regression_data(rev, irrev, mera,
confidence)
        predicted = self._logistic_regression_adjust(x, y, weights)
        predicted = [(*x,y[1]) for (x, y) in sorted(zip(mera,
predicted),
                                                    key=lambda pair:
pair[1][1], reverse=True)]
    return predicted
```

Модуль doc2VowWab.py

```
import nltk
import urllib.parse as urlparse
import collections
import time
import sys
import pymorphy2
import os
import json
from bs4 import BeautifulSoup
from subprocess import Popen
from nltk.corpus import stopwords
import multiprocessing as mp
from functools import partial

SUBLIME_PATH = r'C:\Program Files\Sublime Text 3\sublime_text.exe'
EMED_PATH = r'C:\Program Files\EmEditor\EmEditor.exe'
DATASET_PATH = r'habr\habr_posts\habr_posts'
FILE_BOW = r'text_test.txt'

morph = pymorphy2.MorphAnalyzer()
stop_words = set(stopwords.words('russian'))

def run_editor(file=None, path=SUBLIME_PATH, filename=None):
    with open('file_to_read.txt','w',encoding='utf-8') as fout:
        if type(file) == type(dict()):
            for item in file:
                print(item,':',file[item],file=fout)
        else:
            print(file,file=fout)
    handle = Popen([path,'file_to_read.txt'])

def get_post(post_id):
    with open(os.path.join(DATASET_PATH, str(post_id))) as fin:
        post = json.load(fin)
    return post

def post_to_corpus_line(post):
    post_id = post['_id']
    author = post['author']
```

```
    tags = post['tags']
    date = post['published'][:10]

    words = collections.Counter()
    soup = BeautifulSoup(post['content_html'], 'lxml')
    [x.extract() for x in soup.findAll('code')]
    content_text = soup.getText()

    space_chars = u'«»“”’„*…/_.\\<>"'+u"'"
    for c in space_chars:
        content_text = content_text.replace(c, ' ')
    tokens = nltk.word_tokenize(content_text)
    for token in tokens:
        if len(token) > 2:
            try:
                isnumb = float(token)
                continue
            except:
                token = token.lower().replace(u'ë', u'e')
                word = morph.parse(token)[0].normal_form
                if len(word) > 0 and word not in stop_words:
                    words[word] += 1

    def parse_hub_id(hub_pair, get_name=True):
        if get_name:
            hub_id = hub_pair[0].replace(' ','_').lower()
            return hub_id
        else:
            url_parts = list(filter(lambda s: len(s) > 0,

urlparse.urlsplit(hub_pair[1]).path.split('/')))
            if len(url_parts) >= 2:
                hub_id = '_'.join(url_parts[-2:])
                return hub_id

    hubs = []
    for hub_pair in post['hubs']:
        hub_id = parse_hub_id(hub_pair)
        if hub_id:
            hubs.append(hub_id)

    def construct_bow(words):
        return [
            (
                word.replace(' ', '_').replace(':', '_').replace('|',
'_').replace('\t', '_') +
                ('' if cnt == 1 else ':{}'.format(cnt))
            )
            for word, cnt in words.items()
        ]

    parts = (
```

```
        ['{}'.format(post_id)] +
        ['|date ' + date] +
        ['|author'] + construct_bow({author: 1} if author is not
None else {}) +
        ['|tags'] + construct_bow({tag: 1 for tag in tags}) +
        ['|hubs'] + construct_bow({hub_id: 1 for hub_id in hubs}) +
        ['|words'] + construct_bow(words)
    )
    return ' '.join(parts)

def print_time(cur_time, coll_size, ready_cnt):
    print(str(ready_cnt)+'/'+str(coll_size),                          '-
{0:.0f}%'.format(100*ready_cnt/coll_size))
    print('{0:.2f} seconds'.format(cur_time),
          '({2:.0f}h{0:.0f}m{1:.0f}s) '.format(cur_time//3600,
                                               (cur_time)//60%60,
                                               cur_time-
60*(cur_time//60)), end='')
    print('since begining...')


def listener(q, notif_rate, coll_size):
    ready_cnt = 0
    start = time.time()
    with open(FILE_BOW, 'w', encoding='utf-8') as fout:
        while 1:
            message = q.get()
            if message == 'kill':
                break
            print(message, file=fout)
            fout.flush()
            ready_cnt += 1
            if ready_cnt % (notif_rate) == 0:
                print_time(time.time()-start, coll_size, ready_cnt)
                sys.stdout.flush()

def worker(q, post_id):
    line = post_to_line(get_post(post_id))
    q.put(line)
    return line

def do_parallel(post_ids, notif_rate, coll_size):
    manager = mp.Manager()
    q = manager.Queue()
    func = partial(worker, q)
    pool = mp.Pool(mp.cpu_count()+1)

    watcher = pool.apply_async(listener, (q, notif_rate, coll_size))
    pool.map(func, post_ids)

    q.put('kill')
    pool.close()
```

```
def do_usual(post_ids, notif_rate, coll_size):
    start = time.time()
    with open(FILE_BOW, 'w', encoding='utf-8') as fout:
        ready_cnt = 0
        for post_id in post_ids:
            line = post_to_line(get_post(post_id))
            print(line, file=fout)
            ready_cnt += 1
            if ready_cnt % notif_rate == 0:
                print_time(time.time()-start, coll_size, ready_cnt)

def main(start=None, finish=None, parallel=True, notifications=20):
    post_ids      =      [int(filename)      for      filename      in
os.listdir(DATASET_PATH)
                if not filename.startswith('.')][start:finish]
    coll_size = len(post_ids)
    start_time = time.time()
    if parallel:
        do_parallel(post_ids, coll_size//notifications, coll_size)
    else:
        do_usual(post_ids, coll_size//notifications, coll_size)
    cur_time = time.time()-start_time

    print('\nTotal  time  ({0:.0f}  items)  from  '.format(coll_size),
end='')
    print('{0:.0f}  -  {1:.2f}  seconds  '.format(start,  cur_time),
end='')
    print('({0:.0f}h{1:.0f}m{2:.0f}s)'.format(cur_time//3600,
                                              (cur_time//60)%60,
                                              cur_time-
60*(cur_time//60)))
```

Модуль buildTM.py

```
from sys import path
if r'C:\BigARTM\Python' not in path:
    path.append(r'C:\BigARTM\Python')

import os
import glob
import artm
import time
import matplotlib.pyplot as plt

DATA_PATH = r'habr\habr_posts\no_code\texts135k.txt'
QUERY = r'habr\habr_posts\no_code\query.txt'
BATCHES = 'habr_batches'
QUERY_BATCHES = 'habr_query_batches'

def save_phi_theta(model, save_phi=False, save_theta=True):
    if save_phi:
```

```
        model.get_phi().transpose().to_hdf('phi_store.h5',   'table',
mode='w')
    if save_theta:
        model.get_theta().transpose().to_hdf('theta_store.h5',
'table', mode='w')

def   get_query_data(model,   predict_class=None,   query_file=QUERY,
save_theta_q=True,
                     save_classes=True, filename='thetaQ_store'):
    query_bv = artm.BatchVectorizer(data_path=query_file,
                                    data_format='vowpal_wabbit',
                                    batch_size=10,
                                    target_folder=QUERY_BATCHES,
                                    gather_dictionary=True)

    thetaQ = model.transform(batch_vectorizer=query_bv)
    if predict_class:
        classes     =     model.transform(batch_vectorizer=query_bv,
predict_class_id=predict_class)
    if save_theta_q:
        thetaQ.transpose().to_hdf(filename + '.h5', 'table')
    if predict_class is not None and save_classes:
        classes.sort_values(classes.columns[0]).to_csv(filename     +
'_classes.csv', sep='\t', encoding='utf-8')

def get_tokens(model, flname='res.txt'):
    tokens = model.score_tracker['top_tok'].last_tokens
    with open(flname, 'w', encoding='utf-8') as fl:
        for topic_name in model.topic_names:
            print(topic_name, file=fl)
            for token in tokens[topic_name]:
                print(token, end=' ', file=fl)
            print(file=fl)

def draw_data(model, ConPur=False):
    perp_values = model.score_tracker['PerplAll'].value[1:]
    perp_values_w = model.score_tracker['PerplWords'].value[1:]
    sparsity_phi = model.score_tracker['SparPhiAll'].value
    sparsity_phi_main = model.score_tracker['SparPhiMain'].value
    sparsity_phi_bckgr = model.score_tracker['SparPhiBckgr'].value
    sparsity_theta = model.score_tracker['SparThetAll'].value
    sparsity_theta_main = model.score_tracker['SparThetMain'].value
    sparsity_theta_bckgr                                              =
model.score_tracker['SparThetBckgr'].value

    plt.ion()
    # Perp
    fig, axs = plt.subplots(3, 2)
    fig.suptitle('Перплексия')
    axs[0, 0].set_title('All', fontsize=10)
    axs[0, 1].set_title('Words', fontsize=10)
    for i in range(3):
```

```
        axs[i,      0].plot(range(len(perp_values[i      *      5:])),
perp_values[i * 5:])
        axs[i, 0].grid(True)
        axs[i,     1].plot(range(len(perp_values_w[i     *     5:])),
perp_values_w[i * 5:])
        axs[i, 1].grid(True)
    plt.show()
    # SparPhi
    fig, axs = plt.subplots(3)
    fig.suptitle('SparPhi')
    axs[0].plot(range(len(sparsity_phi)), sparsity_phi)
    axs[1].plot(range(len(sparsity_phi_main)), sparsity_phi_main)
    axs[2].plot(range(len(sparsity_phi_bckgr)), sparsity_phi_bckgr)
    for i in range(3):
        axs[i].grid(True)
    plt.show()
    # SparTheta
    fig, axs = plt.subplots(3)
    fig.suptitle('SparTheta')
    axs[0].plot(range(len(sparsity_theta)), sparsity_theta)
    axs[1].plot(range(len(sparsity_theta_main)),
sparsity_theta_main)
    axs[2].plot(range(len(sparsity_theta_bckgr)),
sparsity_theta_bckgr)
    for i in range(3):
        axs[i].grid(True)
    plt.show()
    if ConPur:
        contrast                                                =
model.score_tracker['TopicKernelScore'].average_contrast
        purity                                                  =
model.score_tracker['TopicKernelScore'].average_purity
        plt.figure(num=4)
        plt.plot(range(len(contrast)), contrast)
        plt.title('Contrast')
        plt.grid(True)
        plt.figure(num=5)
        plt.plot(range(len(purity)), purity)
        plt.title('Purity')
        plt.grid(True)
        plt.show()

try:
    if len(glob.glob(os.path.join(BATCHES, '*.batch'))) == 0:
        start_bv = time.time()
        bv = artm.BatchVectorizer(data_path=DATA_PATH,
                                  data_format='vowpal_wabbit',
                                  batch_size=1000,
                                  target_folder=BATCHES,
                                  gather_dictionary=True)
        end_bv = time.time() - start_bv
```

```
        print('Batches in {0:.0f}m{1:.0f}s'.format((end_bv // 60) %
60, end_bv % 60, ))
        dictionary = bv.dictionary
    else:
        bv = artm.BatchVectorizer(data_path=BATCHES,
                                  data_format='batches',
                                  gather_dictionary=True)
        dictionary = artm.Dictionary()
        dictionary.load(dictionary_path=os.path.join(BATCHES,
'dictionary.dict'))

    if not os.path.isfile(os.path.join(BATCHES, 'dictionary.dict')):
        dictionary.gather(data_path=bv.data_path)
        dictionary.save(dictionary_path=os.path.join(BATCHES,
'dictionary.dict'))

    topics = ['topic_' + str(i) for i in range(250)]
    background_topics = topics[:50]
    main_topics = topics[50:]

    doc_passes = 5

    model = artm.ARTM(topic_names=topics,
                      cache_theta=True,
                      dictionary=dictionary,
                      show_progress_bars=True,
                      num_document_passes=doc_passes,
                      class_ids={'words': 1.0, 'hubs': 10.0},
                      theta_columns_naming='title',
                      theta_name='Habr_theta')
    ## Regularaizers
    # SSPhi

model.regularizers.add(artm.SmoothSparsePhiRegularizer(name='SSPhiMa
in', topic_names=main_topics, tau=0.01))
    model.regularizers.add(
        artm.SmoothSparsePhiRegularizer(name='SSPhiBckgr',
topic_names=background_topics, tau=0.015))
    # SSTheta

model.regularizers.add(artm.SmoothSparseThetaRegularizer(name='SSThe
taMain', topic_names=main_topics, tau=0.01))
    model.regularizers.add(
        artm.SmoothSparseThetaRegularizer(name='SSThetaBckgr',
topic_names=background_topics, tau=0.015))
    # DecorrelatorPhi
    model.regularizers.add(
        artm.DecorrelatorPhiRegularizer(name='DecorrelatorPhiMain',
topic_names=main_topics, tau=1e+3))
    model.regularizers.add(
        artm.DecorrelatorPhiRegularizer(name='DecorrelatorPhiBckgr',
topic_names=background_topics, tau=1e+2))
```

```
    ## Scores
    # Perplexity
    model.scores.add(artm.PerplexityScore(name='PerplAll',
dictionary=bv.dictionary, ))
    model.scores.add(artm.PerplexityScore(name='PerplWords',
class_ids='words', dictionary=bv.dictionary, ))
    # SparPhi
    model.scores.add(artm.SparsityPhiScore(name='SparPhiAll',
class_id='words'))
    model.scores.add(artm.SparsityPhiScore(name='SparPhiMain',
topic_names=main_topics, class_id='words'))
    model.scores.add(artm.SparsityPhiScore(name='SparPhiBckgr',
topic_names=background_topics, class_id='words'))
    # SparThet
    model.scores.add(artm.SparsityThetaScore(name='SparThetAll'))
    model.scores.add(artm.SparsityThetaScore(name='SparThetMain',
topic_names=main_topics))
    model.scores.add(artm.SparsityThetaScore(name='SparThetBckgr',
topic_names=background_topics))
    # Other
    model.scores.add(artm.TopTokensScore(name='top_tok',
num_tokens=10, class_id='words'))
    model.scores.add(artm.TopicKernelScore(name='TopicKernelScore',
                                           class_id='words',
                                           topic_names=main_topics,

probability_mass_threshold=0.5))
    ## Fitting
    model.fit_offline(batch_vectorizer=bv, num_collection_passes=15)
    model.dump_artm_model('250_topics_15itr')
    get_tokens(model, 'res1.txt')

    model.regularizers['SSThetaMain'].tau = 0
    model.regularizers['SSPhiMain'].tau = 0
    model.regularizers['DecorrelatorPhiMain'].tau = 1e+4

    model.fit_offline(batch_vectorizer=bv, num_collection_passes=20)
    model.dump_artm_model('250_topics_35itr')
    get_tokens(model, 'res2.txt')

    alpha_iter = [(10 - x) / 10 for x in range(doc_passes)]
    alpha_iter.reverse()
    model.regularizers['SSThetaMain'].tau = -0.01
    model.regularizers['SSThetaMain'].alpha_iter = alpha_iter
    model.regularizers['SSPhiMain'].tau = -0.01

model.regularizers.add(artm.ImproveCoherencePhiRegularizer(name='Imp
CohPhi',

tau=1.5,
```

```
class_ids='words',

dictionary=dictionary))

    model.fit_offline(batch_vectorizer=bv, num_collection_passes=35)
    model.dump_artm_model('250_topics_70itr')

    save_phi_theta(model, save_phi=False)
    get_tokens(model)
    draw_data(model, ConPur=True)
except e:
    raise e
```

Модуль testSearch.py

```
import pandas as pd
import numpy as np
import operator
import time
import sys
import multiprocessing as mp
from math import log10
from sklearn.linear_model import Ridge
from io import StringIO
from csv import writer
from functools import partial
from sklearn.linear_model import LogisticRegression

QUERY_PATH = r'queries\thetaQ_hexapod_beg.h5'
THETA_PATH = r'theta_store.h5'
KOEFS_LOG = [1.00E+01, 1.50E+01, 2.50E+01, 5.00E+01,
         1.00E+02, 2.00E+02, 5.00E+02, 1.00E+03, 1.00E+05, 5.00E+05]
KOEFS_ADJ = [None, 0.1, 0.25, 0.5, 0.75, 0.1, 1, 1.25, 1.5, 1.75]
GLOB_REV  =  ['156579',  '228969',  '225845',  '135529',  '142740',
'133843', '209062', '170711',

'94012','81322','135570','96452','148964','123283','180571','174609'
,'222535','5598',

'149038','218379','222599','198456','118532','156027','155535','1801
67',
     '115334', '194068']

def timeit(method):
    def timed(*args, **kw):
        ts = time.time()
        result = method(*args, **kw)
        te = time.time()

        print('{} completed in {:.2f}s'.format(method.__name__, (te
- ts)))
```

```
        return result
    return timed

def swaptheta(name, theta):
    thetaQ = pd.read_hdf(r'queries\thetaQ_'+name+'.h5','table')
    thetaQ = thetaQ[[col for col in thetaQ.columns[50:]]]
    mera = cosin(theta, thetaQ)
    return thetaQ, mera

def cosin(theta, thetaQ_in):
    thetaQ = pd.DataFrame(thetaQ_in)
    mera = {}
    start = time.time()
    meraDOWN1= pow(pow(thetaQ.values[0], 2).sum(), 1/2)
    for row in range(len(theta.index)):
        meraUP = thetaQ.values[0].dot(theta.values[row])
        if meraUP == 0:
            continue
        meraDOWN2 = pow(pow(theta.values[row], 2).sum(), 1/2)
        mera[theta.index[row]] = meraUP/(meraDOWN1*meraDOWN2)

    mera = sorted(mera.items(), key=operator.itemgetter(1))
    mera.reverse()
    end = time.time() - start
    print('Calculate in {0:.2f}s...'.format(end))
    return mera

######### adjust ############# start
def adjust_query(rev, irrev,theta, thetaQ, koef=None):
    nrev = len(rev)
    nirrev = len(irrev)
    if koef is None:
        if nrev*nirrev != 0:
            koef = nrev/nirrev if nirrev > nrev else nirrev/nrev
        else:
            koef = 1
    n = nrev + nirrev
    Qorig = pd.Series(data=thetaQ.values[0],index=thetaQ.columns)
    Qrev                                                          =
pd.Series(data=np.zeros(thetaQ.shape[1]),index=thetaQ.columns)
    Qirrev                                                        =
pd.Series(data=np.zeros(thetaQ.shape[1]),index=thetaQ.columns)
    for relev in rev:
        Qrev = Qrev.add(theta.loc[relev]).multiply(koef*nirrev/n)
    for irrelev in irrev:
        Qirrev                                                    =
Qirrev.add(theta.loc[irrelev]).multiply(koef*nrev/n)
    res = Qorig.add(Qrev).subtract(Qirrev)
    res = (res - res.min()) / (res.max() - res.min())
    res = res.divide(res.sum())
    return pd.DataFrame(res).transpose()
```

```
def muda(mera_glob, theta_glob, thetaQ_glob, rev, irrev, koef):
    theta = theta_glob.loc[[item[0] for item in mera_glob]]
    thetaQ = adjust_query(rev, irrev, theta_glob, thetaQ_glob, koef)
    mera = cosin(theta, thetaQ)
    docs = 20
    printed = print_adjust(mera, rev, irrev, thetaQ_glob.index[0])
    return printed

def print_adjust(mera, rev, irrev, doc_name, entries=20):
    to_print = []
    for item in mera:
        if len(to_print) == entries:
            break
        if item[0] not in rev and item[0] not in irrev:
            to_print.append(item)
    print('\nTop {} docs with max cosin for {}:'.format(entries,
doc_name))
    print(*to_print, sep='\n')
    return [item[0] for item in to_print]

def get_adjust_irrev(printed, GLOB_REV, frequency=1):
    irrev = []
    for item in printed:
        if item not in GLOB_REV:
            irrev.append(item)
    return irrev[::frequency]

def check_adjust(printed, GLOB_REV, rev, irrev):
    _rev = []
    for item in printed:
        if item in GLOB_REV:
            _rev.append(item)
    print('\nRelevant docs here [{}]:'.format(len(_rev)))
    print(*_rev, sep='\n')
    print('\nrev/irrev - {}/{}'.format(len(rev),len(irrev)))
    return _rev

def iterate_adjust(theta, thetaQ, mera100, rev, irrev, GLOB_REV,
koef, frequencyIREV, frequencyREV):
    printed = muda(mera100, theta, thetaQ, rev, irrev, koef)
    new_rev = check_adjust(printed, GLOB_REV, rev, irrev)
    rev.extend(new_rev[::frequencyREV])
    irrev.extend(get_adjust_irrev(printed, GLOB_REV, frequencyIREV))
    return rev, irrev

def adjust_forme(theta, thetaQ, mera, koefs, GLOB_REV, frequencyIREV,
frequencyREV):
    steps = {}
    for koef in koefs:
        rev, irrev = [], []
        steps[koef if koef is not None else 'default'] = []
        while(len(rev) < len(GLOB_REV)):
```

```
                rev, irrev = iterate_adjust(theta, thetaQ, mera, rev,
irrev, GLOB_REV, koef, frequencyIREV, frequencyREV)
                steps[koef    if    koef    is    not    None    else
'default'].append((len(rev), len(irrev)))
    print('\n###### QEURY ADJUSTS RESULTS ######')
    for key in steps:
        print('\nFor    koef    {}    ({}    iterations):'.format(key,
len(steps[key])))
        for item in steps[key]:
            print('{}/{}'.format(item[0], item[1]))
######### adjust ############# end

######### logistic ############# start
def data_logistic(theta, mera, rev, irrev, confidence=1e15):
    items = len(mera)
    _x = np.ones((items, 200))
    _y = np.ones(items)
    weights = np.ones(items)
    n = (len(rev) + len(irrev))
    koef_rev = len(irrev) / n
    koef_irrev = len(rev) / n
    for i, item in enumerate(mera):
        _x[i] = theta.loc[item[0]].values
        if item[0] in rev:
            _y[i] = 1
            weights[i] = confidence * koef_rev
        elif item[0] in irrev:
            _y[i] = 0
            weights[i] = confidence * koef_irrev
    return _x, _y, weights

def logistic(theta, mera, rev, irrev, confidence):
    _x,  _y,  weights  =  data_logistic(theta,  mera,  rev,  irrev,
confidence)
    fitting_index = weights > 1
    logistic = LogisticRegression()
    logistic.fit(_x[fitting_index],                _y[fitting_index],
weights[fitting_index])
    return logistic.predict_proba(_x), _x, _y, weights

def print_logistic(predicted, mera, rev, irrev, GLOB_REV, entries=20,
forme=True):
    new_mera = [x for (x, y) in sorted(zip(mera, predicted),
                                       key=lambda  pair:  pair[1][1],
reverse=True)]
    to_print = []
    for item in new_mera:
        if item[0] in rev or item[0] in irrev:
            continue
        if len(to_print) == entries:
            break
        to_print.append(item)
```

```
    new_rev = []
    for item in to_print:
        if item[0] in GLOB_REV:
            new_rev.append(item)
    if not forme:
        print(*to_print, sep='\n')
        print('\nNew relevant documents here
[{}]:'.format(len(new_rev)))
        print(*new_rev, sep='\n')
        print('rev/irrev - [{}/{}]'.format(len(rev), len(irrev)))
    return to_print, new_rev

def first_logistic(mera, GLOB_REV, rev_n, entries, frequency):
    rev = GLOB_REV[:rev_n]
    irrev = get_irrev_logistic(mera[:entries], rev, [], frequency)
    return rev, irrev

def iterate_logistic(theta, mera, rev, irrev, frequency, confidence,
entries, forme=False):
    predicted, x, y, w = logistic(theta, mera, rev, irrev,
confidence)
    # check_logistic(c, y, mera, GLOB_REV, detailed)
    printed, new_rev = print_logistic(predicted, mera, rev, irrev,
GLOB_REV, entries, forme)
    rev.extend([item[0] for item in new_rev])
    if forme:
        print(len(new_rev), end=' ')
    irrev = get_irrev_logistic(printed, rev, irrev, frequency)
    return rev, irrev

def get_irrev_logistic(printed, rev, irrev, frequency=1):
    new_irrev = []
    for item in printed:
        if item[0] not in rev:
            new_irrev.append(item[0])
    irrev.extend(new_irrev[::frequency])
    return irrev

def check_logistic(c, y, mera, GLOB_REV, detailed=False):
    buf1 = []
    buf2 = {'in c, not in glob': [],
            'in glob not in c': [],
            'in c and glob not in rev': []}
    for item in c.nonzero()[0]:
        if item not in y.nonzero()[0]:
            buf1.append(mera[item])
            if mera[item][0] in GLOB_REV:
                buf2['in c and glob not in rev'].append(mera[item])
        if mera[item][0] not in GLOB_REV:
            buf2['in c, not in glob'].append(mera[item])
    print('Items in c but not in y [{}]:'.format(len(buf1)))
```

```
    if detailed: print(*buf1, sep='\n')

    buf3 = []
    for item in y.nonzero()[0]:
        if item not in c.nonzero()[0]:
            buf3.append(mera[item])
    print('\nItems in y but not in c (forgotten) [{}]:'.format(len(buf3)))
    if detailed: print(*buf3, sep='\n')

    for item in GLOB_REV:
        if item not in list(map(lambda x: mera[x][0], c.nonzero()[0])):
            buf2['in glob not in c'].append(item)

    print('\nItems in c but not in glob (incorrectly predicted) [{}]:'.format(len(buf2['in c, not in glob'])))
    if detailed: print(*buf2['in c, not in glob'], sep='\n')
    print('\nItems in glob but not in c (not predicted) [{}]:'.format(len(buf2['in glob not in c'])))
    if detailed: print(*buf2['in glob not in c'], sep='\n')
    print(
        '\nItems in c and glob but not in y (correctly predicted) [{}]:'.format(len(buf2['in c and glob not in rev'])))
    if detailed: print(*buf2['in c and glob not in rev'], sep='\n')
    print('\nItems in c - [{}]'.format(c.nonzero()[0].size))

def log_forme(theta, koefs, mera, GLOB_REV, n_rev, entries, frequency, no_det=True):
    print('\n###### LOG ADJUSTS RESULTS ######')
    for koef in koefs:
        koef_str = '{:E}'.format(koef)
        koef_str = koef_str.split('E')[0].rstrip('0').rstrip('.') + 'e' + koef_str.split('E')[1]
        print('For koef [{}] :'.format(koef_str))
        rev, irrev = first_logistic(mera, GLOB_REV, n_rev, entries, frequency)
        while len(rev) < len(GLOB_REV):
            rev, irrev = iterate_logistic(theta, mera, rev, irrev, frequency, koef, entries, no_det)
        print('\n')
######### logistic ############ end|

def usual_check(mera, GLOB_REV):
    print('\nCosine measure distribution:')
    for i in range(0,100,20):
        cnt = 0
        for item in mera[i:i+20]:
            if item[0] in GLOB_REV:
                cnt += 1
        print(cnt)
```

```
def main():
    start_glob = time.time()
    thetaQ = pd.read_hdf(QUERY_PATH,'table')
    theta = pd.read_hdf(THETA_PATH,'table')
    theta = theta[[col for col in theta.columns[50:]]]
    thetaQ = thetaQ[[col for col in thetaQ.columns[50:]]]
    end = time.time() - start_glob
    print('Read matrices in {0:.2f}s...'.format(end))

    mera = cosin(theta, thetaQ)
    adjust_forme(theta, thetaQ, mera[:100], KOEFS_ADJ, GLOB_REV, 1,
1)
    log_forme(theta, KOEFS_LOG, mera[:100], GLOB_REV, 6, 20, 1)

if __name__ == '__main__':
    main()
```