Правительство Российской Федерации

Федеральное государственное автономное

Образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет

«Высшая школа экономики».

Факультет бизнес-информатики

Кафедра бизнес-аналитики

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

На тему

Проектирование и разработка хранилища данных для статистического анализа текстов

Студент группы № 471

Захаров С.А.

Научный руководитель

к.т.н., доцент Дружаев А.А.

 Рецензент

к.т.н., доцент Перминов Г.И.

Москва 2013

Содержание

[Глава I – Проблема статистического анализа текстов 3](#_Toc357618139)

[1. Актуальность проблемы 3](#_Toc357618140)

[2. Постановка задачи 5](#_Toc357618141)

[3. Определения 5](#_Toc357618142)

[3.1 Анализ текста 6](#_Toc357618143)

[3.2 Текстовые данные 6](#_Toc357618144)

[3.3 Статистический 6](#_Toc357618145)

[4. Классификация аналитических задач 7](#_Toc357618146)

[4.1 Анализ временных рядов 7](#_Toc357618147)

[4.2 Классификация 8](#_Toc357618148)

[4.3 Кластеризация 9](#_Toc357618149)

[5. Методы решения 10](#_Toc357618150)

[5.1 Классификация 11](#_Toc357618151)

[5.2 Кластеризация 13](#_Toc357618152)

[6. Общие принципы 15](#_Toc357618153)

[6.1 Частотные показатели 16](#_Toc357618154)

[6.2 Временные ряды 17](#_Toc357618155)

[7. Исследовательское хранилище данных 17](#_Toc357618156)

[8. Процесс анализа текстов 21](#_Toc357618157)

[8.1 Получение набора данных 21](#_Toc357618158)

[8.2 Предварительная обработка текста 21](#_Toc357618159)

[8.3 Сохранение преобразованных документов в хранилище 23](#_Toc357618160)

[8.4 Применение аналитических методов 23](#_Toc357618161)

[8.5 Интерпретация результатов 23](#_Toc357618162)

[Глава II – Проектирование и разработка хранилища 24](#_Toc357618163)

[1. Концептуальная модель 24](#_Toc357618164)

[2. Логические модели 27](#_Toc357618165)

[2.1 Эксперимент 28](#_Toc357618166)

[2.2 Реляционные таблицы 29](#_Toc357618167)

[2.3 Документно-ориентированный подход 38](#_Toc357618168)

[3. Сведение результатов 42](#_Toc357618169)

[4. Заключение 43](#_Toc357618170)

[Список источников 44](#_Toc357618171)

[Приложение 46](#_Toc357618172)

[Список стоп-слов: 46](#_Toc357618173)

[Список символов разбивки: 46](#_Toc357618174)

# Глава I – Проблема статистического анализа текстов

## 1. Актуальность проблемы

 Текстовая информация – одна из неотъемлимых составляющих практически любой сферы человеческой деятельности. В форме текста могут храниться сведения о чём угодно, текстовые сообщения используются для переписки, текстовые документы – неотъемлимая часть предпринимательства, истории болезней представляют собой последовательные записи, любая статья в СМИ обязательно содрежит хоть немного текста. Весь этот огромный массив разнородной информации несомненно содержит в себе некие скрытые, так или иначе полезные знания, которые из-за объёма источников данных не под силу найти вручную, каким бы старательным и трудолюбивым не был аналитик или команда аналитиков. Идея машинной обработки и анализа текстовой информации появилась ещё в 60-е годы прошлого века. Вот выдержка из статьи, опубликованной в октябре 1958 года журналом IBM Journal, в которой даётся следующее определение системы бизнес-аналитики, как системы, которая будет:

«...использовать машинную обработку данных для автореферирования и автокодирования документов и создания тематических профилей для каждого «делового узла» организации. Как поступающие извне, так и созданные внутри организации документы автоматически реферируются, классифицируются на основе определённых словосочетаний и отправляются в соответствующий узел.».

Но машинные методы анализа могут работать только со структурированными данными, а текст, написанный на естественном языке, к данной категории не относится, поэтому долгие десятилетия почти все усилия бизнес-аналитики были направлены на создание и совершенствование методов анализа числовой или какой-либо другой информации с чётко определённой структурой, поддающейся компьютерной обработке. Только в конце 90-х на основе серьёзных теоретических исследований, в результате которых было создано отдельное направление- так называемый, **Text Mining**, появились первые попытки создать информационную систему для анализа текстов.

 В настоящее время **Text Mining –** хорошо развитое направление компьютерной информационной аналитики, в рамках которого созданы мощные инструменты анализа и разработаны достаточно мощные методы анализа. Помимо технологий машинной обработки текста, в последние два десятилетия быстрыми темпами развивались и технологии хранения данных вообще, и, как ответ на нужды больших копораций в едином, интегрированном месте хранения и обработки корпоративных данных, появились так называемые хранилища данных. По определению Билла Инмона, хранилище данных – это субъектно-ориентированный, интегрированный, стабильны й, изменяемый во времени набор данных для принятия управленческих решений [6 стр. 7]. Анализ текстов имеет прямое отношение к хранилищам данных, так как более 80% всей корпоративной информации составляют текстовые документы, и в них содержится ценная для принятия решений информация. На пересечении двух развивающихся направлений: машинная обработка текстов и проектирование хранилищ данных и появляется концепция **хранилища данных для статистического анализа текстов**.

## 2. Постановка задачи

 Эта работа описывает процесс проектирования и разработки исследовательского хранилища данных, предназначенного для статистического анализа текстов. Под такое определение, в силу разнообразия данных и подходов к анализу, попадает обширный класс задач, из-за чего поиск универсального метода их решения сложен и трудоёмок. В более конкретном случае, например, когда досконально известно, какие данные хранятся, и для чего они хранятся, можно было бы спроектировать идеальное хранилище для решения именно этой задачи, но специфика такого хранилища существенно затруднила бы процесс доработки, например, в связи с появлением нового источника данных или необходимости применить новый, более эффективный метод анализа. Выходом из этой ситуации может стать попытка выявить общие принципы хранения текстовых данных и статистических методов анализа, и спроектировать хранилище на основе этих принципов. По сути, необходимо выяснить, какие данные понадобится хранить, и какие запросы аналитические приложения будут отправлять хранилищу, и организовать его так, чтобы эти запросы выполнялись быстрее и их было легче писать. Именно такой попыткой и является моя работа, результатом которой будут концептуальная схема хранилища данных и его программная реализация.

## 3. Определения

 Для конкретизации решаемой задачи необходимо дать более точные определения некоторым понятиям, о которых пойдёт речь в работе:

### 3.1 Анализ текста

 Анализом текста в рамках этой работы называется процесс исследования текстовых данных и обнаружение сокрытых в них знаний с помощью специальных алгоритмов. Знания должны быть новыми, нетривиальными, практически полезными и доступными для интерпретации. [7, стр. 194]

### 3.2 Текстовые данные

 Под текстовыми данными подразумеваются любые неструктурированные или слабо структурированные тексты на естественном языке. Например: книги, статьи, электронные письма, медицинские карты, новостные заметки или финансовые документы. Обычно, тексты, подлежащие анализу, логически объединены неким общим признаком, которым может выступать предметная область, язык, автор или источник, но это условие не всегда выполняется [6, стр. 34].

### 3.3 Статистический

 Так как значительная часть процесса обработки данных ведётся компьютерной информационной системой, все методы анализа должны быть строго определены и программно реализуемы. Такими методами в данной работе выступают методы математической статистики, которые служат для количественного описания свойств набора данных посредством расчёта статистических показателей или вывода некоторых статистических суждений относительно набора данных. [1, стр.4]

## 4. Классификация аналитических задач

 Все задачи, для решения которых применяются методы статистического анализа, можно разделить на 4 основных категории:

* 1. Задачи расчёта простых статистических показателей. Например, максимума, минимума, среднего, математического ожидания, дисперсии и т.п.;
	2. Задачи, реузльтат решения которых представляет ценность сам по себе. Обычно, такие задачи легко сформулируются одним вопросительным предложением. Например, «Каково процентное соотношение частей речи в наборе документов?» или «Как менялась частота употребления слова «нефть» в новостных сообщениях за последние 5 лет?»;
	3. Задачи анализа временных рядов;
	4. Задачи, результат которых сам по себе ценности не представляет, но может быть использован для решения более сложных задач. К классу более сложных задач, которые решаются методами статистического анализа, относятся [1, 2, 3]:
		+ 1. Классификация;
			2. Кластеризация;

### 4.1 Анализ временных рядов

 Анализ временных рядов в большинстве случаев решает три основных задачи:

1. Выявление структуры временного ряда
2. Выявление связи значений нескольких временных рядов
3. Прогнозирование значений временного ряда

 Первые два пункта необходимы для построения математической модели, описывающей область, характеристики которой отражены в анализируемых временных рядах. Такая модель может быть использована как для объяснения предыдущих изменений и колебаний показателей области, так и для предсказания их будущих значений.

 Применимо к теме данной работы такой анализ мог бы дать ответы на следующие примерные вопросы: Прослеживается ли в изменении частоты употребления слова t какая-нибудь закономерность? Как связана частота появления слова t1 с частотой появления слова t2? Каким уравнением описывается частота появления того или иного слова? Как изменится частота появления тех или иных слов в будущем?

### 4.2 Классификация

 Задачей классификации является определение одной или нескольких заранее определённых категорий, к которым относится документ. В частном случае, эта задача сводится к определению темы документа. Например, каждой новостной заметке присваиваются различные темы, вроде «спорт», «политика», «культура» и т.п. В более строгом, математическом смысле это задача поиска достаточно точной классификационной модели: f : D → L; f(d) = L, где D – набор исследуемых документов, d – один из набора документов, L – множество категорий или тем, f – функция-классификатор. [1, стр. 10]; см. рис. 5.



 *Рис. 5- Классификация документов*

### 4.3 Кластеризация

Кластеризация это разбиение всего множества документов, на отдельные подмножества или кластеры близких по содержанию документов. Результатом кластеризации является набор кластеров P, каждый из которых содержит документы из множества D – множества всех исследуемых документов. Чем более схожи по содержанию документы внутри кластера, и чем больше разница между содержанием документов из разных кластеров, тем выше качество кластеризации. Группировка документов проводится на основании их взаиморасположения в пространстве документов [1, стр. 15]; см. рис. 6.

**

*Рис. 6 – Кластеризация документов*

## 5. Методы решения

 Для решения задач первой и второй категории, зачастую, бывает достаточно выполнить всего один запрос, который фильтрует, группирует и аггрегирует данные нужным исследователю образом. Задачи из третьей и четвёртой категорий решаются с помощью сложных, многоэтапных математических методов. Существует несколько хорошо изученных и проверенных подходов к решению этих задач, и большинство новых методов чаще всего являются их улучшением и, соответственно, используют, в основном, те же данные, что и старые методы. Этот путь развития методов статистического анализа и позволяет предположить, что выделив основные принципы работы классических методов решения, можно тем самым удовлетворить потребности и их улучшенных или изменённых версий.

Нет смысла приводить в данной работе все методы анализа временных рядов, потому что и без их описания можно сделать вывод, что все они опираются на одни и те же данные: если говорить о наборе текстовых документов – временные ряды, характеризующие изменения частоты употребления определённых слов или слов с определёнными характеристиками (например, прилагательных). Можно составить следующий список методов решения задач классификации и кластеризации, к каждому из которых далее даётся краткое описание принципа работы:

**Классификация:**

1. Байесовский классификатор (**Naȉve Bayes Classifier**)
2. Метод k ближайших соседей (**k nearest neighbor classification**)
3. Дерево принятия решений (**Decision tree**)
4. Метод опорных векторов (**Support Vector Machines**)

**Кластеризация:**

1. Иерархические методы
2. Метод k средних (**k means**)
3. Сечение пополам методом k средних (**Bi-section k means**)
4. Метод самоорганизующихся карт (**Self-organizing maps**)
5. Метод максимального правдоподобия (**EM-algorithm**)

### 5.1 Классификация

#### 5.1.1 Байесовский классификатор

 При построении классификатора делается предположение, что слова отдельного документа зависят от некоторого вероятностного механизма распределения, а категория документа, в свою очередь, связана со словами, встречающимися в документе. Далее строится формула, описывающая эту зависимость, которая возвращает вероятность того, что документ, включающий определённые слова, относится к той или иной категории. В итоге документу присваивается категория, для которой формула выдала наибольшую вероятность.

#### 5.1.2 Метод k ближайших соседей

Этот метод, вместо того, чтобы строить формулу зависимости категории документа от его содержания, берёт за основу то, что похожие документы относятся к одним и тем же категориям. Существует множество разнообразных показателей схожести текстов, чаще всего основанных на вхождении в сравниваемые документы одних и тех же слов. Схожесть документов рассчитывается для всего корпуса, затем выбираются k наиболее схожих документов. Всем неклассифицированным документам в этой группе присваевается наиболее часто встречаемая в группе категория.

#### 5.1.3 Дерево принятия решений

Дерево принятия решений – это особый тип классификатора, основанный на последовательном применении заранее определённых правил классификации к набору документов. Правила классификации выводят следующим образом: в тренировочном наборе документов M выбирается слово, которое лучше всех прочих может определить категорию документов. Весь набор разбивается на 2 новых – M+ и M- соответственно, содержащие и не содержащие выбранное слово документы. Процедура рекурсивно повторяется для новых групп, пока документы во всех полученных группах не будут принадлежать к одной и той же категории.

#### 5.1.4 Метод опорных векторов

 Все классифицируемые документыпредставлены в виде векторов – частот слов, входящих в документ (подробнее см. Моделирование:концепутальная модель). Вектора-документы распределяются в общем пространстве документов, и метод пытается найти в этом пространстве такую плоскость, суммарное расстояние от которой до крайних документов по обе стороны плоскости, было бы максимальным. Эта плоскость и делит документы на две разных категории.

### 5.2 Кластеризация

#### 5.2.1 Иерархические методы

Все иерархические методы кластеризации основаны на разбиении всех документов на кластеры, образующие иерархию. Иерархия формируется или снизу вверх или сверху вних. В первом случае каждый документ изначально представляет собой отдельный кластер. Далее, наиболее схожие (расположенные ближе всего друг к другу) кластеры объединяются, формируя новые кластеры. Работа метода завершается, когда останется всего один кластер. Если разбиение идёт сверху вниз, то изначальный набор документов представлен одним общим кластером, который последовательно разбивается на подкластеры так, чтобы расстояние между ними было наибольшим возможным, пока не будет достигнуто их требуемое количество. Под расстоянием здесь понимается расстояние между векторами в пространстве векторов-документов, о которых шла речь в описании метода опорных векторов.

#### 5.2.2 Метод k средних (k means)

Случайным образом выбираются k точек в пространстве векторов-документов. Эти точки представляют собой центры масс кластеров. Все документы разбиваются на k кластеров, в соответствии с тем, к какому центру масс они ближе расположены, после чего центры масс пересчитываются. Документы перераспределяются по расстоянию до новых центров масс. Этот процесс происходит до тех пор, пока центры масс каждого кластера не станут стабильным, то есть не перестанут меняться после перерасчёта.

#### 5.2.3 Сечение пополам методом k средних (Bi-section k means)

Изначально все документы представлены одним большим кластером. Метод последовательно разбивает самый большой из имеющихся кластеров пополам, используя метод k средних, пока не будет получено необходимое количество кластеров. Иногда вместо самого большого кластера выбирается кластер с наибольшей дисперсией.

#### 5.2.4 Метод самоорганизующихся карт (Self-organizing maps)

 Метод использует специальную структуру, называющуюся нейронной сетью. Такая сеть состоит из двух слоёв:

 Первый слой – входной, представляет собой набор нейронов, каждый из которых соответствует слову в векторе-документе, поступающем на обработку.

 Второй слой – выходной, представляет собой плоскую карту, состоящую из такого количества нейронов, сколько кластеров нужно получить в итоге. Каждому из нейронов на карте связан со всеми нейронами входного слоя.

На первоначальном этапе каждой связи случайным образом проставляется небольшое числовое значение – вес связи. В процессе обучения на вход сети последовательно поступают вектора-документы. Когда вектор-документ поступает на обработку, нейроны из выходного слоя рассчитывают показатель схожести, основываясь на значениях самого вектора-документа и весов сработавших для выходного нейрона связей. Нейрон с самым высоким показателем схожести (то есть самый близкий к входному вектору-документу) вместе со своими соседями (определяемыми заранее заданной функцией) адаптируется, то есть весовые значения каждого адаптирующегося нейрона меняются так, чтобы они стали ближе к поступившему на вход вектору документу. Обученную сеть можно использовать для соотнесения произвольного вектора-документа с одним из нейронов выходного слоя, то есть с одним из кластеров.

#### 5.2.5 Метод максимального правдоподобия (EM-algorithm)

 Суть метода в том, что любой документ можно с определённой вероятностью отнести к тому или иному кластеру. Метод основывается на статистическом предположении, что каждый документ был создан следующим образом: с определённой вероятностью был выбран один из набора кластеров, затем на основе распределения вероятностей появления слов в документе из этого кластера генерируются слова документа. Метод пытается определить, какой именно кластер был выбран для создания документа.

Описания методов большей частью взяты из [1. Стр. 10-21]

## 6. Общие принципы

 Анализируя статистические методы анализа, можно заметить, что в большинстве своём все они на начальном этапе опираются на одни и те же данные – некие стандартные количественные характеристики документа или набора документов.

Быстрота выполнения запросов, возвращающих эти данные, и является главным показателем качества решения задачи проектирования и разработки исследовательского хранилища для статистического анализа текстов. Все количественные характеристики можно разбить на две основных группы:

1. Частотные показатели
2. Временные ряды

###  6.1 Частотные показатели

1. Количество всех слов в документе или наборе документов
2. Количество определённых слов в документе или наборе документов
3. Частотность слова в документе

- отношение количества вхождений слова к количеству всех слов

1. TF-IDF – статистическая мера, используемая для оценки важности слова в контексте документа, рассчитываемая по формуле:

TF-IDF(t, d)= tf(t, d) \*log2(m/ df(t)), где

m – количество документов в наборе,

tf(t , d) - сколько раз слово t встречается в документе d,

df(t) - сколько раз слово t встречается во всех документах набора

1. Иные взвешенные частотные показатели.
2. Векторное представление документа:

вектор td = (tf(t1, d), . . . , tf(tm, d)), где

d – один из документов,

tf(t, d) – сколько раз слово t встречается в документе d (возможен и другой частотный показатель).

1. Матрица слов-документов:

Столбцы представляют собой полный словарь набора документов (то есть набор уникальных слов из всех документов), строки – сами документы. 1 на пересечении строки и столбца означает, что документ содержит слово, 0 – не содержит. Такие матрицы – один из самых распространённых способов представления текста для дальнейшей машинной обработки [2, стр. 5]. Зачастую, вместо 1 и 0 на пересечениях строк и столбцов используют другие показатели, например, количество вхождений слова в документ или значения одной из, так называемых, «функций взвешивания». [4]

### 6.2 Временные ряды

 Временной ряд - это статистический показатель, описывающий исследуемый процесс, рассчитанный в разные моменты времени или аггрегированный за равные временные интервалы. Например, количество вхождений определённого слова в набор документов, рассчитанный для каждого месяца с 2005 до 2010 года. Получение и обработка таких данных – классический приём не только статистического анализа, но и бизнес-аналитики в целом.

## 7. Исследовательское хранилище данных

 Поддержка статистического анализа является одной из важнейших функций хранилища данных [6, стр. 141]. Лишь небольшая часть всех пользователей хранилища занимаются статистическим анализом, но результаты их работы зачастую приносят наибольшую пользу при принятии важных стратегических решений, так как задачи статистического анализа включают не только количественное описание некоторого набора данных, но и выявление некоторых зависимостей, позволяющих делать предположения о данных, которые поступят в будущем.

 Существует ряд важных отличий между статистическим анализом и другими формами взаимодействия с данными, и эти отличия оказывают непосредственное влияние на проектирование хранилища. Во-первых, для проведения статистического анализа обычно требуется одновременный доступ к сравнительно большому объёму данных, возможно, собранных за несколько лет или даже десятилетий. Во-вторых, запросы аналитического приложения к данным, разительно отличаются по сложности, необходимым для их выполнения ресурсам и вычислительной трудоёмкости от всех остальных запросов, поступающих к хранилищу (см. сравнение на рис. 1). Сами данные должны быть достаточно детализированы и структурированы особым, наиболее пригодным для проведения анализа, образом. К тому же, при решении задач статистического анализа, помимо стандартных, регламентированных подходов к их решению, прибегают и к эвристическим

методам исследования.



 *Рис .1 - Сравнение обычных и статистических запросов*



 *Рис .2- Применение эвристического метода исследований*

 Эвристическое исследование – итеративный процесс (см. рис. 2), цель которого изначально неизвестна. К данным последовательно применяются разные методы анализа, исследователь изучает результаты, которые, возможно, будут использованы на следующих этапах анализа, пытаясь таким образом прийти к какому-то заключению, получить некие полезные знания. Необходимость проведения эвристического анализа также влечёт за собой возникновение нескольких требований, а именно:

* возможность эффективного применения различных методов анализа;
* возможность проведения анализа на одном и том же, постоянном наборе данных, чтобы точно знать, что на результаты анализа повлияла смена метода, а не изменения в данных;

 В силу специфики статистического анализа оказывается удобным создавание отдельного хранилища, предназначенного только для исследовательских целей, связанного с основным. Такое хранилище физически отделено от основного. Набор данных в исследовательском хранилище (некоторая часть данных из основного в совокупности с данными из внешних источников) может быть зафиксирован на определённое время. Прошедшие предварительную обработку данные из основного хранилища попадают в исследовательское уже в пригодном для проведения анализа виде (обычно, самая высокая степень детализации). Сложные аналитические запросы не нагружают основное хранилище, а исследователь получает возможность работать с ограниченным и контролируемым набором именно тех данных, которые нужны ему. На рис. 3 показано взаиморасположение основного и исследовательского хранилищ и поток информации между ними. [6, стр. 147]



*Рис.3 – Поток информации между основным и исследовательским хранилищами*

## 8. Процесс анализа текстов

 Процесс анализа текста представляет собой следующую последовательность действий:

* 1. Получение набора данных для дальнейшей обработки и анализа
	2. Предварительная обработка текста
	3. Сохранение преобразованных документов в хранилище
	4. Применение аналитических методов
	5. Интерпретация результатов

### 8.1 Получение набора данных

На этом шаге исследователь определяет тот набор данных, который подвергнется анализу. В нашем случае такими данными являются текстовые документы, содержащиеся в основном хранилище в своём изначальном, неструктурированном виде или сторонние документы из внешней среды. Для корректной работы с неструктурированными документами необходимо знать их формат: обычно, в хранилище неструктурированные данные хранятся в особом формате двоичного представления данных (например, в BLOB-полях); не зная их оригинальный формат, текст невозможно будет восстановить. Кроме того, должен быть указан язык (или языки), на которых написан текст. Все остальные аттрибуты документов (например, автор, дата поступления в хранилище и т.п.) для проведения анализа важны, но не критичны.

### 8.2 Предварительная обработка текста

 После получения исходного набора документов, каждый из них должен быть преобразован таким образом, чтобы его можно было хранить в чётко структурированной среде, которая необходима для проведения статистического анализа. Этот шаг сам по себе представляет собой довольно сложный процесс, основанный на методах обработки естественного языка, и состоит из следующих стандартных процедур [1, стр. 6-7, 6, стр. 301]:

1. Подготовка текста
2. Приведение всего текста к одному регистру;
3. Удаление пунктуации, шрифтов и т.п.;
4. Проверка орфографии и исправление ошибок по необходимости;
5. Удаление шумовых слов

- то есть, слов, не представляющих ценности для статистического анализа, Примером таких слов в английском языке могу служить: a, and, the, was, that, which, to, from...;

1. Замена синонимов общими понятиями;
2. Томографический анализ

- многозначные слова заменяются общими понятиями, соответствующими каждому из возможных значений;

1. Замена слов на их основы

- однокоренные слова или различные формы одного и того же слова приводятся к общему виду, которым обычно служит их чистая основа;

1. Приведение альтернативных написаний (имена, аббревиатуры и т.п.) к общему виду;

 Зачастую, в этот список включают классификацию или кластеризацию, но в виду сложности этих задач, их эффективнее решать на более позднем этапе, когда текст уже структурирован. В конце концов, текст, пройдя через все эти преобразования, превращается в упорядоченный набор отдельных слов и их аттрибутов (например, более общее понятие для профессионального термина, ссылка на все возможные значения слова, часть речи...), который уже может быть сохранён в структурированной среде, такой, например, как таблицы реляционной базы данных.

### 8.3 Сохранение преобразованных документов в хранилище

 Полученный набор структурированных текстовых данных – конечное представление подлежащих анализу документов – распределяется по структуре хранилища в соответствии с заранее определённым методом. Этот шаг, в совокупности с двумя предыдущими, составляет классический ETL-процесс применительно к текстовым данным.

### 8.4 Применение аналитических методов

После того, как документы были перенесены в исследовательское хранилище данных в пригодной для анализа форме, специалисты (с помощью специальных аналитических приложений или напрямую) обращаются к хранилищу и на основе полученных данных, применяя статистические методы, проводят исследование.

### 8.5 Интерпретация результатов

 Полученные в ходе исследований результаты интерпретируются и в дальнейшем могут служить как основой для следующего этапа исследований, так и для принятия каких-либо решений (например, принять или отклонить изначальную гипотезу исследования). Иногда, результаты сами по себе представляют собой ценные данные, которые в дальнейшем записываются в исследовательское или даже в основное хранилище.

Схема всего процесса может быть представлена следующим рисунком [7, стр. 195]:

*Рис. 4*

# Глава II – Проектирование и разработка хранилища

## 1. Концептуальная модель

##

 В случае хранилища текстов для их последующего анализа, текстовый документ и слова в тексте документов – единственные два объекта моделируемой области. Каждый документ состоит из отдельных слов. У каждого слова есть набор харктеристик, например: часть речи. Статистические методы анализа работают только с чётко структурированными данными, следовательно, необходимо подобрать такую модель, в которой каждый документ будет представлен некоторой строго определённой структурой. Большинство наиболее популярных методов опираются на наличие или отсутствие слов в документах или количество вхождений того или иного слова в документ, не учитывая внутреннюю грамматическую или смысловую связь между словами отдельного текста. [8 стр.4] Поэтому, наиболее универсальной моделью для решения практических задач анализа текстов будет так называемая «**bag-of-words**»-модель: каждый документ представлен вектором <f1, f2, …, fn>, где fi – частота вхождения слова ti в документ. Набор документов будет, соответственно, представлен набором таких векторов, отсортированных по i, то есть матрицей слов-документов. Важно сказать, что иные частотные показатели, о которых говорилось в предыдущей части работы, могут быть легко расссчитаны на основе такой матрицы уже силами аналитического приложения, получившего данные из хранилища. В модели также стоит учитывать то, что зачастую объектом анализа будет не набор документов и их свойств, а отдельное слово и колебания частоты его употребления во времени. К тому же необходима возможность в случае необходимости дополнить модель, добавив новые характеристики документам или словам. Исследовательское хранилище данных, в этом случае, буквально только хранит опорные данные для исследований, но внутри самого хранилища не происходит никакого анализа: хранилище лишь возвращает необходимые данные (вектора документов или временные ряды слов) по запросу аналитического приложения.

Итак, на основании проведённого исследования проблемы можно составить список требований к хранилищу данных для статистического анализа текстовых документов:

1. Исследовательское хранилище должно быть, как логически, так и физически отделено от основного;
2. Документы в хранилище представлены представлены набором отдельных слов, составляющих текст документа. Нет необходимости сохранять грамматическую и смысловую связь текста;
3. Должна быть предусмотрена возможность добавления к документам и словам дополнительных аттрибутов;
4. Критерием качества модели хранилища будет скорость выполнения запросов, возвращающих:
5. Матрицу слов-документов для всех документов в хранилище;
6. Временные ряды для всех слов общего словаря документов в хранилище;

Концептуальную модель такого хранилища можно представить на диаграмме следующим образом: *рис. 7*

## 2. Логические модели

 Первым очевидным решением организации хранилища данных по вышеуказанной концептуальной схеме является хранение двух таблиц с предрассчитанными значениями для всего корпуса исследуемых документов. В этом случае доступ к данным осуществлялся бы мгновенно. К сожалению, если бы понадобилось добавить в хранилище дополнительный набор документов, то все таблицы пришлось бы перерасчитывать заново (словарь мог пополниться, значения временных рядов поменяться). Из-за этого недостатка использования полностью предрасчитанных таблиц появляется необходимость создания альтернативных моделей представления текстовых данных.

 Следующим шагом было создание нескольких логических моделей хранилища, начиная от самой простой и улучшая каждую последующую. После создания логических моделей стало возможным на их основе построить физические, то есть схему базы данных для конкретной СУБД, и провести их сравнительный анализ. Всего было создано 4 модели.

Три из них основаны на реляционных таблицах (SQL), одна на документно-ориентированной модели (NoSQL).

### 2.1 Эксперимент

Эксперимент заключался в сравнительном анализе важных характеристик моделей, выявленных на этапе концептуального проектирования. Каждая из четырёх логических моделей была превращена в физическую, реализована и проверена. Сама проверка состояла из трёх последовательных задач, время выполнения которых для каждой модели замерялось и записывалось:

1. Загрузка данных в хранилище;
2. Перевод всего корпуса документов в общую матрицу слов-документов;
3. Расчёт временного частотного ряда для каждого слова из словаря всего корпуса документов;

 В качестве опытных данных была взята довольно популярная коллекция текстовых документов Reuters-21758, обычно используемая для тестирования алгоритмов классификации текстов. Все тексты из неё были безо всяких изменений извлечены и записаны в простую таблицу в реляционной базе данных. Всего выбранный источник насчитывает 19042 документа, в которых, после удаления стоп-слов и стэмминга, находится 28306 различных слов. Даты для временных рядов были сгенерированны собственными силами – их около 14000. Таблица состоит из двух столбцов – идентификатор и сам текст документа в кодировке UTF-16. Эта база данных и послужила источником для всех остальных хранилищ. Вся база на диске заняла 252.13 MB. Список стоп-слов и символов, по которым проводилось разбиение текстов на отдельные слова, см. в Приложении.

 Код внешнего приложения, записывающего и преобразовывающего данные, по возможности не менялся от хранилища к хранилищу, чтобы изменения во времени выполнения были объяснимы только различиями в структуре хранилищ. В NoSQL варианте это условие сохранить не удалось: в силу ограниченных возможностей хранилища, некоторую часть расчётов пришлось переложить на внешнее приложение.

 Для реализации SQL-хранилищ использовалась технология Microsoft SQL Server 2008 R2, для NoSQL – MongoDb. Все внешние приложения были написаны на C#.

### 2.2 Реляционные таблицы

#### 2.2.1 Одна таблица

 В качестве первого, самого примитивного варианта, была выбрана схема базы данных, состоящая из всего одной таблицы. На рис.8 представлена диаграмма получившейся базы данных (в ходе эксперимента автор не делал разницы между логической и физической моделью, потому что, в силу простоты схем, логическая модель может быть быстро и однозначно переведена в физическую, и нет необходимости моделировать их в два шага).



 *рис. 8*

 Для получения матрицы слов-докментов и временных рядов были написаны SQL-запросы, для оптимизации быстродействия которых было сделано всё возможное, а именно – построены два индекса. Во время загрузки документов в хранилище индексы были отключены, из-за необходимости перестраивать их вставке каждого нового ряда, они существенно её замедляют. Далее приводятся запросы, после каждого из

них – индекс для оптимизации запроса.

**Запросы и соответствующие им индексы**

**Ряд из матрицы с.-д. для документа @i:**

SELECT COUNT(dt.doc\_id) AS occ

 FROM (

 SELECT DISTINCT dt.term

 FROM dbo.Doc\_Term dt

 ) AS x

 LEFT JOIN dbo.Doc\_Term dt

 ON dt.term = x.term

 AND dt.doc\_id = @i

 GROUP BY

 x.term

 ORDER BY

 x.term

**Индекс:**

CREATE NONCLUSTERED INDEX [doctermIDX] ON [dbo].[Doc\_Term]

(

 [doc\_id] ASC

)

INCLUDE ( [term])

**Временной ряд для слова @term:**

 SELECT x.doc\_date AS date,

 ISNULL(COUNT(dt.term), 0) AS occ

 FROM (

 SELECT DISTINCT(dt.doc\_date)

 FROM Doc\_Term dt

 ) AS x

 LEFT JOIN dbo.Doc\_Term dt

 ON x.doc\_date = dt.doc\_date

 AND dt.term = @term

 GROUP BY

 x.doc\_date

 ORDER BY

 x.doc\_date

**Индекс:**

CREATE NONCLUSTERED INDEX [termdateIDX] ON [dbo].[Doc\_Term]

(

 [term] ASC

)

INCLUDE ( [doc\_date])

**Результаты эксперимента**

 В таблице приведены результаты трёх экспериментов и место, занимаемое всей БД на диске:

|  |  |  |  |
| --- | --- | --- | --- |
| **Загрузка** | **Матрица слов-документов** | **Временные ряды** | **Размер** |
| 00:05:24.18 | 00:32:06.75 | 01:45:00.00 | 2085.44 MB |

 *Табл. 1*

 Для каждого слова, каждого документа в таблицу пишется новый ряд.

Из очевидных минусов можно отметить высокую степень дублирования информации: дублируются и идентификаторы документов, и их даты, и слова, встречающиеся в документе больше одного раза. Если понадобится добавить к документам или словам новые аттрибут, это станет очень трудоёмкой и длительной задачей, так как придётся проставлять значения для всех строк таблицы, что, кроме того, из-за дублирования информации потребует большого объёма памяти. Из плюсов – простота и высокая скорость записи.

 Главную проблему в данной схеме представляет необходимость JOIN'ить таблицу с самой собой по текстовому полю или дате, что является крайне затратной для выполнения операцией. Кроме того, размер базы стал в 8 раз больше исходной (с целыми текстами).

#### 2.2.2 Две таблицы

 В этом варианте все документы и даты из загрузки вынесены в отдельную таблицу, а слова связаны с ними посредством внешнего ключа. Диаграмма на рис. 9.



 *рис. 9*

**Запросы и соответствующие им индексы**

**Ряд из матрицы с.-д. для документа @i:**

 SELECT COUNT(t.doc\_id) AS occ

 FROM (

 SELECT DISTINCT t.term

 FROM dbo.Term t

 ) AS x

 LEFT JOIN dbo.Term t

 ON x.term = t.term

 AND t.doc\_id = @i

 GROUP BY

 x.term

 ORDER BY

 x.term

**Индексы:**

CREATE NONCLUSTERED INDEX [docidIDX] ON [dbo].[Term]

(

 [doc\_id] ASC

)

INCLUDE ( [term])

CREATE NONCLUSTERED INDEX [termIDX] ON [dbo].[Term]

(

 [term] ASC

)

**Временной ряд для слова @term:**

 SELECT d.date,

 isnull(COUNT(t.term\_id), 0) AS occ

 FROM dbo.[Document] d

 LEFT JOIN dbo.Term t

 ON d.doc\_id = t.doc\_id

 AND t.term = @term

 GROUP BY

 d.date

 ORDER BY

 d.date

**Результаты эксперимента**

|  |  |  |  |
| --- | --- | --- | --- |
| **Загрузка** | **Матрица слов-документов** | **Временные ряды** | **Размер** |
| 00:06:13.36 | 00:32:10.60 | 00:09:55.01 | 268.44 MB |

 *Табл. 2*

 Видно, что время загрузки стали чуть больше из-за необходимости поддерживать связь по внешнему ключу, и время построения матрицы слов-документов осталось прежним, что, впрочем, было ожидаемым – осталась необходимость self-JOIN'а по значениям слов. Зато временные ряды получили колоссальный прирост в скорости выполнения, а размер базы существенно сократился. Добавление нового аттрибута к словам всё ещё остаётся трудоёмкой задачей, так как в модели отсутствует отдельная таблица, представляющая слова из общего словаря документов.

#### 2.2.3 Три таблицы с предрасчётом

 Сделаны ещё пара шагов к оптимизации: все слова вынесены в отдельную таблицу - словарь, связь между документами и словами идёт через вспомогательную таблицу, в которой также хранятся предрассчитанные на этапе загрузки количества вхождений слов в документы. Диграмма на рис. 10.



*рис.10*

**Запросы и соответствующие им индексы**

**Ряд из матрицы с.-д. для документа @i:**

 SELECT ISNULL(dt.occs, 0) AS occ

 FROM dbo.Term t

 LEFT JOIN dbo.Doc\_Term dt

 ON dt.doc\_id = @i

 AND dt.term\_id = t.id

 ORDER BY

 t.term

**Индекс:**

CREATE NONCLUSTERED INDEX [dtIDX] ON [dbo].[Doc\_Term]

(

 [doc\_id] ASC

)

INCLUDE ( [term\_id], [occs])

**Временной ряд для слова @termId:**

 SELECT d.date,

 ISNULL(SUM(dt.occs), 0) AS occ

 FROM dbo.[Document] d

 LEFT JOIN dbo.Doc\_Term dt

 ON d.id = dt.doc\_id

 AND dt.term\_id = @termId

 GROUP BY

 d.date

 ORDER BY

 d.date

**Индекс:**

CREATE NONCLUSTERED INDEX [TSidx] ON [dbo].[Doc\_Term]

(

 [term\_id] ASC

)

INCLUDE ( [doc\_id],[occs])

**Результаты эксперимента:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Загрузка** | **Матрица слов-документов** | **Временные ряды** | **Размер** |
| 00:07:20.68 | 00:16:37.94 | 00:09:33.12 | 1303.13 MB |

 *Табл. 3*

 Время загрузки опять незначительно выросло (теперь нужно было поддерживать два внешних ключа и вести словарь), размер базы в 5 раз больше исходной. Построение матрицы с.-д. идёт в два раза быстрее, чем в предыдущем варианте, временные ряды чуть-чуть ускорились. Важно то, что в этом варианте, в отличие от предыдущих двух, удалось обойтись без вложенных запросов – источника больших проблем с производительностью.

### 2.3 Документно-ориентированный подход

 В MongoDb понятие схемы или структуры данных отсутствует, как таковое. Каждый документ в коллекции документов может иметь произвольный вид (произвольное количество полей, произвольное же их содержание). Изначально планировалось представить каждый документ в следующем виде:

{

 \_id : 7A322E7B-FAEC-45F3-8AC4-450E9AC208AB

 date : 09.09.2009 15:43:34.12

 words :

 [

 {word:”word1”, occ:1}

 {word:”word2”, occ:2}

 …

 {word:”wordN”, occ:5}

 ]

}

, то есть хранить каждое слово отдельным поддокументом в массиве исходного документа с указанием количества вхождений этого слова в текст документа, но, к сожалению, механизм работы с вложенными документами в этой СУБД ещё не до конца разработан и не позволяет решить поставленные перед нами задачи без помощи сторонних приложений. В итоге, все документы стали иметь вид:

{

 \_id : 7A322E7B-FAEC-45F3-8AC4-450E9AC208AB

 date Field: 09.09.2009 15:43:34.12

 word1 : 1

 word2 : 2

 …

 wordN:N

}

 На каждое слово, встречаемое в документе, заводится отдельное поле, значением которого становится количество вхождений слова в документ.

Следует оговориться, что в силу особенностей СУБД всё-таки пришлось прибегнуть к обработке данных сторонним приложением, тогда как все предыдущие варианты возвращали уже готовые для проведения анализа структуры данных.

Для получения словаря всего корпуса текстов использовался, за неимением другой возможности, механизм map-reduce:

mr = db.runCommand({

 "mapreduce" : "docs",

 "map" : function() {

 for (var key in this) { emit(key, null); }

 },

 "reduce" : function(key, stuff) { return null; },

 "out": "docs" + "\_keys"

})

Для получения вектора для каждого документа использовался механизм LINQ платформы .Net Framework:

var s = docs.FindAllAs<BsonDocument>().OrderBy(p => p.GetValue("n"));

            foreach (var doc in s)

            {

                var r = (from p in dict

                         join c in doc.Elements

                         on p.GetValue("\_id") equals c.Name into j1

                         from j in j1.DefaultIfEmpty(

 BsonElement.Create(

 p.GetValue("\_id").AsString,

 BsonValue.Create(0))

 )

                         select new

                         {

                             term = j.Name,

                             occ = j.Value

                         }).ToList().OrderBy(p => p.term);

            }

Для временного ряда использовался стандартный механизм аггрегации MongoDb:

db.docs.aggregate( [

 { $match: { one : { $exists : true } },

 { $group: { \_id: "$dateField",

 total: { $sum: "$term" } } }

] )

(Для слова term)

**Результаты эксперимента:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Загрузка** | **Матрица слов-документов** | **Временные ряды** | **Размер** |
| 00:00:17.67 | 00:08:46.54 | ~02:38:00.00 | 256 MB |

 *Табл. 4*

 Моментальная загрузка, время построения матрицы с.-д. в два раза меньше наилучшего реляционного варианта, размер самый маленький из всех предыдущих, но получение временных рядов идёт дольше, даже чем в самой примитивной табличной схеме. Добавление новых аттрибутов к словам документа в данной схеме остаётся задачей, у которой не существует тривиального решения, хотя в изначальной схеме она решается просто добавлением нового поля в каждый вложенный документ-слово. Главный минус этой схемы, помимо невероятно медленного времени выполнения запроса по временным рядам, состоит в том, что в ней нет отдельной коллекции, представляющей полный словарь всего корпуса исследуемых документов.

## 3. Сведение результатов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Модель** | **Загрузка** | **Матрица слов-документов** | **Временные ряды** | **Размер** |
| 1 таблица | 00:05:24.18 | 00:32:06.75 | ~01:45:00.00 | 2085.44 MB |
| 2 таблицы | 00:06:13.36 | 00:32:10.60 | 00:09:55.01 | 268.44 MB |
| предрасчёт | 00:07:20.68 | 00:16:37.94 | 00:09:33.12 | 1303.13 MB |
| Mongo | 00:00:17.67 | 00:08:46.54 | ~02:38:00.00 | 256 MB |

 *Табл. 5*

 Из испробованных четырёх моделей, наибольшего внимания заслуживает модель с предрасчётом, так как она сравнительно быстро справляется со обеими задачами и легко дорабатываема (например, если понадобится включить в модель новый аттрибут для слов, это легко будет сделать, добавив поле в таблицу со словарём и заполнив его для каждого слова всего один раз). Стоит отметить, что такая схема – это канонический вариант представления табличных данных в реляционных базах данных, а в основе концептуальной модели хранилища лежит именно таблица.

## 4. Заключение

 В ходе данной работы было проведено исследование процесса статичтического анализа текстовых данных, особенностей хранения данных для таких исследований, ключевые аспекты методов решения аналитических задач. На основе результатов этого исследования был составлен список требований к хранилищу данных для статистического анализа текстов, который лёг в основу проектирования и разработки опытных образцов. Была создана единая концептуальная модель хранилища и 4 логических модели, реализующих эту концепцию. После создания логических моделей стало возможным построить реальные хранилища данных, используя две различных технологии хранения и обработки информации: реляционные таблицы и хранилище документов с произвольной структурой. Эксперименты, проведённые на полученных хранилищах, включали три этапа: предварительную обработку и загрузку данных, получение временных рядов по всем словам из общего словаря документов и перевод всех документов хранилища в форму матрицы слов-документов. Анализ результатов позволил выявить наиболее подходящую для решения поставленной задачи модель и выявить недостатки всех остальных моделей. Главная ценность работы состоит в простой, осмысленной и оправданной концептуальной модели, по которой можно видеть, какие задачи должно решать исследовательское хранилище, в какой форме аналитические приложения будут запрашивать данные и какие возможные расширения могут понадобиться аналитику.

# Список источников

[1.] Andreas Hotho, Andreas Nürnberger, Gerhard Paaß, *A Brief Survey of Text Mining*, May 13, 2005

[2.] Feinerer, I., K. Hornik, and D. Meyer, “Text mining infrastructure in R,” Journal of Statistical Software 25:5, Mar 2008, <http://www.jstatsoft.org/v25/i05>.

[3.] Louise Francis, Matt Flynn, *Text Mining Handbook*, Casualty Actuarial Society E-Forum, Spring 2010

[4.] Gerard Salton, Christopher Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing & Management Vol. 24, No. 5, pp. 513-523, 1988

[5.] Anna Huang, *Similarity Measures for Text Document Clustering*, NZCSRSC 2008, April 2008, Christchurch, New Zealand

[6.] Bill H. Inmon, *Data Warehousing 2.0 Modeling and Metadata Strategies for Next Generation Architectures*, Forest Rim Technology, LLC, April 2010

[7.] Барсегян А. А., Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2007

[8.] Sholom M. Weiss, Nitin Indurkhya, Tong Zhang, Fred J. Damerau, *Text Mining: Predictive Methods for Analyzing Unstructured Information*

[9.] Mark Sanderson, *Reuters test collection,* Glasgow University Computing Science Department, Saturday, 11 June, 1994

[10.] Bill H. Inmon, *Data Warehousing 2.0 Modeling and Metadata Strategies for Next Generation Architectures*, April 2010

[11.] Bodo H¨usemann, Jens Lechtenb¨orger, Gottfried Vossen,*Conceptual DataWarehouse Design*

[12.] Carsten Felden, Peter Chamoni, *Recommender Systems based on an Active Data Warehouse with Text Documents*

[13.] Jaideep Srivastava, Hung Q. Ngo, *Statistical Databases*

[14.] Ted Dunning, *Accurate methods for the statistics of surprise and coincidence*, Computational Linguistics, Volume 19, Number 1, 1993

[15.] ChengXiang Zhai, *Beyond Search: Statistical Topic Models for Text Analysis*, Keynote at SIGIR, 2011, July 26, 2011

[16.] M.J. Mart´ın-Bautista, C. Molina, E. Tejeda, and M. Amparo Vila, *Using Textual Dimensions in Data Warehousing Processes*, 2010

[17.] Christan Grant, Joir-dan Gumbs, Kun Li, Daisy Zhe Wang, George Chitouras, *MADden: Query-Driven Statistical Text Analytics*, October 29–November 2, 2012

[18.] Christopher D. Manning, Hinrich Schütze, Foundations of Statistical Natural Language Processing, The MIT Press, Second printing, 1999

[19.] [http://docs.mongodb.org](http://docs.mongodb.org/)

[20.] <http://stackoverflow.com/>

[21.] [http://msdn.microsoft.com](http://msdn.microsoft.com/)

[22.] <http://en.wikipedia.org/wiki/Bag-of-words_model>

[23.] <http://en.wikipedia.org/wiki/Document-term_matrix>

# Приложение

## Список стоп-слов:

a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at, be, because, been, before, being, below, between, both, but, by, can't, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't, down, during, each, few, for, from, further, had, hadn't, has, hasn't, have, haven't, having, he, he'd, he'll, he's, her, here, here's, hers, herself, him, himself, his, how, how's, i, i'd, i'll, i'm, i've, if, in, into, is, isn't, it, it's, its, itself, let's, me, more, most, mustn't, my, myself, no, nor, not, of, off, on, once, only, or, other, ought, our, ours , ourselves, out, over, own, same, shan't, she, she'd, she'll, she's, should, shouldn't, so, some, such, than, that, that's, the, their, theirs, them, themselves, then, there, there's, these, they, they'd, they'll, they're, they've, this, those, through, to, too, under, until, up, very, was, wasn't, we, we'd, we'll, we're, we've, were, weren't, what, what's, when, when's, where, where's, which, while, who, who's, whom, why, why's, with, won't, would, wouldn't, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves;

## Список символов разбивки:

' ', ',', '.', ',', '-', '\_', '=', '+', '!', '\'', ';', ':', '%', ']', '[', '?', '\*', '(', ')', '/', '\\', '#', '\t', '\n', '"', '>', '<', '$', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0';

В процессе разбора текстов использовался стэммер компании Snowball, адаптированный для С# компанией Iveonik (<http://www.iveonik.com/blog/2011/08/stemmery-snowball-na-csharp-free-download/>).