**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н. Э. БАУМАНА**

**Факультет информатики и систем управления**

**Кафедра теоретической информатики и компьютерных технологий**

Наброски дипломного проекта

# «Автоматическое установление связей между сообщениями твиттера и новостными статьями»

Выполнил:
студент ИУ9-101
Выборнов А. И.

Руководитель:
Лукашевич Н.В.

Москва 2016

# Аннотация

В данной работе рассматривается применение методов машинного обучения и обработки естественного языка к решению задачи автоматической связи сообщений твиттера и новостных статей.

Проведен обзор существующих подходов к решению задачи. Проведено исследование эффективности различных методов решения задачи и разных способов построения набора данных. Реализовано автоматическое установления связей между сообщениями твиттера и новостными статьями в формате рекомендаций для твита наиболее подходящих новостных статей.

В организационно-экономической части представлено технико-экономическое обоснование разработки, проанализирована структура затрат проекта и построен календарный план-график проекта.

Пояснительная записка к работе содержит текст на ... листах формата А4, ... таблиц, ... блок-схем и ... рисунков, список литературы из ... библиографических ссылок. К данной работе разработана презентация в формате *.pdf из ... слайдов.

# Содержание

# Введение

В современном мире всё больший вес приобретают социальные медиа (преимущественно социальные сети). Их главное отличие от традиционных медиа (газеты, тв) заключается в том, что контент порождается тысячами и миллионами людей. Социальные медиа не заменяют традиционные новостные источники, а дополняют их. Они могут служить полезным социальным датчиком того, насколько популярна история (тема) и насколько долго. Часто, обсуждения в социальных медиа основаны на событиях из новостей и, наоборот, социальные медиа влияют на новостные события.

Одной из самых популярных социальных сетей является Twitter (Твиттер) — социальная сеть для публичного обмена сообщениями. Главной особенностью Твиттера является малый размер сообщений (140 символов), называемых твитами. Часто твиты являют собой описание происходящего прямо сейчас события, отклик на него.

Происходящие в мире события описываются статьями в новостных изданиях. Новостные статьи и твиты пользователей твиттера не редко описывают одно и то же событие. Существует актуальная проблема установления связей между твитами и новостными статьями, которые описывают одно и тоже событие. Выявление связи между сообщениями твиттера и новостями позволит как расширить информативность твитов, так и обогатить новости.

Среди преимуществ расширения новости с помощью твитов можно выделить такие, как определение отношения аудитории к новости, дополнительные признаки для тематической классификации новостей, дополнительная информация для аннотирования новостей.

Современные методы обработки естественного языка хорошо работают, используя большой массив текста в качестве входных данных, однако, они становятся неэффективными, когда применяются на коротких текстах, таких как твиты. Существенным преимуществом расширения твита с помощью новости является появляющаяся возможность использования большого количества методов обработки естественного языка.

Данная работа ставит целью исследование и разработку методов автоматического установления связей между сообщениями твиттера и новостными статьями.

# 1. Обзор

Решение задачи по установлению взаимосвязи между твитами и новостными статьями в общем случае представляет собой решение задачи определения семантического сходства между короткими текстами. Методы естественной обработки языка не позволяет с высокой степенью точности определить семантическое сходство между короткими текстами, поэтому установление связей между твитами и новостями должно опираться на дополнительную информацию о предметной области.

## 1.1. Терминология

Решение задачи предполагает использование наработок различных дисциплин, таких как: обработка естественного языка, машинное обучение, информационный поиск, а основным источником данных выступают интернет ресурсы. Поэтому в работе используется специфичная терминология.

*Твиттер* — социальная сеть для публичного обмена короткими (до 140 символов) сообщениями при помощи веб-интерфейса, SMS, средств мгновенного обмена сообщениями или сторонних программ-клиентов.

*Твит* — термин сервиса микроблоггинга Твиттер, обозначающий сообщение, публикуемое пользователем в его твиттере. Особенностью твита является его длина, которая не может быть больше 140 знаков.

*Ретвит* — сообщение, целиком состоящее из цитирования сообщения одного пользователя Твиттера другим.

*Новость* — оперативное информационное сообщение, которое представляет политический, социальный или экономический интерес для аудитории в своей свежести, то есть сообщение о событиях произошедших недавно или происходящих в данный момент.

*Хэштег* — слово или фраза, которым предшествует символ #, используется в различных социальных сетях (Twitter, Facebook, Instagram) для объединения группы сообщений по теме или типу. Например: #искусство, #техника, #смешное, #анекдоты и т.д.

*URL* (от англ. Uniform Resource Locator — единый указатель ресурса) — единообразный определитель местонахождения ресурса. URL служит стандартизированным способом записи адреса ресурса в сети Интернет.

*Обработка естественного языка* (англ. Natural language processing) — направление математической лингвистики, которое изучает проблемы компьютерного анализа и синтеза естественных языков.

*Именованная сущность* — последовательность слов, являющаяся именем, названием, идентификатором, временным, денежным или процентным выражением.

*Аннотирование текста* — краткое представление содержания текста в виде аннотации (обзорного реферата).

*Информационный поиск* — процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности, и наука об этом поиске.

*TF-IDF* (от англ. TF — term frequency, IDF — inverse document frequency) — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален количеству употребления этого слова в документе, и обратно пропорционален частоте употребления слова в других документах коллекции.

*TF-IDF матрица* — матрица, строки которой соответствуют словам из корпуса, а столбцы текстам. Значение ячейки матрицы $(i, j)$ равно значению метрики tf-idf для слова, соответствующего строчке $i$, и текста, соответствующего столбцу $j$.

*WTMF* — метод машинного обучения, применяемый для анализа схожести между короткими текстами [4].

*Тематическое моделирование* — способ построения модели коллекции текстовых документов, которая определяет, к каким темам относится каждый из документов.

*LDA* (от англ. Latent Dirichlet allocation — Латентное размещение Дирихле) — методов тематического моделирования, позволяющий объяснять результаты наблюдений с помощью неявных групп.

## 1.2. Существующие подходы к решению задачи

Задача автоматического установления связей между твитами и новостными статьями до сих пор не имеет устоявшегося решения. В рамках предварительного исследования были отобраны наиболее перспективные подходы к решению задачи, а именно:

- метод WTMT-G, представляющий собой доработку метода WTMF, которая позволила учитывать информацию о связях между текстами;
- обобщённый метод, позволяющий по новости находить относящиеся к ней высказывания из социальных медиа;
- связывание твитов с новостями на основе словарей соответствий;

Также рассматривается классическое решение задачи определения схожести текстов на основе частотности употребления слов. Ниже представлен краткий обзор выбранных методов.

Стоит также ввести несколько определений употребляемых в дальнейшем: под связью *текст-текст* подразумевается определение двух текстов как схожих на основе некоторой дополнительной информации; под связью *текст-слово* подразумевается определение двух текстов как схожих только на основе слов, из которых состоит текст.

### 1.2.1. Определение схожести текстов на основе частотности употребления слов

Наиболее простым и очевидным подходом к решению задачи связывания твитов с новостными статьями, является связывание текстов, наиболее близких по частотности употребления слов. Способ основан на сравнении значений метрики TF-IDF, поэтому в дальнейшем будем называть этот спосок TF-IDF методом.

Решение задачи связывания твитов с новостными статьями на основе частотности употребления слов можно представить в виде небольшого алгоритма:

1. объединить тексты всех твитов и тексты всех новостей — (для новости текст это конкатенация заголовка и краткого изложения);
2. в качестве корпуса использовать объединение начальных форм всех слов, используемых в текстах, за вычетом списка стоп-слов (под списком стоп-слов подразумевается набор часто употребимых слов языка, которые вне контекста не несут смысловой нагрузки, к примеру, предлоги);
3. по множеству текстов и построенному корпусу построить TF-IDF матрицу;
4. каждому тексту сопоставить столбец TF-IDF матрицы, соответствующий тексту (вектор для сравнения);
5. рассматривая вектор для сравнения в качестве координат в метрическом пространстве, для каждого твита найти список наиболее похожих на него новостей.

В работе в качестве меры близости в метрическом пространстве используется косинусная мера близости — мера численно равная косинусу угла между векторами. В дальнейшем каждый раз, когда говориться о схожести или близости двух векторов, подразумевается близость согласно косинусной мере.

### 1.2.2. Обобщённый метод, сопоставляющий новостной статье высказывания из социальных медиа

В рамках метода решается следующая задача: по новости необходимо найти высказывания в социальных сетях, которые на неё неявно ссылаются. Метод был предложен в статье [2]. Поставленная задача решается в три этапа:

1. по заданной новостной статье формируется несколько моделей запросов, которые создаются как на основе структуры статьи, так и на основе явно связанных со статьей высказываний из социальных медиа.
2. построенные модели используются для получения высказываний из индекса целевого социального медиа, результатом являются несколько ранжированных списков;

3. полученные списки объединяются с использованием особой техники слияния данных.

Авторы также предлагают способ, созданный для борьбы с дрейфом запроса (порождение менее подходящего запроса), который возникает при большом объёме используемого текста. Способ основан на выборе дополнительных отличительных условий.

Для экспериментальной оценки используются данные из различных медиа, таких как Twitter, Digg, Delicious, the New York Times Community, Wikipedia, а так же из блогов.

В результате работы показано, что модели запросов, основанные на различных источниках данных, повышают точность выявления высказываний из социальных медиа; методы слияния ранжированных списков приводят к значительному повышению производительности в сравнении с другими подходами.

### 1.2.3. Связывание твитов с новостями на основе словарей соответствий

Метод связывания твитов с новостными статья, основанный на словарях соответствий, предложен в статье [3]. *Словарь соответствий* — множество слов, которые встречаются только в твитах и, соответственно, не встречаются в новостях. Авторы предложили способ автоматического установление связи между множеством твитов и множеством новостей определённой темы. Темы извлекаются из новостей на основе методов тематического моделирования.

Значительную сложность при решении проблемы связывания твитов с новостями вызывают малый размер твита и различия в словарях: в твитах используются аббревиатуры, неформальный язык, сленг; в новостях, напротив, используется литературный язык. В частности, твиты могут вообще не нести смысловой нагрузки.

Твиттер предлагает хэштэги, как механизм для категоризации твитов. Но этот подход обладает рядом недостатков, таких как: не все записи содержат хэштеги, хэштег не содержит информацию о событии, хэштег сформулирован в слишком общей форме, твит содержит несколько хэштегов. Следовательно использование одних хэштегов приведёт к низкому качеству связывания твитов с новостями.

Для решения задачи и преодоления описанных выше проблем, авторами работы предлагается следующий подход:

1. С помощью метода LDA из множества новостей извлекается набор тем. Тема характеризуется распределением частот слов, характерных для этой темы.

2. Каждой полученной теме сопоставляется множество наиболее близких к ней твитов.

3. Из полученных твитов извлекаются слова, которые дополняют характеристику рассматриваемой темы.

4. Полученные слова образуют словарь соответствий и служат «мостом» к другим твитам.

В результате работы продемонстрирован способ установления связей между множеством твитов и множеством новостей с использованием словарей соответствий.

### 1.2.4. Метод WTMF

Метод WTMF предназначен для определения семантической близости коротких текстов [4]. Этот метод учитывает отсутствующие в тексте слова в виде признаков короткого текста. Под отсутствующими словами подразумеваются все слова из корпуса, составленного из всех текстов, за исключением слов из рассматриваемого короткого текста, то есть отсутствующие слова можно трактовать как негативный сигнал.

Работа метода WTMF основана на разложении TF-IDF матрицы $X$ в произведение двух матриц $P$ и $Q$:

$$X \sim P^T Q.$$

На рисунке 1 показано как матрица $X$ приближается произведением двух матриц $P^T$ размера $M \times K$ и $Q$ размера $K \times N$.

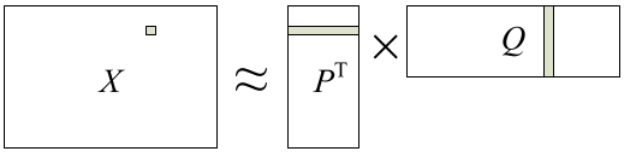


Рисунок 1 — Разложение TF-IDF матрицы ($X$) на произведение матриц $P$ и $Q$

Каждый текст $s_j$ представлен в виде вектора $Q_{\cdot,j}$ размерности $K$, каждое слово $w_i$ представлено в виде вектор $P_{i,\cdot}$. Если $X_{ij} = (P_{i,\cdot}, Q_{\cdot,j})$ близко к нулю, то это трактуется как отсутствующее слово.

Задачей метода является минимизация целевой функции ($\lambda$ - регуляризирующий член, матрица W определяет вес элементов матрицы X):

$$\sum_i \sum_j W_{ij}(P_{i,\cdot} \cdot Q_{\cdot,j} - X_{ij})^2 + \lambda||P||_2^2 + \lambda||Q||_2^2.$$

Для получения векторов $P_{i,\cdot}$ и $Q_{\cdot,j}$ используется алгоритм описанный в статье [6]. Сначала P и Q инициализируются случайными числами. Затем запускается итеративный пересчёт P и Q по следующим формулам (эффективный способ расчёта описан в [5]):

$$P_{i,\cdot} = (QW_i'Q^T + \lambda I)^{-1} QW_i' X_{i,\cdot}^T,$$

$$Q_{\cdot,j} = (PW_j'P^T + \lambda I)^{-1} PW_j' X_{j,\cdot}.$$

Здесь $W_i' = diag(W_{i,.})$ - диагональная матрица полученная из $i$-ой строчки матрицы $W$, аналогично $W_j' = diag(W_{.,j})$ - диагональная матрица полученная из $j$-ого столбца матрицы $W$. Матрица $W$ определяется следующим образом:

$$W_{ij} = \begin{cases} 1, & if\ X_{ij} \neq 0, \\ w_m, & otherwise. \end{cases},$$

где $w_m$ положительно и $w_m << 1$.

Столбцы построенной матрицы Q представляют собой вектора для сравнения текстов между собой. Тексту, на основе которого построена $i$-я строка TF-IDF матрицы $X$, в соответствие ставится $i$-й столбей матрицы Q.

В статье предложен подход для поиска семантической близости текстов, который на небольших текстах работает лучше чем подход, основанный на частотности слов.

### 1.2.5. Метод WTMF-G

Метод WTMG-G решает задачу установления связей между твитами и новостными статьями, путём построения модели, которая учитывает неявные связи между текстами, предложен в статье [1].

Метод WTMF-G (WTMF on Graphs) представляет собой доработанный метод WTMF, позволяющий хорошо моделировать семантику коротких текстов, но не учитывающий некоторые специфичные для твитов и новостей характеристики, которыми обладает исходная выборка и которые взаимосвязаны с семантической близостью текстов:

1. хештеги, которые являются прямым указанием на смысл твита;
2. именованные сущности, которые с высокой точностью можно извлекать из новостей;
3. информацию о времени публикации твитов и новостей.

Метод WTMF-G расширяет возможности метода WTMF, путём учёта взаимосвязи текстов на основе специфичных для твитов и новостных статей характеристик, то есть позволяет учесть информацию о взаимосвязи текст-текст.

Для решения задачи необходимо иметь эталонный набор данных, на котором будет производится оценка качества полученного решения. Сначала за общий период времени собираются твиты и новости. Для твита помимо текста хранится информация о времени публикации и авторе работы. Для новости хранится время публикации, заколовок, краткое изложение и URL.

На основе собранной информации строится набор данных, который состоит из трёх частей:

1. множество новостей — все собранные новости;
2. множество связей твит-новость, под связью подразумевается явное указание URL новости в тексте твита;
3. множество твитов – все твиты, имеющие связь с одной из собранных новостей.

К построенному набору данных применяется метод WTMF-G — то есть метод WTMF, расширенный путём добавления связей текст-текст. Добавление связей текст-текст происходит путём модификации регуляризирующего члена *lambda*. Для каждой пары связанных текстов $j_1$ и $j_2$:

$$\lambda = \delta \cdot (\frac{Q_{\cdot,j_1} \cdot Q_{\cdot,j_2}}{|Q_{\cdot,j_1}||Q_{\cdot,j_2}|} - 1)^2,$$

коэффициент $\delta$ задаёт степень влияния связей текст-текст.

Так как новый регуляризирующий член *lambda* зависит от $|Q_{\cdot,j}|$, который меняется во время итерации, вводим упрощение: длина вектора $Q_{\cdot,j}$ не изменяется во время итерации. Также необходимо модифицировать итеративный процесс построения матриц $P$ и $Q$ следующим образом:

$$P_{i,\cdot} = (QW_i'Q^T + \lambda I)^{-1}QW_i'X_{i,\cdot}^T,$$

$$Q_{\cdot,j} = (PW_j'P^T + \lambda I + \delta L_j^2 Q_{\cdot,n(j)} diag(L_{n(j)}^2) Q_{\cdot,n(j)}^T)^{-1}(PW_j'X_{j,\cdot} + \delta L_j Q_{\cdot,n(j)} L_{n(j)}).$$

В этих формулах $n(j)$ — список связанных текстов с текстом $j$. $Q_{\cdot,n(j)}$ — матрица, состоящая из связанных векторов для $Q_{\cdot,j}$. $L_j$ - длина вектора $Q_j$ на начало итерации, $L_n(j)$ — вектор длин векторов связанных с $j$, то есть $Q_{\cdot,n(j)}$, полученный на начало итерации.

В статье показано, что добавление информации о взаимосвязи текст-текст позволяет повысить качество установления связей между твитами и новостными статьями. Качество метода WTMF-G, измеренное с использованием метрики MRR (метрика описана в главе 6.1.1), в сравнении с такими популярными подходами как: TF-IDF, LDA, WTMF, показано в таблице 1.

Таблица 1: Значение метрики MRR для алгоритма WMTF-G в сравнении с другими подходами.

| **Алгоритм** | TF-IDF | LDA | WTMF | WTMF-G |
|---|---|---|---|---|
| **Значение MRR** | 0.4602 | 0.1313 | 0.4531 | 0.4791 |

В таблице 1 показано, что алгоритм WTMF-G даёт лучшее качество, чем прочие подходы.

## 1.3. Выбор подхода для решения задачи

В качестве основного подхода, на основе которого строится решение задачи по установлению связей между твитами и новостями, был выбран WTMF-G. Основной причиной подобного

выбора является то, что большинство подходов учитывают только статистические зависимости вида текст-слово; метод WTMF-G, напротив, не ограничивается зависимостями текст-слово, а позволяет учесть взаимосвязь текст-текст, что, как ожидается, даст прирост качества в решении задачи установления связей.

Также, в рамках работы задача по установления связей между твитами и новостями решена классическим подходом для установления связей между текстами — определение схожести текстов на основе частотности употребления слов. Этот подход даёт хорошие результаты на больших текстах. Результаты этого метода помогут оценить влияние связей вида текст-текст в методе WTMF-G на качество полученного решения.

# 2. Постановка задачи

Задачей данной работы является автоматическое установления связей между твитами и новостными статьями. Это включает в себя как исследование с целью нахождения налучшего метода для установления связей, так и написание программного обеспечения, которое позволит автоматизировано устанавливать связи между твитами и новостными статьями.

В разделе 1 проведено исследование существующих методов автоматического установления связей между сообщениями твиттера и новостными статьями, выбраны методы, на основе которых необходимо реализовать программный комплекс, позволяющий устанавливать связи между твитами и новостными статьями.

Для установления связей должен быть собран эталонный набор данных, которой состоит из множества твитов, новостей и связей между ними. Выбранный алгоритм WTMF-G накладывает ограничение на формат эталонного набора: для каждого твита существует связь с единственной новостью.

Решение задачи установления связей между твитами и новостными статьями в общем случае неоднозначно: как твиту может соответствовать несколько новостей, так и новостной статье может соотвествовать несколько твитов. Отталкиваясь от существующего ограничения: твит связан с единственной новость, получаем что для оценки качества установления связей хорошо подходят метрики, принятые в информационном поиске. Для использования подобных метрик будем рассматривать твит как запрос, в терминологии информационного поиска, а список новостей как ответ нашей системы установления связей. То есть для каждого твита мы получаем список новостей, ранжированный по мере убывания их схожести, в дальнейшем будем называть подобный список рекомендацией, а процесс установления связей построением рекомендаций.

Результатом работы является создание программного комплекса, который реализует такие методы машинного обучения как WTMF, WTMF-G, TF-IDF, позволяет для произвольного твита построить рекомендацию новостей с использованием любого из предложенных методов машинного обучения, а также способен оценить качество используемого метода машинного обучения.

# 3. Получение и разметка данных

Для работы с алгоритмами автоматического связывания твитов и новостей был построен набор данных, который состоит из четырёх частей:

1. множество новостей — все собранные новости;
2. множество связей твит-новость, под связью подразумевается пара твит-новость, где в твите говорится о описанной в статье новости;
3. множество твитов – все твиты, имеющие связь с одной из собранных новостей.
4. множество связей текст-текст.

Для построения набор данных необходимо: собрать твиты и новости за длительный промежуток времени; предобработать полученные данные; выявить (разметить) связи твит-новость; построить множество связей текст-текст.

## 3.1. Получение данных

Получение данных включает в себя не только скачивание информации (твиты и новости), но и выявление корректных источников данных. В рамках получения данных сделано:

1. реализовано скачивание твитов и новостей из разных новостных источников;
2. получены твиты за небольшой промежуток времени;
3. расшифрованы сокращённые URL;
4. определён список наиболее популярных новостных источников в твиттере;
5. в течение длительного времени собраны данные как с твиттера, так и с новостных источников.

### 3.1.1. Получение твитов

Для получения данных твиттера используется Twitter Streaming API — сервис, предоставляющий разработчикам возможность в реальном времени получить поток данных твиттера. С помощью Twitter Streaming API можно бесплатно получить 1% от всех публичной информации твиттера: публикация и удаления твитов.

Для работы с Twitter Streaming API на сайте https://apps.twitter.com/ зарегистрировано новое приложение и получен набор секретных ключей, которые требуются для авторизации. Для упрощения работы с Twitter Streaming API использована библиотека tweepy, предоставляющая удобный интерфейс на языке Python.

Twitter Streaming API предоставляет данные в формате JSON (от англ. JavaScript Object Notation) — текстовый формат обмена данными, удобный для чтения человек, первоначально создавался как формат на текстового описания и сериализации объектов языка программирования JavaScript.

В рамках работы использована информация о публикации твитов. Каждое событие о создании твита описывается в виде большого количества параметров (несколько десятков), в работе использованы следующие:

1. created_at — дата создания,
2. id — уникальный идентификатор,
3. retweeted_status — существует, только если твит является ретвитом, содержит информацию о ретвитнутом твите,
4. lang — язык,
5. entities — информация о хэштегах и ссылках, которые упоминаются в твите,
6. text — текст твита.

Каждое событие о создании твита обрабатывается. Результатом обработки является структура данных, в которой содержится вся необходимая информация о созданном твите. Обработка происходит следующим образом:

1. Если твит не на русском языке, он отбрасывается.
2. Если твит является ретвитом, то взводится специальный флажок и дальнейшая работа происходит с исходным ретвитнутым твитом. Это делается для того, чтобы получить полноценный текст исходного твита.
3. Из поля entities извлекается информация о хэштегах и ссылках, встречаемых в твите.

Вся полученная информация помещена в хранилище твитов. Хранилище твитов реализованно использованием Python библиотеки shelve.

### 3.1.2. Получение новостных статей

Получение новостных статей происходит через RSS потоки — специальное API, предоставляемое интернет ресурсами и позволяющее получать информацию в формате RSS. *RSS* — семейство XML-форматов, предназначенных для описания лент новостей, анонсов статей, изменений в блогах и т.п. Формат RSS выбран ввиду его поддержки всеми популярными новостными источниками. Для работы с RSS потоками использована Python библиотека feedparser, позволяющая скачивать и анализировать данные в формате RSS.

RSS поток представляет собой переодически обновляемый список статей. Каждая статья обладает рядом параметров в работе использованы следующие:

1. published — дата создания,
2. summary — краткое изложение новостной статьи,
3. link — URL, который ведёт на описываемую новостную стать.,
4. title — заголовок новостной статьи,

Скачивание RSS потоков происходит следующим образом: переодически получается актуальное состояние всех RSS потоков, из них вычленяются все новые статьи, которые предобрабатываются и добавляются в хранилище новостей. Хранилище новостей реализованно с использованием Python библиотеки shelve.

### 3.1.3. Расшифровка сокращённых URL

*Сокращение URL* — это сервис, предоставляемый разными компаниями, заключающийся в создании дополнительного, в общем случае более короткого URL, введущего на искомый адрес. Обычно применяется с целью экономии длины сообщения или для предотвращения непреднамеренно искажения URL. В общем случае механизм сокращений реализовывается путём переадресации короткого URL на искомый.

В твиттере все ссылки автоматически сокращается с помощью сервиса *t.co*. Также многие ссылки добавляются в твиттер уже сокращёнными через сторонние сервисы. Для автоматического выявления связей между твитами и новостями с целью построения тестового набора данных необходимо уметь по сокращённому URL получить исходный.

*Расшифровка сокращённых URL* — процесс получения по сокращённому URL исходного адреса. На практике часто встречается применение сокращения URL каскадом: сокращение уже сокращённого URL, в таком случае расшифровка заключается в получении исходного URL, который не является сокращённой ссылкой. Можно трактовать задачу расшифровки следующим образом: необходимо получить URL адрес на котором завершится процесс переадресации.

Расматриваемая задача требует обработки большого количества твитов и следовательно большого количества расшифровок сокращённых URL ( в главе 3.2 получено, что количество ссылок, требуемых для анализа превышает $10^5$). Поэтому возникает требование к повышенной эффективности решения.

В качестве базового решения используется стандартный API языка Python, позволяющий получить содержимое веб-страницы по URL, а следовательно адрес целевой страницы на которую вела сокращённая ссылка. Случаи в которых исходный URL не был получен, будем называть ошибочными. Базовое решение было оптимизировано следующим образом:

1. Работа только с заголовками ответа. Это позволило снизить количество данных пересылаемых по сети. Работа с заголовками требует логики для принятия решения об остановке — то есть выявления искомого URL.

2. Использование многопоточности. Так как большую часть времени код, получающий заголовок страницы ждёт ответа сервера, то асинхронность позволит значительно увеличить быстродействие.

3. Использование «воронки» данных. При увеличении количества потоков стало появляться большее количество ошибок, ввиду того, что загруженность интернет-канала повышает время ответа http-запросов. Для их снижения было выбран подход «воронки» данных с последующей коррекцией ошибок. Данный подход на первом этапе обрабатывает все ссылки в $N$ потоков, на втором этапе все ошибочные ссылки полученные на первом обрабатываются в $\frac{N}{10}$ потоков и так далее, вплоть до 1 потока на итерацию.

### 3.1.4. Выявление источников новостей

Задача выявления источников новостей требует статистического исследования ссылок, которые встречаются в твитах. Для определения ссылок ведущих на новостные источники из всех URL извлекалось полное доменное имя (в дальнейшем доменное имя). Также стоит отметить, что новостные агрегаторы (к примеру Яндекс-новости, Рамблер-новости) не рассматривались ввиду того, что они агрегируют очень большое количество новостных статей с множества разнородных источников. То есть информацию с новостных агрегаторов очень сложно собрать и в дальнейшем дорого обрабатывать.

Для грубой оценки использована выборка 1, содержащая 35704 твитов, 13670 ссылок, 12510 уникальных ссылок. Статистика по 20 наиболее часто встречаемым доменным именам в выборке 1 представленна в таблице 2.

Как видно из таблицы 2 популярные новостные агенства составляют лишь малую долю от общего количества используемых ссылок (3.3%). Для получения более точной количественной информации за неделю собрана выборка 2, содержащая 341863 твитов, 134945 ссылок, 115940 уникальных ссылок. Статистика по 20 наиболее часто используемым доменным именам в выборке 2 представленна в таблице 3.

Как видно из таблицы 3 среди твитов, собранных на довольно большом промежутке времени (неделя), популярные новостные источники составляют лишь малую долю от общего числа употребляемых ссылок (3%).

В работе одновременно использовано 5 самых популярных новостных источников, а именно: `ria.ru`, `lifenews.ru`, `lenta.ru`, `russian.rt.com`, `www.gazeta.ru`.

Таблица 2: 20 наиболее часто вречаемых доменных имён в выборке 1 (всего 12510 уникальных ссылок)

| **Доменное имя** | **Количество ссылок** | **Процент от общего числа ссылок** | **Новостной источник** |
|---|---|---|---|
| twitter.com | 3521 | 25.76 | нет |
| www.facebook.com | 1418 | 10.37 | нет |
| t.co | 405 | 2.96 | нет |
| www.youtube.com | 315 | 2.30 | нет |
| news.yandex.ru | 239 | 1.75 | нет |
| su.epeak.in | 214 | 1.57 | нет |
| www.instagram.com | 198 | 1.45 | нет |
| www.periscope.tv | 191 | 1.40 | нет |
| l.ask.fm | 121 | 0.89 | нет |
| lifenews.ru | 109 | 0.80 | да |
| ria.ru | 108 | 0.79 | да |
| vk.com | 93 | 0.68 | нет |
| news.7crime.com | 82 | 0.60 | нет |
| lenta.ru | 74 | 0.54 | да |
| russian.rt.com | 61 | 0.45 | да |
| linkis.com | 57 | 0.42 | нет |
| www.gazeta.ru | 53 | 0.39 | да |
| tass.ru | 43 | 0.31 | да |
| www.swarmapp.com | 42 | 0.31 | нет |
| pi2.17bullets.com | 36 | 0.26 | нет |

Таблица 3: 20 наиболее часто вречаемых доменных имён в выборке 2 (всего 115940 уникальных ссылок)

| **Доменное имя** | **Количество ссылок** | **Процент от общего числа ссылок** | **Новостной источник** |
|---|---|---|---|
| twitter.com | 36807 | 31.75 | нет |
| apps.facebook.com | 6234 | 5.38 | нет |
| www.youtube.com | 3659 | 3.16 | нет |
| m.vk.com | 2400 | 2.07 | нет |
| www.periscope.tv | 2215 | 1.91 | нет |
| news.yandex.ru | 2041 | 1.76 | нет |
| www.instagram.com | 1798 | 1.55 | нет |
| su.epeak.in | 1624 | 1.4 | нет |
| www.facebook.com | 1406 | 1.21 | нет |
| lifenews.ru | 888 | 0.77 | да |
| ria.ru | 863 | 0.74 | да |
| l.ask.fm | 803 | 0.69 | нет |
| vk.com | 696 | 0.6 | нет |
| lenta.ru | 647 | 0.56 | да |
| pi2.17bullets.com | 577 | 0.5 | нет |
| news.7crime.com | 567 | 0.49 | нет |
| russian.rt.com | 564 | 0.49 | да |
| www.gazeta.ru | 523 | 0.45 | да |
| linkis.com | 485 | 0.42 | нет |
| ask.fm | 430 | 0.37 | нет |

## 3.2. Описание собранных данных

Набор данных сформирован на основе заранее собранной и предподготовленной информации. Для формирования набора данных использовалась информация собранная с 06.04.2016 по 17.04.2016. Было получено множество, основные характеристики которого приведены в таблице 4.

Таблица 4: Сводная характеристика по собранному множеству твитов и новостей за период с 06.04.2016 по 17.04.2016

| **Метрика** | **Значение** |
|---|---|
| Количество твитов | 495552 |
| Количество новостей | 13711 |
| Количество твитов, содержащих ссылку | 150510 |
| Количество уникальных ссылок, встречаемых в твитах | 101017 |
| Количество твитов, содержащих ссылку на новости из рассматриваемых новостных источников | 4324 |
| Количество уникальных ссылок на новости из рассматриваемых новостных источников | 2979 |

В дальнейшей под собранными данными будет подразумеваться описанное в таблице 4 множество.

## 3.3. Разметка наборов данных

Для построения требуемого в работе набора данных необходимо найти множество связей твит-новость. Процесс поиска связей твит-новость называется *разметкой набора данных*. В работе использовано два способа разметки:

1. автоматическая разметка набора данных;
2. ручная разметка набора данных.

В результате разметки получено большое количество пар твит-новость, в которых твит, практически полностью совпадает с заголовком новости. В дальнейшем, для удобства обозначение подобных пар, будем называть связь твит-новость *тривиальной*, если в заголовке статьи встречается менее половины слов из твита.

### 3.3.1. Автоматическая разметка набора данных

Автоматически построенный набор данных состоит из всех собранных новостей и твитов, которые содержат ссылку на одну из собранных новостей. Автоматически разметив полученную

информация, построили множество состоящее из 4324 твитов, 13711 новостей, а также 4324 связей между ними.

Полученный набор данных был проанализирован, с целью выявления количества нетривиальных связей. Для каждого твита был взят его текст, для каждой новости — заголовок. Все полученные тексты поэтапно преобразованы согласно алгоритму, который сопоставляет тексту множество слов и состоит из следующей последовательность действий:

1. текст конвертируется в формат unicode;
2. текст разбивается на токены;
3. полученное множество токенов очищается от токенов, не являющихся словами;
4. из множества удаляются все слова входящие в словарь стопслов;
5. из множества удаляются все дубли.

На основе предподготовленных данных для каждой пары твит, связанная с твитом новость измерялось две метрики: длина пересечения слов твита и слов новости, нормализованная по длине новости; количество слов в твите, которые не встречаются в новости.

На рисунке 2 изображена зависимость количества пар твит-новость от длины пересечения слов твита и слов новости, нормализованной по длине новости. Как видно из рисунка 2

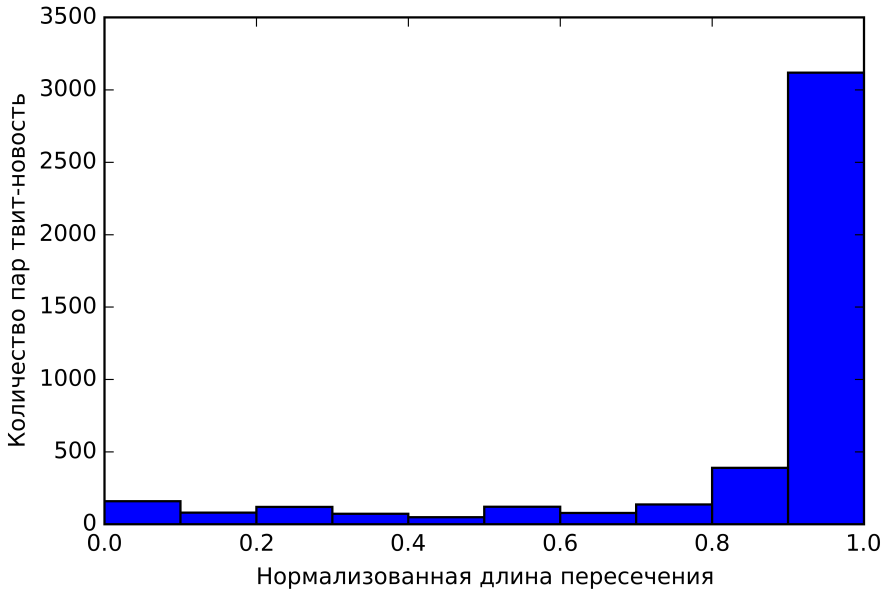


Рисунок 2 — Зависимость количества пар твит-новости от нормализованный длины пересечения множества слов (автоматически размеченный набор данных).

слова в подавляющем большинстве твитов полностью совпадают со словами в соответствующей новости. Среди 4324 пар твит-новость в 3082 парах твит полностью совпадает с заголовком новости. Остаётся 1242 пар, которые не являются просто копией заголовка, среди этих пар нас интересуют те, в которых твит не является обрезанной частью заголовка статьи.

Для выявления количества пар, где твит содержит информацию не содержащуюся в заголовке статьи посмотрим на зависимость количества пар твит-новость от процента уникальных слов в твите, эта зависимость изображена на рисунке 3. Как видно из рисунка 3 количество

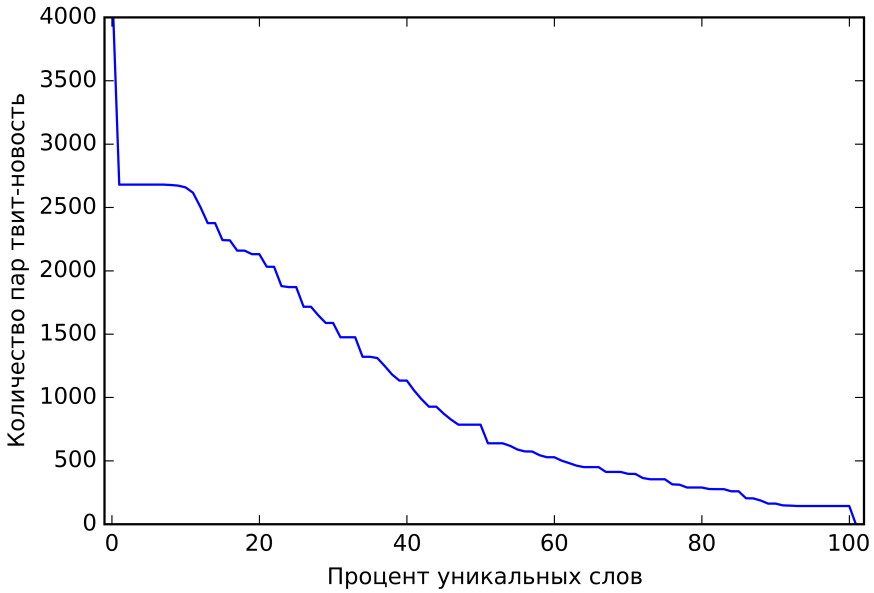


Рисунок 3 — Зависимость количества пар твит-новости от процента уникальных слов в твите (автоматически размеченный набор данных).

пар твит-новость, образующих нетривиальную связь, достаточно мало.

На основе исследования зависимости можно получить грубую оценку количества нетривиальных связей. В исследуемом наборе данных таких связей порядка 500-1000, что очень мало и составляет примерно 12-23% от общего числа пар.

### 3.3.2. Ручная разметка набор данных

Для получения большего количества нетривиальных пар твит-новость была предпринята ручная разметка набора данных. Для ручной разметки были предподготовлены данные. Основные этапы предподготовки данных:

1. на основе множества новостей строится список именованных сущностей $L$;
2. случайным образом берётся подмножество $T$ множества твитов;

3. из полученного множества $T$ удаляются все твиты, которые удолетворяют следующим правилам:

   а) твит является ретвитом,

   б) твит содержит ссылку на URL с плохим доменным именем (под *плохим доменным именем* подразумевается доменное имя, которое достаточно популярно и не ведёт на новостной источник, в работе использовался следующий список плохих доменных имён: `apps.facebook.com`, `ask.fm`, `twitter.com`, `apps.facebook.com`, `www.instagram.com`, `vk.cc`),

   в) в приведённом к нормальной форме (определение нормальной формы находится в главе**??**) тексте твита содержится менее 2 слов из списка именованных сущностей $L$;

4. с помощью метода определения схожести текстов на основе частотности употребления слов каждому твиту сопоставляется 10 наиболее схожих с ним новостей.

В качестве результата предподготовки получили множество пар твит-ранжированный список новостей.

Предподготовленные данные были размечены. Разметка заключается в записи специальной отметки рядом с подходящей новости из предложенного списка для каждого твита. Для каждого твита отмечалась только одна, наиболее подходящая новость, или не отмечалось ни одной.

Было сформировано множество из 7373 пар твит-ранжированный список новостей. В нём было выявлено 1600 связей твит-новость. Получаем набор данных, состоящий из 1600 твитов, 13711 новостей, а также 1600 связей между ними.

Для сравнения вручную построенного набора данных с набором данных, полученным автоматически были построены две зависимости — зависимость количества пар твит-новость от длины пересечения слов твита и слов новости и зависимость количества пар твит-новости от процента уникальных слов в твите. На рисунке 4 изображена зависимость количества пар твит-новость от длины пересечения слов твита и слов новости, нормализованная по длине новости. Как видно из рисунка 4 было получено распределение намного более близкое к равномерному, чем в случае автоматически размеченного набора данных.

На рисунке 5 изображена зависимость количества пар твит-новость от процента уникальных слов в твите. Как видно из рисунка 5 количество твитов более с чем половиной уникальных слов сравнимо с аналогичным количеством в автоматически размеченном наборе данных, несмотря на то, что автоматически размеченный набор данных почти в три раза больше, чем вручную размеченный набор данных. Количественные значения полученных метрик приведены в таблице 5.

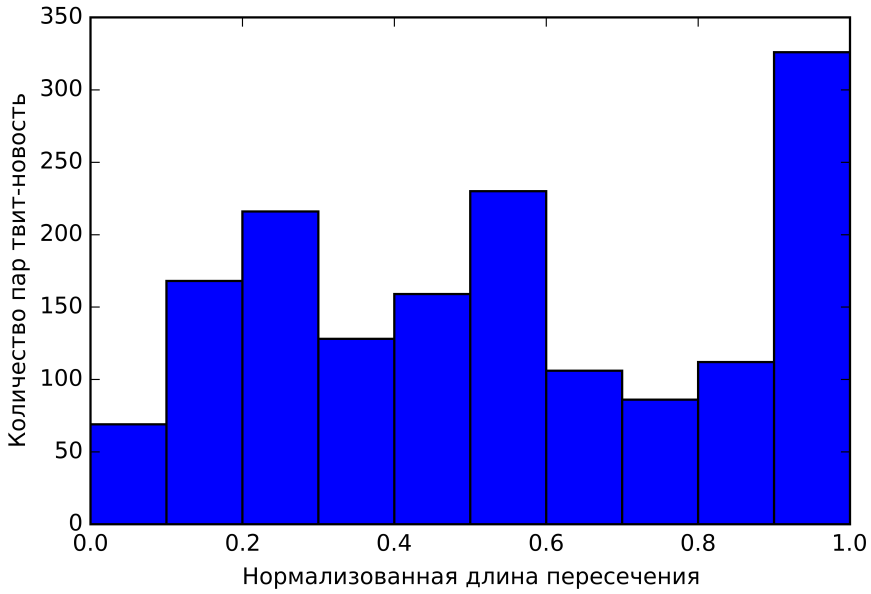


Рисунок 4 — Зависимость количества пар твит-новости от нормализованный длины пересечения множества слов (вручную размеченный набор данных).

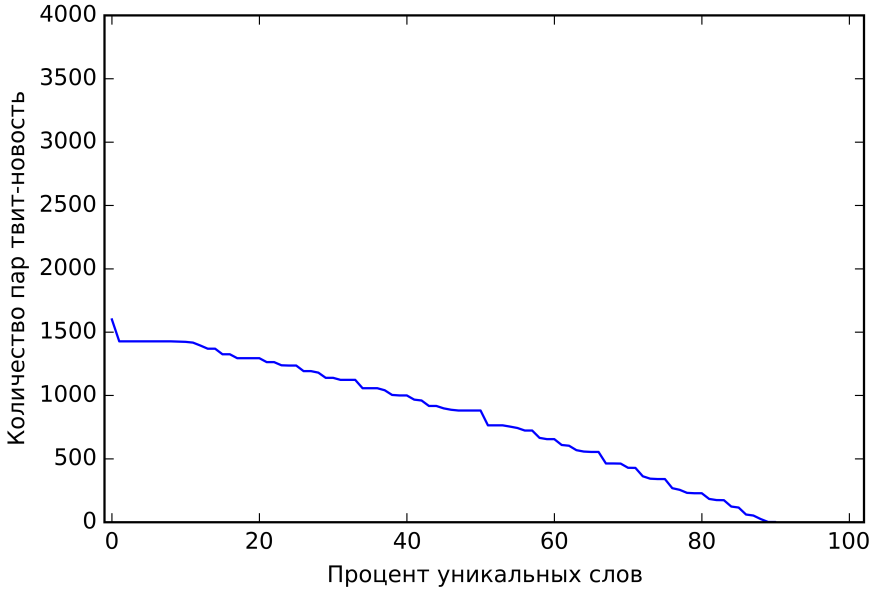


Рисунок 5 — Зависимость количества пар твит-новости от процента уникальных слов в твите (вручную размеченный набор данных).

Как видно из таблицы 5 вручную собранный набор данных намного более качественный, чем автоматический. Но как в ручном, так и в автоматическом наборе данных содержится очень мало нетривиальных связей твит-новость (в сравнении с количеством новостей).

Таблица 5: Сравнение количества твитов

| **Метрика** | **Автоматически размеченный набор данных** | **Вручную размеченный набор данных** |
|---|---|---|
| Количество связей | 4324 | 1600 |
| Количество нетривиальных связей | 746 | 976 |
| Процент нетривиальных связей от общего числа связей (%) | 17.25 | 61.00 |

## 3.4. Построение связей текст-текст

Построение связей текст-текст предполагает поиск потенциально семантически близких текстов. При построении связей текст-текст было использовано три способа:

1. построение связей на основе общих хэштегов,
2. построение связей на основе общих именованных сущностей,
3. построение связей на основе близости по времени.

**Связь твитов с помощью хэштегов.** Из твитов извлекаются все хэшетеги. Затем в хэштеги превращаются все слова во всех твитах, которые совпали с ранее извлечёнными хэштегами. Для каждого твита и для каждого хэштега извлекается $k$ твитов, которые содержат этот хэштег. Если хэштег появлялся в более чем $k$ твитах, то берём $k$ твитов наиболее близких во времени к исходному.

**Связь твитов с помощью именованных сущностей.** К краткому изложению новостей применяются методы извлечения именованных сущностей. Для каждого твита, содержащего именованную сущность в виде отдельного слова извлекается $k$ твитов, которые содержат эту же именованную сущность. Если именованная сущность содержалась более чем в $k$ твитах, то берём $k$ твитов наиболее близких во времени к исходному.

**Связь твитов и новостей на основе близости по времени** Для каждого твита (новости) выбираем $k$ связей с наиболее схожими твитами (новостями) в окрестности 24 часов.

Построение связей текст-текст было реализовано для $k = 10$. Для избежания появления большого количества «лишних» записей использовался набор эвристических ограничений:

1. удаление слишком популярных хэштегов (слишком популярным считаем хэштег, который встретился более чем в 10 твитах из обучающей выборки);
2. твиты, считаются связанными когда содержат не менее 2 общих хэштегов или именованных сущностей;

3. связь не устанавливается если тексты слишком похожи (если косинусная мера близости текстов больше 0.99);

4. в случае установления связей на основе времени публикации и схожести текстов, слишком не похожие тексты отбрасываются (слишком не похожие тексты это тексты с мерой близости меньше чем 0.3).

## 3.5. Сформированные наборы данных

На основе собранной информации было сформировано несколько базовых эталонных наборов, а именно:

1. auto — автоматически размеченный набор данных;

2. manual – вручную размеченный набор данных;

3. total — набор данных состоящий из объединения всех размеченных связей (то есть объединение auto и manual);

4. cutted — набор данных, основанный на наборе total, в котором количество новостей сравнимо с количеством твитов (набор данных создавался для изучения влияния соотношения количества новостей и твитов на качество установления связей).

Также рассматриваются эталонные наборы данных без тривиальных связей, подобный набор образуется путём удаления из базового эталонного набора твитов, создающих тривиальную связь. Обозначим эталонные наборы данных с удалёнными тривиальными связями как auto_nt, manual_nt, total_nt и cutted_nt, полученные путём удаления тривиальных связей из базовых эталонных наборов auto, manual, total и cutted, соответственно.

Ключевая информация характеризующая эталонные наборы представлена в таблице 6.

Таблица 6: Сводная таблица по эталонным наборам данных

| **Набор данных** | **Количество твитов** | **Количество новостей** |
|---|---|---|
| manual | 1600 | 13711 |
| auto | 4324 | 13711 |
| total | 5798 | 13711 |
| cutted | 5798 | 6011 |
| manual_nt | 976 | 13711 |
| auto_nt | 746 | 13711 |
| total_nt | 1709 | 13711 |
| cutted_nt | 1709 | 6011 |

# 4. Установления взаимосвязей между новостями и твитами

Задача автоматического установления связей между твитами и новостями решена посредством написания программного комплекса, который обладает следующими возможностями:

1. сбор необходимой для решения задачи информации;
2. получение рекомендаций новостей для произвольных твитов;
3. вариативность в выборе метода для построения рекомендаций;
4. возможность получить информацию о качестве используемого метода.

Поставленная задача достигается за счёт реализации отдельных преобразований над данными в рамках написанного программного комлекса с использованием языка программирования Python версии 2.7. Далее приводится описание реализованных преобразований и подробный разбор отдельных моментов.

## 4.1. Архитектура

Программный комплекс позволяет производить все требуемые для решения задачи преобразования над данными, а именно:

1. получение данных из твиттера;
2. получение данных из новостной rss-ленты;
3. расшифровка сокращённых URL;
4. автоматическое построение набора данных;
5. построение моделей для методов WTMF и WTMF-G;
6. построение рекомендаций на основе методов WTMF, WTMF-G и TF-IDF;
7. оценка качества рекомендаций;
8. получение результатов рекомендаций в пригодном для чтения формате;

Результаты всех преобразований, за исключением оценки качества и получения рекомендаций, хранятся в специальном промежуточном хранилище (побробное описания хранилища производится в разделе 5).

Каждое преобразование, в общем случае, независимо от других. Для построения рекомендаций, необходимо выполнить цепочку преобразований. Примеры цепочек преобразований для получения данных, построения рекомендаций и оценки их качества приводятся ниже.

Визуализация цепочек преобразований производится при помощи блок-схем [11]. Для удобства восприятия блоки действия (изображаются прямоугольником) выделяются зелёным цветом, а прочие используемые блоки, такие как ввод-вывод данных (изображаются параллелограммом) и хранимые данные (изображаются фигурой, представляющей собой прямоугольник, в котором две противолежащие стороны заменены на две одинаковые и параллельные кривые, совпадающие с секцией окружности), выделяются синим цветом.

Получение данных заключается в скачивании новостей из RSS потоков и твитов, с использованием Twitter Streaming API, в течение длительного промежутка времени, с последующим помещением всех данных в промежуточное хранилище. В работе в качестве хранилища выступает python shelve. Получение данных в виде блок-схемы изображено на рисунке 6.

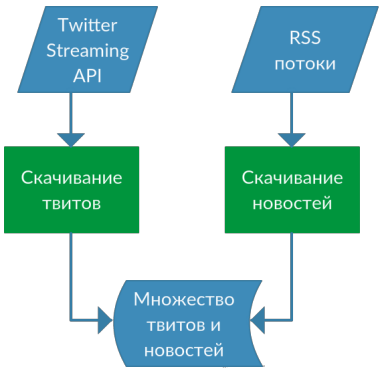


Рисунок 6 — Блок-схема получения данных

На основе полученного множества новостей и твитов происходит автоматическое построение набора данных. В рамках автоматического построения набора данных происходит расшифрка сокращённых URL. Набор данных эта структура состоящая из списка новостей и списка твитов, где для каждого твита указана ссылка на единственную новость.

Результатом работы всех реализованных методов является сопоставление численных векторов (векторов для сравнения) каждому обрабатываемому тексту, с помощью которых можно оценить насколько похожи любые два текста.

Метод TF-IDF не имеет стадии обучения модели, поэтому применяется непосредственно к набору данных и получает вектора для сравнения, для всех текстов, которые были переданы ему на вход. Получаемые вектора обладают размерностью совпадающей с размером корпуса.

В отличие от метода TF-IDF методы WTMF и WTMF-G состоят из двух стадий: обучения и применения модели. На стадии обучения методы строят модель (в сериализованной модели помимо самой модели содержится набор данных, на основе которого была построена

модель) и получают вектора для сравнения для всех элементов набора данных. На стадии применения методы WTMF и WTMF-G на основе ранее построенной модели для произвольного множества твитов строят векторы для сравнения полученных на вход твитов и новостей из набора данных.

На основе множества, состоящего из твитов и новостей, для каждого элемента в котором существует вектор для сравнения, строятся рекомендации. Рекомендации представляют собой множество твитов, к каждому из которых сопоставлен ранжированный по мере убывания схожести список новостей.

На основе построенных рекомендаций можно как произвести оценку качества ранее использованного метода, так и получить их в виде текстового файла, который содержит информацию в пригодном для чтения формате. Оценка качества полученного метода происходит возможно, только если рекомендации были получены из набора данных.

Процесс оценки качества различных методов рекомендаций, а также получение рекомендаций для твитов из набора данных изображён на рисунке 7.

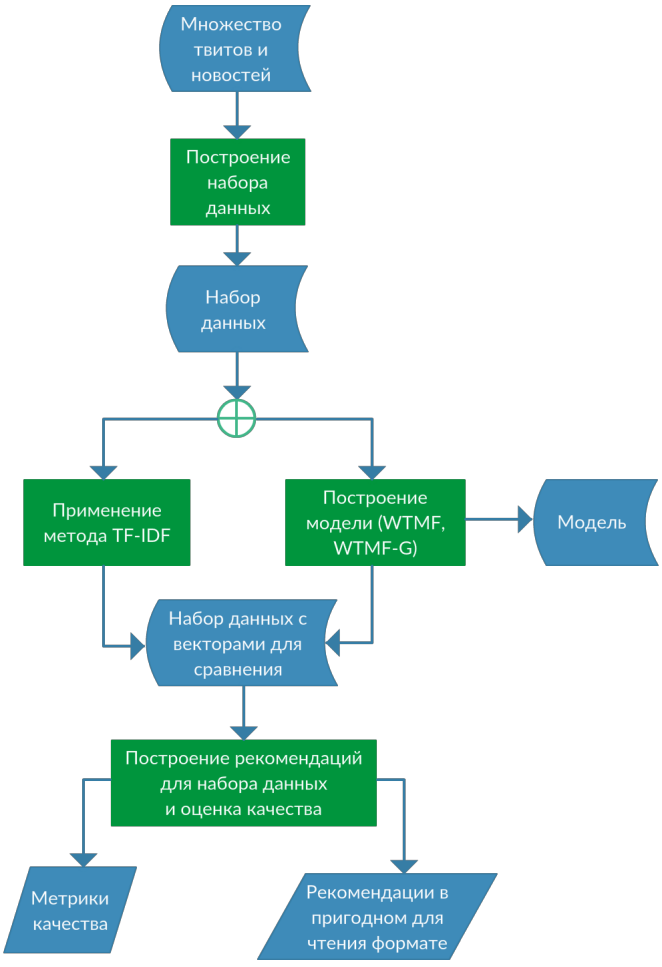


Рисунок 7 — Блок-схема процесса оценки качества используемых методов

Дополнительным результатом изображённого на рисунке 7 процесса является построенная модель (для методов WTMF и WTMF-G), которую можно применить на произвольное множество твитов. Процесс получения рекомендаций для произвольных твитов изображён на рисунке 8.

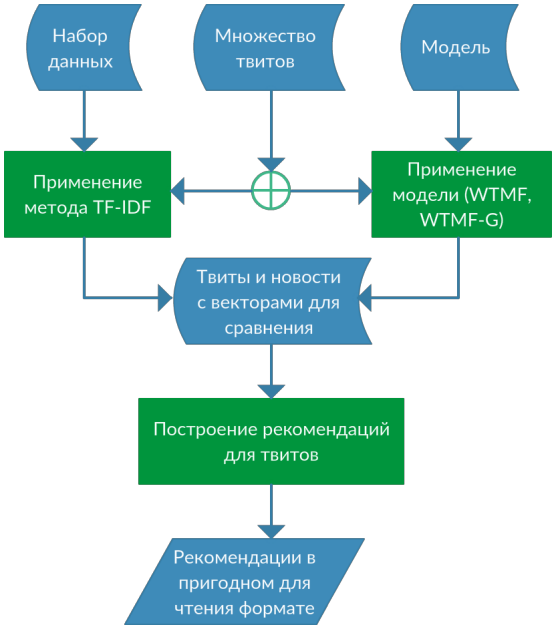


Рисунок 8 — Блок-схема процесса получения рекомендаций для различных методов

## 4.2. Обработка естественного языка

Работа посвящена поиску семантической близости текстов, поэтому в ней имеет место использование решений таких задач обработки естественного языка, как:

1. токенизация — разбиение предложения на слова;
2. лемматизация — процесс приведения словоформы к лемме;
3. извлечение именованных сущностей.

Описанные выше задачи решены с использованием набора сторонних библиотек для языка Python, а именно:

1. ntlk — платформа, для написания приложений на языке Python, обрабатывающих естественный язык;
2. pymorphy2 — морфологический анализатор;
3. polyglot — библиотека, позволяющая извлекать именованные сущности из текстов на разных языках.

Для решения задачи токенезации используется стандартный токенизатор, реализованный в ntlk. Задача лемматизации решается в случае русского языка с помощью морфологического анализатора pymorphy2, в случае английского языка с помощью морфологического анализатора WordNet, реализованного в ntlk.

Извлечение именованных сущностей происходит с помощью библиотеки polyglot. В используемой библиотеке реализуется выявление именованных сущностей на основе заранее сформированного и размеченного корпуса именованных сущностей. Корпус формируется на основе данных из Википедии.

## 4.3. Метод WTMF

Модель для метода WTMF построена на основе мнзаранее подготовленного набора данных. В контексте работы набор данных состоит из множества новостей и твитов, из которых в процессе работы извлекается набор текстов (для твита — текст твита, для новости — конкатенация заголовка и краткого изложения статьи).

По множеству текстов, которые получены из набора данных, построена модель, пригодная для сериализации, состоящая из матрицы $P$ (здесь и далее используются обозначения введённые в главе 1.2.4). Построение модели зависит от четырёх констант:

1. $K$ — размерность вектора, по которому производится сравнение (если TF-IDF матрица $X$ была размера $M \times N$, то по завершении работы алгоритма будут получены две матрицы $P$ размера $K \times M$ и $Q$ размера $K \times N$);

2. $I$ — число итераций алгоритма построения модели;

3. $w_M$ — коэффициент, задающий вес негативного сигнала при построении матрицы весов $W$;

4. $\lambda$ — регуляризирующий член.

Применение полученной модели на множество твитов представляет собой следующий процесс: сначала строится TF-IDF матрица $X$ для новостей из набора данных и множества твитов, затем на основе новой матрицы $X$ строится весовая матрица $W$, и наконец на основе построенных матриц $X$ и $W$ и посчитанной на этапе обучения матрицы $P$ выполняется половина итерации алгоритма обучения, а именно получение матрицы $Q$ по матрице $P$:

$$Q_{\cdot,j} = (PW'_jP^T + \lambda I)^{-1}PW'_jX_{j,\cdot}.$$

В результате получаем вектора для сравнения твитов из заданного множества.

## 4.4. Метод WTMF-G

Построение модели для метода WTMF-G основывается на построение модели метода WTMF. Набор данных состоит из множества новостей и твитов и связей вида текст-текст, из которых, в процессе работы извлекается набор текстов. (для твита — текст твита, для новости — конкатенация заголовка и краткого изложения статьи).

По множеству текстов, которые получены из набора данных, построена пригодная для сериализации модель, представляющая собой матрицу $P$. Построение модели зависит от четырёх констант:

1. $K$ — размерность вектора, по которому производится сравнение (если TF-IDF матрица $X$ была размера $M \times N$, то по завершении работы алгоритма будут получены две матрицы $P$ размера $K \times M$ и $Q$ размера $K \times N$);

2. $I$ — число итераций алгоритма построения модели;

3. $w_M$ — коэффициент, задающий вес негативного сигнала при построении матрицы весов $W$;

4. $\delta$ — коэффициент, задающий степень влияния связей вида текст-текст.

Применение полученной модели на множество твитов производится аналогично применению модели для метода WTMF за исключением двух моментов: во-первых, необходимо на

основе новостей из набора данных и множества твитов перестроить связи текст-текст, во-вторых получение матрицы $Q$ происходит по следующей формуле:

$$Q_{\cdot,j} = (PW'_jP^T + \lambda I + \delta L_j^2 Q_{\cdot,n(j)} diag(L^2_{n(j)}) Q^T_{\cdot,n(j)})^{-1} (PW'_j X_{j,\cdot} + \delta L_j Q_{\cdot,n(j)} L_{n(j)}).$$

В результате получаем вектора для сравнения твитов из заданного множества.

## 4.5. Эффективная работа с матрицами

Построение и применение моделей WTMF и WTMF-G требует большого количества операций над матрицами, что на практике занимает продолжительное время. Поэтому актуальна задача по повышению эффективности работы с матрицами.

Для эффективной работы с матрицами используются программные библиотеки для языка Python numpy и scipy (базируется на библиотеке numpy и расширяет её функционал).

Повышение производительности при работе с матрицами производится на примере оптимизации времени расчёта формулы получения строк матрицы $P$, которая используется при построении моделей WTMF и WTMF-G. На каждой итерации построения модели происходит многократное выполнение формулы (число выполнений порядка $10^4$, зависит от размера корпуса):

$$P_{i,\cdot} = (QW'_iQ^T + \lambda I)^{-1} QW'_iX^T_{i,\cdot}.$$

В начале была написана наивная реализация алгоритма, которая показала производительность, не приемлемую в рамках решения задачи. Затем наивная реализация оптимизировалась следующим образом:

1. переход к перемножению матриц с использованием высокопроизводительной библиотеки для языка C OpenBlass (в библиотеке numpy существует возможность перейти к использованию для работы с матрицами некоторых библиотек, написанных на языке C [7]);

2. сохранение в отдельной переменной переиспользуемых результатов вычислений над матрицами;

3. переписывание кода для работы с разреженными матрицами;

4. удаление лишних приведений матриц к формату python list и обратно.

Результаты оптимизации приведены в таблице 7.

Получили, что оптимизированное решение работает в 325 раз быстрее наивной реализации. Дальнейшая оптимизация не производилась, так как получено решение работающее за приемлемое время.

Таблица 7: Оптимизация работы с матрицами

| **Добавленная оптимизация** | **Время за 100 итераций (с)** | **Прирост производительности (раз)** |
|---|---|---|
| Наивная реализация | 205 | 1 |
| Перемножение с помощью OpenBlass | 55 | 3.73 |
| Переиспользование результатов | 15.15 | 3.63 |
| Работа с разреженными матрицами | 0.75 | 20.2 |
| Сокращение количества приведений типов | 0.63 | 1.21 |

# 5. Руководство пользователя

Программный комплекс состоит из двух приложений, каждое из которых устанавливается и используется в отдельности.

- twnews_consumer — приложение, предназначенное для того, чтобы получать твиты с твиттера и новости с rss каналов.
- twnews — приложение, позволяющее по твитам и новостям, произвести все необходимые для автоматического построения рекомендций преобразования данных и на основе полученных признаков произвести обучение и оценку модели.

Оба пакета ориентированы на работу в операционных системах семейства linux. Для начала работы необходимо получить содержимое git-репозитория: https://github.com/art-vybor/twnews.git, в котором расположены исходный код обоих приложений. Если установлен пакет git, то получение git-репозитория происходит с помощью следующей команды:

```
$ git clone https://github.com/art-vybor/twnews.git
```

Для сборки приложений необходимо иметь установленный менеджер пакетов языка Python — pip. С помощью него необходимо установить Python библиотеку setuptools:

```
$ pip install setuptools
```

Отдельно стоит отметить, что для корректной работы пакетов twnews и twnews_consumer, все директории, указываемые в конфигурации пакетов должны быть созданы заранее.

## 5.1. Пакет twnews_consumer

Исходный код приложения twnews_consumer расположен в папке consumer в корне репозитория. Приложение позволяет выкачивать и сохранять твиты и новости в формате, удобном для дальнейшей работы пакета twnews.

### 5.1.1. Конфигурирование

Конфигурирование пакета twnews_consumer производится в файле twnews_consumer/defaults.py. После изменения параметров, необходимо переустановить пакет. Описание задаваемых параметров находится в таблице 8.

### 5.1.2. Установка

Для установки приложения, необходимо зайти в папку consumer с исходным кодом пакет twnews_consumer, находящуюся в корне репозитория, и выполнить команду:

Таблица 8: Описание конфигурации пакета twnews_consumer

| Имя параметра | Пример значения | Описание |
|---|---|---|
| LOG_FILE | '/var/log/twnews_consumer.log' | Путь до файла с логом |
| LOG_LEVEL | logging.INFO | Уровень подробности лога |
| TWNEWS_DATA_PATH | '/home/user/twnews_data/' | Путь до директории, в которую будут сохранены данные |
| RSS_FEEDS | {<br>'lenta': {'rss_url':<br>'http://lenta.ru/rss'},<br>'rt': {'rss_url':<br>'https://russian.rt.com/rss'},<br>} | Новостные источники, которые требуется выкачать |
| TWEETS_LANGUAGES | ['ru'] | Список языков, твиты с использованием которых выкачиваются из твиттера |

```
$ make install
```

Во время установки, с целью распаковки секретного ключа, необходимого для работы с API твиттера, требуется ввести секретный пароль.

### 5.1.3. Использование

Результатом работы пакета является множество новостей и твитов, выкаченных за время работы программы. Для новостей сохраняются заголовок, краткое описание, ссылка на новость, время публикации и имя ресурса, на котором новость была опубликована. Для твитов сохраняются текст, численный уникальный идентификатор ,время публикации, информацию о хэштегах и ссылках и является ли он ретвитом.

Приложение обладает интерфейсом командной строки. Для старта скачивания новостей, необходимо запустить команду:

```
$ twnews_consumer download --news
```

Для старта скачивания сообщения твиттера, необходимо запустить команду:

```
$ twnews_consumer download --tweets
```

Для завершения скачивания, необходимо использовать сочетание клавиш «Ctrl-C». Приложение обработает прерывание и корректно завершится.

Приложение записывает вспомогательную информацию о своём статусе и обработанных ошибках в файл лога. Поэтому из файла лога можно узнать актуальную информацию о работе приложения. Пример:

```
$ tail -f  /var/log/twnews_consumer.log
```

```
2016-04-05 11:37:14: RSS> Start consume rss feeds
2016-04-05 11:37:17: TWITTER> Starting write to /mnt/yandex.disk/twnews_data/
    logs/tweets.shelve
2016-04-05 11:37:17: TWITTER> Starting to consume twitter
2016-04-05 12:33:32: TWITTER> ('Connection broken: IncompleteRead(0 bytes read
    , 512 more expected)', IncompleteRead(0 bytes read, 512 more expected))
```

Приложение twnews_consumer устойчиво к ошибкам, возникающим при получении новостей и твитов, востановление работы заключается в перезапуске скачивания информации. Ввиду этого приложение реализует скачивание данных согласно семантики *at-most-once* — гарантируется отсутствие дублей, но допускается потеря данных.

## 5.2. Пакет twnews

Пакет twnews располагается в папке core в корне репозитория. Приложение позволяет обрабатывать данные полученные с помощью консьюмера с целью автоматического установления связей между твитами и новостными статьями и оценки качества полученного решения.

### 5.2.1. Конфигурирование

Конфигурирование пакета twnews производится в файле twnews/defaults.py. После изменения параметров, необходимо переустановить пакет. Описание задаваемых параметров находится в таблице 9.

### 5.2.2. Установка

Для установки приложения, необходимо зайти в папку core с исходным кодом пакет twnews, находящуюся в корне репозитория, и выполнить команду:

```
$ make install
```

Для повышения производительности рекомендуется вручную собрать пакет numpy с использованием низкоуровневой математической библиотеки OpenBLAS [7].

### 5.2.3. Использование

Финальным результатом работы приложения является получение рекомендаций для произвольных твитов. Построение рекомендаций происходит в несколько стадий (каждая стадия представляет собой отдельный вызов приложения), количество стадий различается для различных методов рекомендаций.

Приложение обладает интерфейсом командной строки. Для удобства использования функционал приложения разбивается на набор независимых друг от друга точек входа, каж-

Таблица 9: Описание конфигурации пакета twnews

| **Имя параметра** | **Пример значения** | **Описание** |
|---|---|---|
| LOG_FILE | '/var/log/twnews.log' | Путь до файла с логом |
| LOG_LEVEL | logging.INFO | Уровень подробности лога |
| TWNEWS_DATA_PATH | '/home/user/twnews_data/' | Путь до директории, в которую были скачены данные приложением twnews_consumer |
| DATASET_FRACTION | 1.0 | Часть множества скаченных твитов на основе которых происходит построение набора данных |
| TMP_FILE_DIRECTORY | '/tmp/twnews/' | Путь до директории в которую будут сохранены временные данные |
| DEFAULT_WTMF_OPTIONS | { 'DIM': 90, 'WM': 0.95, 'ITERATIONS': 1, 'LAMBDA': 1.95 } | Настройки метода WTMF |
| DEFAULT_WTMFG_OPTIONS | { 'DIM': 220, 'WM': 5, 'ITERATIONS': 1, 'DELTA': 0.06, 'LAMBDA': 6 } | Настройки метода WTMF-G |

дая из которых представляет интерфейс для выполнения одной из стадий работы построения рекомендаций. Список всех точек входа представлен в таблице 10.

В дальнейшем приводится более детальное описание каждой точки входа. Для каждой точки входа в качестве примера использования приводится две команды командной строки: вызов и результат вызова автоматически порождаемой приложением справочной информации, которая описывает параметры передаваемые в точку входа, и один из вариантов вызова точки входа.

Точка входа tweets_sample позволяет получить случайный набор твитов из собранных данных; параметризуется двумя необязательными параметрами length — задаёт количество твитов в результате, и output_dir — определяет директорию в которой будет сохранён файл с твитами; на выход в результирующей директории порождается файл с твитами, его имя печается на экран. Пример использования:

```
$ twnews tweets_sample -h
usage: twnews tweets_sample [-h] [--length LENGTH] [--output_dir OUTPUT_DIR]
```

Таблица 10: Описание точек входа пакета twnews

| **Название точки входа** | **Пример команды запуска** | **Описание** |
|---|---|---|
| tweets_sample | twnews tweets_sample | Получение случайного набора твитов из собранных данных |
| resolver | twnews resolver | Расшифровка сокращённых ссылок |
| build_dataset | twnews build_dataset | автоматическое построение набора данных |
| train | twnews train | Построение модели для методов WTMF и WTMF-G |
| apply | twnews apply | Применение модели для методов WTMG и WTMF-G |
| tfidf | twnews tfidf | Применение метода TF-IDF |
| build_recommendation | twnews build_recommendation | Построение рекомендаций |
| recommendation | twnews recommendation | Обработка полученных рекомендаций |

```
optional arguments:
  -h, --help            show this help message and exit
  --length LENGTH       num of tweets in sample (default: 10)
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)
$ twnews tweets_sample --length 100
```

Точка входа resolver позволяет расшифровать все сокращённые ссылки, используемые в скаченном приложением twnews_consumer множестве данных (параметр resolve). Отображение коротких ссылок в расшифрованные хранится отдельным файлом и в дальнешем используется при построении набора данных. Также точка входа resolver позволяет получить статистику по все расшифрованным ссылкам (параметр analyze). Пример использования:

```
$ twnews resolver -h
usage: twnews resolver [-h] (--resolve | --analyze)

optional arguments:
  -h, --help  show this help message and exit
  --resolve   resolve urls from all tweets (default: False)
  --analyze   print stats of resolved urls (default: False)

$ twnews resolver resolve
```

Точка входа build_dataset позволяет автоматически построить набор данных; параметризуется двумя параметрами unique_words — задаёт процент уникальных слов в твите для пар твит-новость, и output_dir — определяет директорию в которой будет сохранён файл с набором

данных; Имя построенного набора данных печается на экран. Пример использования:

```
$ twnews build_dataset -h
usage: twnews build_dataset [-h] [--unique_words UNIQUE_WORDS]
                            [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --unique_words UNIQUE_WORDS
                        percent of unique words in tweet by corresponding news
                        (default: 0.0)
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)

$ twnews build_dataset
```

Точка входа train позволяет построить модели для методов WTMF и WTMF-G; каждый метод параметризуется тремя параметрами dataset — название используемого набора данных, input_dir — директория где находится используемый набор данных и output_dir — директория в которую будет сохранён файл с модель и файл содержащий новости и твиты из набора данных с векторами для сравнения; Имя построенной модели печается на экран. Пример использования:

```
$ twnews train -h
usage: twnews train [-h] (--wtmf | --wtmf_g) --dataset DATASET
                    [--input_dir INPUT_DIR] [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --wtmf                wtmf method (default: False)
  --wtmf_g              wtmf_g method (default: False)
  --dataset DATASET     dataset name (default: None)
  --input_dir INPUT_DIR
                        input directory (default: /home/avybornov/tmp)
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)

$ twnews train --wtmf --dataset dataset_auto
```

Точка входа apply позволяет применить ранее построенные модели для методов WTMF и WTMF-G на набор твитов; каждый метод параметризуется четырьмя параметрами: model — название используемого набора данных, tweets — название файла с набором твитов, который был построен с использованием точки входа tweets_sample, input_dir — директория где нахо-

дится используемая модель и набор твитов; output_dir — директория в которую будет сохранён файл содержащий новости из набора данных (использованного на этапе обучения) и твиты из переданного файла с векторами для сравнения; Имя полученного файла печается на экран. Пример использования:

```
$ twnews apply -h
usage: twnews apply [-h] (--wtmf | --wtmf_g) --model MODEL --tweets TWEETS
                    [--input_dir INPUT_DIR] [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --wtmf                wtmf method (default: False)
  --wtmf_g              wtmf_g method (default: False)
  --model MODEL         model name (default: None)
  --tweets TWEETS       name of file with tweets (default: None)
  --input_dir INPUT_DIR
                        input directory (default: /home/avybornov/tmp)
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)
$ twnews apply --wtmf --model WTMF_model --tweets tweets_file
```

Точка входа tfidf позволяет применить метод TFIDF для нахождения векторов сравнения для новостей из набора данных и твитов из переданного файла (если файл не указан используются твиты из набора данных); параметризуется четырьмя параметрами dataset — название используемого набора данных, tweets — название файла с набором твитов, который был построен с использованием точки входа tweets_sample, input_dir — директория где находится используемый датасет и набор твитов; output_dir — директория в которую будет сохранён файл содержащий новости и твиты с векторами для сравнения; Имя порождённого файла, который содержит новости и твиты с векторами для сравнения, печается на экран. Пример использования:

```
$ twnews tfidf -h
usage: twnews tfidf [-h] --dataset DATASET --tweets TWEETS
                    [--input_dir INPUT_DIR] [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --dataset DATASET     dataset name (default: None)
  --tweets TWEETS       name of file with tweets (default: None)
  --input_dir INPUT_DIR
                        input directory (default: /home/avybornov/tmp)
```

```
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)
```

```
$ twnews tfidf --dataset dataset_manual --tweets tweets_file
```

Точка входа build_recomendation позволяет построить рекомендации по набору новостей и твитов из набора данных с векторами для сравнения (если указан файл содержащий твиты с векторами для сравнения, то твиты берутся из него); параметризуется четырьмя параметрами dataset_applied — название используемого набора данных с векторами для сравнения, tweets_applied — название файла с набором твитов с векторами для сравнения, input_dir — директория где находится используемый датасет и набор твитов; output_dir — директория в которую будет сохранён файл с рекомендациями; Имя порождённого файла, который содержит рекомендации, печается на экран. Пример использования:

```
$ twnews build_recommendation -h
usage: twnews build_recommendation [-h] --dataset_applied DATASET_APPLIED
                                   --tweets_applied TWEETS_APPLIED
                                   [--input_dir INPUT_DIR]
                                   [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --dataset_applied DATASET_APPLIED
                        dataset_applied name (default: None)
  --tweets_applied TWEETS_APPLIED
                        tweets_applied name (default: None)
  --input_dir INPUT_DIR
                        input directory (default: /home/avybornov/tmp)
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)
```

```
$ twnews build_recommendation --dataset_applied dataset_auto_applied
```

Точка входа recommendation позволяет вывести на экран (параметр eval) или в текстовый файл (параметр print) построенные ранее рекомендациям или снять с них метрики (параметр eval), дополнительно параметризуется параметризуется тремя параметрами recomended — название файла с рекомендациями, input_dir — директория где находятся файл с рекомендациями; output_dir — директория в которую будет сохранён текстовый файл с рекомендациями; Имя порождённого файла, который рекомендации в человекочитаемом формате, печается на экран. Пример использования:

```
$ twnews recommendation -h
```

```
usage: twnews recommendation [-h] (--eval | --dump | --print) --recommended
                             RECOMMENDED [--input_dir INPUT_DIR]
                             [--output_dir OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --eval                eval recomendation results (default: False)
  --dump                dump recomendation result to file (default: False)
  --print               print recomendation result to stdout (default: False)
  --recommended RECOMMENDED
                        tweets_applied name (default: None)
  --input_dir INPUT_DIR
                        input directory (default: /home/avybornov/tmp)
  --output_dir OUTPUT_DIR
                        output directory (default: /home/avybornov/tmp)
```

```
$ twnews recommended --eval --recommended dataset_auto_applied
```

Приложение записывает вспомогательную информацию о своём статусе и обработанных ошибках в файл лога. Поэтому из файла лога можно узнать актуальную информацию о работе приложения. Пример:

```
$ tail -f /var/log/twnews.log
INFO:root:2016-04-08 10:20:08.256411: News successfully loaded
INFO:root:2016-04-08 10:20:23.006948: Function iteration started with time
    measure
INFO:root:2016-04-08 10:33:32.930520: Function iteration finished in 13m9
    .9234058857s
INFO:root:2016-04-08 10:33:32.940666: Function find_topk_sim_news_to_tweets
    started with time measure
INFO:root:2016-04-08 10:34:42.360326: Function find_topk_sim_news_to_tweets
    finished in 1m9.41950583458s
INFO:root:2016-04-08 10:34:42.587674: Function iteration started with time
    measure
INFO:root:2016-04-08 10:48:04.453983: Function iteration finished in 13m21
    .8661620617s
INFO:root:2016-04-08 10:48:04.466846: Function find_topk_sim_news_to_tweets
    started with time measure
INFO:root:2016-04-08 10:49:18.958096: Function find_topk_sim_news_to_tweets
    finished in 1m14.4910538197s
```

```
INFO:root:2016-04-08 10:49:19.171160: Function iteration started with time
    measure
```

# 6. Эксперименты

Главное целью проведения экспериментов является сравнение двух реализованных методов автоматического установления связей между твитами и новостными статьями: метод основанный на частотности употребления слов и WTMF-G. Для исследования влияния на качество добавления информации о взаимосвязях вида текст-текст также производится сравнительное тестирование методов WTMF и WTMF-G.

Ввиду малого числа твитов в наборах данных тестирование производится на тех же выборках, на которых производится обучение. Также, ввиду того, что алгоритмы WTMF и WTMF-G используют случайным образом инициализированные матрицы, каждый запуск этих алгоритмов производился трёхкратно, в результатах приведено усреднённое значение.

## 6.1. Методы оценки качества

Для оценки качества рассматриваются метрики применимые для решения задач информационного поиска. Твит рассматривается как запрос, а список новостей как ответ. Для каждого твита, получаемый список новостей ранжирован по мере убывания их схожести. В работе использованы две метрики: $MRR$ и $TOP_I$, их описание дано ниже.

### 6.1.1. Метрика качества $MRR$

$MRR$ (от англ. Mean reciprocal rank) — статистическая метрика, используемая для измерения качества алгоритмов информационного поиска. Пусть $rank_i$ — позиция первого правильного ответа в $i$-м запросе, $n$ — общее количество запросов. Тогда значение $MRR$ можно получить по формуле:

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{rank_i}.$$

### 6.1.2. Метрика качества $TOP_I$

$TOP_I$ — группа метрик, используемых для оценки качества алгоритмов информационного поиска. Значение метрики $TOP_I$ численно равно проценту запросов с правильным ответом, входящим в первые $I$ ответов. Пусть $n$ — общее количество запросов, $Q_I(i)$ — равно 1, если правильный ответ на $i$-й запрос входит в первые $I$ предложенных ответов, 0 — в противном случае. Тогда значение $TOP_I$ можно получить по формуле:

$$TOP_I = \frac{1}{n}\sum_{i=1}^{n}Q_I(i).$$

В дальшейшем будут рассматриваться следующие три метрики из группы метрик $TOP_I$: $TOP_1$, $TOP_3$.

## 6.2. Оптимизация качества WTMF, путём варьирования параметров

Оптимизация параметров модели для метода WTMF будет производится на наборе данных cutted, используя метрику MRR. Модель WTMF зависит от четырёх параметров: $K$, $I$, $\lambda$, $w_m$. Параметры $K$ и $I$ влияют на время построения модели, а параметры $\lambda$ и $w_m$ не влияют на время построения модели.

В качестве начального приближения берутся значения параметров, которое использовали авторы работы [1], а именно: $K = 30$, $I = 3$, $\lambda = 20$, $w_m = 0.1$.

Оптимизируются параметры, не влияющие на время работы алгоритма: $\lambda$ и $w_m$. Для этого фиксируются остальные параметры: $I = 1$, $K = 30$. Для начала находится оптимальный порядок значений начального приближения. Результаты занесены в таблицу 11.

Таблица 11: Качество работы алгоритма WMTF для различных значений $\lambda$ и $w_m$ при фиксированных значениях $I = 1$, $K = 30$.

| $\lambda \backslash w_m$ | **0.001** | **0.01** | **0.1** | **1** | **10** | **100** |
|---|---|---|---|---|---|---|
| **0.2** | 0.6855 | 0.6877 | 0.7482 | 0.3651 | 0.1526 | 0.1485 |
| **2** | 0.7000 | 0.7015 | 0.7173 | **0.7525** | 0.3707 | 0.1605 |
| **20** | 0.6964 | 0.7081 | 0.7149 | 0.7308 | 0.7507 | 0.3784 |
| **200** | 0.7075 | 0.6991 | 0.7010 | 0.7016 | 0.7146 | 0.7448 |
| **2000** | 0.6970 | 0.7070 | 0.6991 | 0.7114 | 0.6994 | 0.7044 |

Как видно из таблицы 11 в целом получена достаточно однородная картина для всех порядков $\lambda$ и $w_m$. Заметное снижение качества происходит при большом порядке $w_m$ и малом порядке $\lambda$. Максимальное значение метрики достигнуто при $\lambda = 2$ и $w_m = 1$. Для уточнения значения коэффициентов, производится исследование качества работы алгоритма в окрестностях максимального значения метрики. Результаты приведены в таблице 12.

Таблица 12: Качество работы алгоритма WMTF для различных значений $\lambda$ и $w_m$ при фиксированных значениях $I = 1$, $K = 30$.

| $\lambda \backslash w_m$ | **0.9** | **0.95** | **1** | **1.1** | **1.2** |
|---|---|---|---|---|---|
| **1.9** | 0.7442 | 0.7451 | 0.7536 | 0.7542 | 0.7544 |
| **1.95** | 0.7447 | **0.7554** | 0.7452 | 0.7439 | 0.7504 |
| **2** | 0.7507 | 0.7528 | 0.7504 | 0.7515 | 0.7566 |
| **2.05** | 0.7413 | 0.7505 | 0.7424 | 0.7525 | 0.7479 |
| **2.1** | 0.7405 | 0.7484 | 0.7485 | 0.7502 | 0.7501 |

Из таблицы 12 получаем оптимальные значения коэффициентов $\lambda = 0.95$ и $w_m = 1.95$.

Оптимизируются параметры, влияющие на время работы алгоритма: $K$ и $I$. Для этого фиксируются остальные параметры: $\lambda = 0.95$, $w_m = 1.95$. Для начала находится примерное значение коэффициента $K$ и оптимальное значение $I$. Результаты занесены в таблицу 13.

Таблица 13: Качество работы алгоритма WMTF для различных значений $K$ и $I$ при фиксированных значениях $\lambda = 0.95$, $w_m = 1.95$.

| $K \backslash I$ | **1** | **2** | **3** |
|---|---|---|---|
| **5** | 0.1232 | 0.1593 | 0.1838 |
| **10** | 0.3521 | 0.4102 | 0.4437 |
| **30** | 0.7426 | 0.7422 | 0.7158 |
| **60** | **0.8326** | 0.8117 | 0.7620 |

Как видно из таблицы 13 увеличение $K$ приводит к значительному улучшению качества работы алгоритма, увеличении $I$ приводит к улучшению качества алгоритма только при малых значениях параметра $K$, при больших значениях $K$ увеличение параметра $I$ приводит к ухудшению качества. Максимальное значение метрики достигнуто при $K = 60$ и $I = 1$. Для уточнения значения коэффициента $K$, производится исследование качества работы алгоритма при фиксированном значении коэффициента $I$. Результаты приведены в таблице 14.

Таблица 14: Качество работы алгоритма WMTF для различных значений $K$ при фиксированных значениях $I = 1$, $\lambda = 0.95$, $w_m = 1.95$.

| **K** | **Значение метрики MRR** |
|---|---|
| **10** | 0.3595 |
| **20** | 0.6460 |
| **30** | 0.7496 |
| **40** | 0.8003 |
| **50** | 0.8220 |
| **60** | 0.8424 |
| **70** | 0.8472 |
| **80** | 0.8535 |
| **82** | 0.8549 |
| **84** | 0.8597 |
| **86** | 0.8592 |
| **88** | 0.8572 |
| **90** | **0.8675** |
| **92** | 0.8580 |
| **94** | 0.8604 |
| **96** | 0.8612 |
| **98** | 0.8644 |
| **100** | 0.8655 |
| **110** | 0.8627 |

Из таблицы 14 получаем оптимальное значение коэффициента $K = 90$. протестировать для большей размерности 150-200

В итоге оптимизации качества рекомендаций на основе алгоритма WMTF были получены оптимальные параметры: $K = 90$, $I = 1$, $\lambda = 0.95$, $w_m = 1.95$.

## 6.3. Оптимизация качества WTMF-G, путём варьирования параметров

Оптимизация параметров модели для метода WTMF-G будет производится на наборе данных cutted, используя метрику MRR. Модель WTMF-G зависит от пяти параметров: $K$, $I$, $\lambda$, $\delta$, $w_m$. Параметры $K$ и $I$ влияют на время построения модели, а параметры $\lambda$ и $w_m$ не влияют на время построения модели.

В качестве начального приближения параметров взяты оптимальные параметры для метода WTMF, а именно $K = 90$, $I = 1$, $w_m = 1.95$,$\lambda = 0.95$. В качестве начального приближения параметра $\delta$ берем значение 0.1.

Сначала оптимизируем параметры, влияющие на регуляризующий член: $\lambda$ и $\delta$, Для этого фиксируем остальные параметры: $I = 1$, $K = 90$, $w_m = 1.95$. Сначала найдём оптимальный порядок значений начального приближения. Результаты занесены в таблицу 15.

Таблица 15: Качество работы алгоритма WMTF-G для различных значений $\lambda$ и $\delta$ при фиксированных значениях $I = 1$, $K = 90$, $w_m = 1.95$.

| $\lambda\backslash\delta$ | **0.001** | **0.01** | **0.1** | **1** | **10** |
|---|---|---|---|---|---|
| **0.01** | 0.3889 | 0.3842 | 0.3924 | 0.3900 | 0.3895 |
| **0.1** | 0.4895 | 0.4875 | 0.4886 | 0.4850 | 0.4847 |
| **1** | 0.8227 | 0.8256 | 0.8242 | 0.8225 | 0.8212 |
| **10** | 0.8477 | 0.8440 | **0.8496** | 0.8454 | 0.8495 |
| **100** | 0.8294 | 0.8318 | 0.8283 | 0.8240 | 0.8243 |

Как видно из таблицы 15 порядок параметра $\delta$ оказывает влияние на качество, но достаточно слабое, порядок параметра $\lambda$, напротив очень сильно влияет на получаемое качество. Максимальное значение метрики достигнуто при $\lambda = 10$ и $\delta = 0.1$. Для уточнения значения коэффициентов, производится исследование качества работы алгоритма в окрестностях максимального значения метрики. Результаты приведены в таблице 16.

В таблице 16 получена достаточно однородная картина. Возьмём в качестве оптимального значения коэффициент полученную точку максимум: $\lambda = 6$, $\delta = 0.06$.

Рассмотрим влияние параметра $w_m$ и найдём его оптимальное значение. Сначала рассмотрим качество алгоритма для различных порядков $w_m$. Результаты занесены в таблицу 17

Как видно из таблицы 17 порядок параметра $w_m$ оказывает заметное влияние на качество. При значительном увеличении до $10^2$ качество начинает резко падать. Максимальное

Таблица 16: Качество работы алгоритма WMTF-G для различных значений $\lambda$ и $\delta$ при фиксированных значениях $I = 1$, $K = 90$, $w_m = 1.95$.

| $\lambda\backslash\delta$ | **0.06** | **0.08** | **0.1** | **0.12** | **0.14** |
|---|---|---|---|---|---|
| **4** | 0.8501 | 0.8512 | 0.8489 | 0.8530 | 0.8476 |
| **6** | **0.8589** | 0.8524 | 0.8511 | 0.8580 | 0.8493 |
| **8** | 0.8483 | 0.8528 | 0.8539 | 0.8439 | 0.8498 |
| **10** | 0.8504 | 0.8455 | 0.8416 | 0.8453 | 0.8408 |
| **12** | 0.8453 | 0.8398 | 0.8472 | 0.8376 | 0.8415 |
| **14** | 0.8462 | 0.8456 | 0.8387 | 0.8398 | 0.8377 |

Таблица 17: Качество работы алгоритма WMTF-G для различных значений $w_m$ при фиксированных значениях $I = 1$, $K = 90$, $\lambda = 6$, $\delta = 0.6$.

| $w_m$ | 0.01 | 0.05 | 0.1 | 0.5 | 1 | 5 | 10 | 50 | 100 |
|---|---|---|---|---|---|---|---|---|---|
| **MRR** | 0.8283 | 0.8296 | 0.8285 | 0.8359 | 0.8442 | **0.8639** | 0.8391 | 0.6094 | 0.5035 |

значение метрики достигнуто при $w_m = 5$, уточним полученное значение. Для уточнения значения коэффициента $w_m$, производится исследование качества работы алгоритма в окрестностях максимального значения метрики. Результаты приведены в таблице 18.

Таблица 18: Качество работы алгоритма WMTF-G для различных значений $w_m$ при фиксированных значениях $I = 1$, $K = 90$, $\lambda = 6$, $\delta = 0.6$.

| $w_m$ | 3.0 | 3.5 | 4.0 | 4.5 | 5.0 | 5.5 | 6.0 | 6.5 | 7.5 |
|---|---|---|---|---|---|---|---|---|---|
| **MRR** | 0.8592 | 0.8585 | 0.8594 | 0.8603 | **0.8639** | 0.8591 | 0.8586 | 0.8574 | 0.8536 |

Из таблицы 18 получаем оптимальное значение параметра $w_m = 5$. Теперь рассмотрим оставшиеся два параметра $K$ и $I$. Качество работы алгоритма WTMF-G для различных значений $K$ и $I$ приведено в таблице 19.

Как показано в таблице 19, WTMF-G показывает аналогично методу WTMF поведение при изменении параметров $K$ и $I$, а именно, в среднем с увеличением количества итераций, качество работы алгоритма уменьшается. Максимум был достигнут при $K = 220$, $I = 1$.

В итоге оптимизации качества рекомендаций на основе алгоритма WMTF-G были получены оптимальные параметры: $K = 220$, $I = 1$, $\delta = 0.6$, $\lambda = 6$, $w_m = 5$.

Таблица 19: Качество работы алгоритма WMTF-G для различных значений $K$ и $I$ при фиксированных значениях $w\_m = 5$, $\lambda = 6$, $\delta = 0.06$.

| $K \backslash I$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 30 | 0.7529 | 0.7927 | 0.7577 | 0.6736 | 0.5794 |
| 40 | 0.7992 | 0.8194 | 0.7695 | 0.6813 | 0.5828 |
| 50 | 0.8269 | 0.8349 | 0.7834 | 0.6830 | 0.5801 |
| 60 | 0.8419 | 0.8450 | 0.7984 | 0.7056 | 0.6006 |
| 70 | 0.8557 | 0.8466 | 0.7977 | 0.7036 | 0.6002 |
| 80 | 0.8614 | 0.8511 | 0.7990 | 0.7032 | 0.5957 |
| 90 | 0.8606 | 0.8522 | 0.8039 | 0.7088 | 0.6038 |
| 100 | 0.8606 | 0.8527 | 0.8022 | 0.7089 | 0.6021 |
| 110 | 0.8686 | 0.8553 | 0.8074 | 0.7123 | 0.6065 |
| 120 | 0.8693 | 0.8579 | 0.8097 | 0.7174 | 0.6085 |
| 130 | 0.8725 | 0.8588 | 0.8160 | 0.7264 | 0.6206 |
| 140 | 0.8740 | 0.8597 | 0.8157 | 0.7248 | 0.6241 |
| 150 | 0.8763 | 0.8620 | 0.8171 | 0.7263 | 0.6200 |
| 160 | 0.8740 | 0.8596 | - | - | - |
| 170 | 0.8768 | 0.8606 | - | - | - |
| 180 | 0.8785 | 0.8613 | - | - | - |
| 190 | 0.8767 | 0.8616 | - | - | - |
| 200 | 0.8769 | 0.8613 | - | - | - |
| 210 | 0.8786 | 0.8613 | - | - | - |
| 220 | **0.8816** | 0.8632 | - | - | - |
| 230 | 0.8814 | 0.8646 | - | - | - |
| 240 | 0.8758 | 0.8632 | - | - | - |

## 6.4. Сравнительные результаты

Для выявления влияния добавления связей текст-текст на результаты работы метода WMTF-G производится сравнительное тестирование алгоритма WTMF и WTMF-G. Тестирование производится для различных наборов данных. Результаты тестирования приведены в таблице 20.

Таблица 20: Сравнительное тестирование алгоритмов WTMF и WTMF-G.

| **Набор данных** | **Метрика MRR** | | **Метрика** $TOP_1$ | | **Метрика** $TOP_3$ | |
|---|---|---|---|---|---|---|
| | **WTMF** | **WTMF-G** | **WTMF** | **WTMF-G** | **WTMF** | **WTMF-G** |
| manual | 0.7293 | 0.7854 | 0.6187 | 0.6756 | 0.8193 | 0.8775 |
| auto | 0.8640 | 0.8685 | 0.8096 | 0.8149 | 0.9148 | 0.9195 |
| total | 0.8196 | 0.8487 | 0.7452 | 0.7828 | 0.8851 | 0.9049 |
| cutted | 0.8630 | 0.8816 | 0.8045 | 0.8204 | 0.9118 | 0.9279 |
| manual_nt | 0.6194 | 0.7000 | 0.4733 | 0.5502 | 0.7223 | 0.8217 |
| auto_nt | 0.5297 | 0.5695 | 0.4436 | 0.4798 | 0.5750 | 0.6099 |
| total_nt | 0.5729 | 0.6341 | 0.4587 | 0.5143 | 0.6448 | 0.7296 |
| cutted_nt | 0.6495 | 0.7039 | 0.5371 | 0.5974 | 0.7372 | 0.7840 |

Как видно из таблицы 20 алгоритм WMTF-G показывает стабильно более высокий результат чем алгоритм WMTF, из этого можно сделать вывод, что добавление связей текст-текст позволяет построить более точные рекомендации.

Сравним два метода рекомендаций: TF-IDF и WTMF-G. Сначала посмотрим на результаты полученные на базовых эталонных наборах данных, то есть тех, которые наряду с нетривиальными содержат большое количество тривиальных связей. Результаты тестирования приведены в таблице 21.

Таблица 21: Сравнительное тестирование алгоритмов TF-IDF и WTMF-G на базовых эталонных наборах данных.

| **Набор данных** | **Метрика MRR** | | **Метрика** $TOP_1$ | | **Метрика** $TOP_3$ | |
|---|---|---|---|---|---|---|
| | **TF-IDF** | **WTMF-G** | **TF-IDF** | **WTMF-G** | **TF-IDF** | **WTMF-G** |
| manual | 0.8310 | 0.7854 | 0.7337 | 0.6756 | 0.9137 | 0.8775 |
| auto | 0.8817 | 0.8685 | 0.8344 | 0.8149 | 0.9299 | 0.9195 |
| total | 0.8610 | 0.8487 | 0.8044 | 0.7828 | 0.9249 | 0.9049 |
| cutted | 0.9075 | 0.8816 | 0.8499 | 0.8204 | 0.9453 | 0.9279 |

Как видно из таблицы 21 метод TF-IDF показывает заметно более лучший результат на всех наборах данных. В целом для метода TF-IDF получены неожиданно высокие результаты, качество полученное для метода TF-IDF авторами метода WTMF-G при связывании твитов и новостей почти в два раза меньше (качество полученное авторами метода WTMF-G приведено в разделе **??**). Настолько высокие результаты метода TF-IDF получены по следующим причи-

нам: во-первых, в русском твиттере очень много тривиальных связей твит-новость, во-вторых, ввиду специфики русского сегмента твиттер, в заголовках новостей и твитах их описывающих оказалось большое количество общих слов.

С целью нивелирования влияния тривиальных связей было проведено тестирование на наборах данных, которые содержат исключительно нетривиальные связи. Результаты экспериментов приведены в таблице 22.

Таблица 22: Сравнительное тестирование алгоритмов TF-IDF и WTMF-G на наборах данных с нетривиальными твитами.

| **Набор данных** | **Метрика MRR** | | **Метрика** $TOP_1$ | | **Метрика** $TOP_3$ | |
|---|---|---|---|---|---|---|
| | **TF-IDF** | **WTMF-G** | **TF-IDF** | **WTMF-G** | **TF-IDF** | **WTMF-G** |
| manual_nt | 0.7565 | 0.7000 | 0.6250 | 0.5502 | 0.8688 | 0.8217 |
| auto_nt | 0.6035 | 0.5695 | 0.5254 | 0.4798 | 0.6461 | 0.6099 |
| total_nt | 0.6914 | 0.6341 | 0.5833 | 0.5143 | 0.8688 | 0.7296 |
| cutted_nt | 0.7485 | 0.7039 | 0.6442 | 0.5974 | 0.8303 | 0.7840 |

Как видно из таблицы 22 на наборах данных с нетривиальными твитами наблюдается такой же результат, как и для наборов данных с полным набором твитов. То есть метод TF-IDF во всех рассматриваемых случаях показывает более высокое качество, чем метод WTMF-G. Из этого следует, что рассматриваемые нетривиальные твиты, оказались «большими» текстами, очень похожими на заголовки новостей — это вызвано влиянием специфики русского сегмента твиттера.

Также из таблиц 21 и 22 можно оценить влияние различных наборов данных. Автоматически собранный датасет для полного набора твитов даёт более лучшее качество, чем в ручную собранный, это вызвано влиянием большого числа тривиальных связей. Как и ожидается результаты на наборах данных с нетривиальными твитами показывают обратную картину.

На наборе данных cutted для полного набора данных разница качества между TF-IDF и WTMF-G наименьшая - это вызвано тем, что оптимизация метода WTMF-G производилась на этом наборе данных.

# 7. Технико-экономическое обоснование

Разработка программного обеспечения — достаточно трудоемкий и длительный процесс, требующий выполнения большого числа разнообразных операций. Организация и планирование процесса разработки программного продукта или программного комплекса при традиционном методе планирования предусматривает выполнение следующих работ [9]:

- формирование состава выполняемых работ и группировка их по стадиям разработки;
- расчет трудоемкости выполнения работ;
- установление профессионального состава и расчет количества исполнителей;
- определение продолжительности выполнения отдельных этапов разработки;
- построение календарного графика выполнения разработки;
- контроль выполнения календарного графика.

Далее приведен перечень и состав работ при разработке программного средства для автоматического установления связей между сообщениями твиттера и новостными статьями. Отметим, что процесс разработки программного продукта характеризуется совместной работой разработчиков постановки задач и разработчиков программного обеспечения.

Укрупненный состав работ по стадиям разработки программного продукта [10] [8]:

1. Техническое задание:
    - Постановка задач, выбор критериев эффективности,
    - Разработка технико-экономического обоснования разработки,
    - Определение состава пакета прикладных программ, состава и структуры информационной базы,
    - Выбор языков программирования,
    - Предварительный выбор методов выполнения работы,
    - Разработка календарного плана выполнения работ;
2. Эскизный проект:
    - Предварительная разработка структуры входных и выходных данных,
    - Разработка общего описания алгоритмов реализации решения задач,
    - Разработка пояснительной записки,
    - Консультации разработчиков постановки задач,

- Согласование и утверждение эскизного проекта;

3. Технический проект:

   - Разработка алгоритмов решения задач,
   - Разработка пояснительной записки,
   - Согласование и утверждение технического проекта,
   - Разработка структуры программы,
   - Разработка программной документации и передача ее для включения в технический проект,
   - Уточнение структуры, анализ и определение формы представления входных и выходных данных,
   - Выбор конфигурации технических средств;

4. Рабочий проект:

   - Комплексная отладка задач и сдача в опытную эксплуатацию,
   - Разработка проектной документации,
   - Программирование и отладка программ,
   - Описание контрольного примера,
   - Разработка программной документации,
   - Разработка, согласование программы и методики испытаний,
   - Предварительное проведение всех видов испытаний;

5. Внедрение:

   - Подготовка и передача программной документации для сопровождения с оформлением соответствующего Акта,
   - Передача программной продукции в фонд алгоритмов и программ,
   - Проверка алгоритмов и программ решения задач, корректировка документации после опытной эксплуатации программного продукта;

Трудоемкость разработки программной продукции зависит от ряда факторов, основными из которых являются следующие: степень новизны разрабатываемого программного комплекса, сложность алгоритма его функционирования, объем используемой информации, вид ее представления и способ обработки, а также уровень используемого алгоритмического языка программирования. Чем выше уровень языка, тем трудоемкость меньше.

По степени новизны разрабатываемый проект относится к *группе новизны A* – разработка программных комплексов, требующих использования принципиально новых методов их создания, проведения НИР и т.п.

По степени сложности алгоритма функционирования проект относится к *2 группе сложности* - программная продукция, реализующая учетно-статистические алгоритмы.

По виду представления исходной информации и способа ее контроля программный продукт относится к *группе 12* - исходная информация представлена в форме документов, имеющих различный формат и структуру и *группе 22* - требуется печать документов одинаковой формы и содержания, вывод массивов данных на машинные носители.

## 7.1. Трудоемкость разработки программной продукции

Трудоемкость разработки программной продукции ($\tau_{PP}$) может быть определена как сумма величин трудоемкости выполнения отдельных стадий разработки программного продукта из выражения:

$$\tau_{PP} = \tau_{TZ} + \tau_{EP} + \tau_{TP} + \tau_{RP} + \tau_V,$$

где $\tau_{TZ}$ — трудоемкость разработки технического задания на создание программного продукта; $\tau_{EP}$ — трудоемкость разработки эскизного проекта программного продукта; $\tau_{TP}$ — трудоемкость разработки технического проекта программного продукта; $\tau_{RP}$ — трудоемкость разработки рабочего проекта программного продукта; $\tau_V$ — трудоемкость внедрения разработанного программного продукта.

### 7.1.1. Трудоемкость разработки технического задания

Расчёт трудоёмкости разработки технического задания ($\tau_{PP}$) [чел.-дни] производится по формуле:

$$\tau_{TZ} = T^Z_{RZ} + T^Z_{RP},$$

где $T^Z_{RZ}$ — затраты времени разработчика постановки задачи на разработку ТЗ, [чел.-дни]; $T^Z_{RP}$ — затраты времени разработчика программного обеспечения на разработку ТЗ, [чел.-дни]. Их значения рассчитываются по формулам:

$$T^Z_{RZ} = t_Z \cdot K^Z_{RZ},$$

$$T^Z_{RP} = t_Z \cdot K^Z_{RP},$$

где $t_Z$ – норма времени на разработку ТЗ на программный продукт (зависит от функционального назначения и степени новизны разрабатываемого программного продукта), [чел.-дни]. В нашем случае по таблице получаем значение (группа новизны – А, функциональное назначение

– технико-экономическое планирование):

$$t_Z = 79.$$

$K^Z_{RZ}$ — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ТЗ. В нашем случае (совместная разработка с разработчиком ПО):

$$K^Z_{RZ} = 0.65.$$

$K^Z_{RP}$ — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ. В нашем случае (совместная разработка с разработчиком постановки задач):

$$K^Z_{RP} = 0.35.$$

Тогда:

$$\tau_{TZ} = 79 \cdot (0.35 + 0.65) = 79.$$

### 7.1.2. Трудоемкость разработки эскизного проекта

Расчёт трудоёмкости разработки эскизного проекта ($\tau_{EP}$) [чел.-дни] производится по формуле:

$$\tau_{EP} = T^E_{RZ} + T^E_{RP},$$

где $T^E_{RZ}$ — затраты времени разработчика постановки задачи на разработку эскизного проекта (ЭП), [чел.-дни]; $T^E_{RP}$ — затраты времени разработчика программного обеспечения на разработку ЭП, [чел.-дни]. Их значения рассчитываются по формулам:

$$T^E_{RZ} = t_E \cdot K^E_{RZ},$$

$$T^E_{RP} = t_E \cdot K^E_{RP},$$

где $t_E$ – норма времени на разработку ЭП на программный продукт (зависит от функционального назначения и степени новизны разрабатываемого программного продукта), [чел.-дни]. В нашем случае по таблице получаем значение (группа новизны – А, функциональное назначение – технико-экономическое планирование):

$$t_E = 175.$$

$K^E_{RZ}$ — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ЭП. В нашем случае (совместная разработка с разработ-

чиком ПО):

$$K_{RZ}^{E} = 0.7.$$

$K_{RP}^{E}$ — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ. В нашем случае (совместная разработка с разработчиком постановки задач):

$$K_{RP}^{E} = 0.3.$$

Тогда:

$$\tau_{EP} = 175 \cdot (0.3 + 0.7) = 175.$$

### 7.1.3. Трудоемкость разработки технического проекта

Трудоёмкость разработки технического проекта ($\tau_{TP}$) [чел.-дни] зависит от функционального назначения программного продукта, количества разновидностей форм входной и выходной информации и определяется по формуле:

$$\tau_{TP} = (t_{RZ}^{T} + t_{RP}^{T}) \cdot K_V \cdot K_R,$$

где $t_{RZ}^{T}$ — норма времени, затрачиваемого на разработку технического проекта (ТП) разработчиком постановки задач, [чел.-дни]; $t_{RP}^{T}$ — норма времени, затрачиваемого на разработку ТП разработчиком ПО, [чел.-дни]. По таблице принимаем (функциональное назначение — технико-экономическое планирование, количество разновидностей форм входной информации — 2 (твиты, новости), количество разновидностей форм выходной информации — 2 (набор связей твит-новости, оценка работы рекомендательной системы)):

$$t_{RZ}^{T} = 52,$$

$$t_{RP}^{T} = 14.$$

$K_R$ — коэффициент учета режима обработки информации. По таблице принимаем (группа новизны — А, режим обработки информации — реальный масштаб времени):

$$K_R = 1.67.$$

$K_V$ — коэффициент учета вида используемой информации, определяется по формуле:

$$K_V = \frac{K_P \cdot n_P + K_{NS} \cdot n_{NS} + K_B \cdot n_B}{n_P + n_{NS} + n_B},$$

где $K_P$ — коэффициент учета вида используемой информации для переменной информации; $K_{NS}$ — коэффициент учета вида используемой информации для нормативно-справочной инфор-

мации; $K_B$ — коэффициент учета вида используемой информации для баз данных; $n_P$ — количество наборов данных переменной информации; $n_{NS}$ — количество наборов данных нормативно-справочной информации; $n_B$ — количество баз данных. Коэффициенты находим по таблице (группа новизны - А):

$$K_P = 1.70,$$

$$K_{NS} = 1.45,$$

$$K_B = 4.37.$$

Количество наборов данных, используемых в рамках задачи:

$$n_P = 3,$$

$$n_{NS} = 0,$$

$$n_B = 1.$$

Находим значение $K_V$:

$$K_V = \frac{1.70 \cdot 3 + 1.45 \cdot 0 + 4.37 \cdot 1}{3 + 0 + 1} = 2.3675.$$

Тогда:

$$\tau_{TP} = (52 + 14) \cdot 2.3675 \cdot 1.67 = 261.$$

### 7.1.4. Трудоемкость разработки рабочего проекта

Трудоёмкость разработки рабочего проекта ($\tau_{RP}$) [чел.-дни] зависит от функционального назначения программного продукта, количества разновидностей форм входной и выходной информации, сложности алгоритма функционирования, сложности контроля информации, степени использования готовых программных модулей, уровня алгоритмического языка программирования и определяется по формуле:

$$\tau_{RP} = (t^R_{RZ} + t^R_{RP}) \cdot K_K \cdot K_R \cdot K_Y \cdot K_Z \cdot K_{IA},$$

где $t^R_{RZ}$ — норма времени, затраченного на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком постановки задач, [чел.-дни]. $t^R_{RP}$ — норма времени, затраченного на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком ПО, [чел.-дни]. По таблице принимаем (функциональное назначение — технико-экономическое планирование, количество разновидностей форм входной информации — 2 (твиты, новости), количество разновидностей форм выходной информации — 2 (набор связей твит-

новости, оценка работы рекомендательной системы)):

$$t_{RZ}^{R} = 15,$$

$$t_{RP}^{R} = 91.$$

$K_K$ — коэффициент учета сложности контроля информации. По таблице принимаем (степень сложности контроля входной информации — 12, степень сложности контроля выходной информации — 22):

$$K_K = 1.00.$$

$K_R$ — коэффициент учета режима обработки информации. По таблице принимаем (группа новизны — А, режим обработки информации — реальный масштаб времени):

$$K_R = 1.75.$$

$K_Y$ — коэффициент учета уровня используемого алгоритмического языка программирования. По таблице принимаем значение (интерпретаторы, языковые описатели):

$$K_Y = 0.8.$$

$K_Z$ — коэффициент учета степени использования готовых программных модулей. По таблице принимаем (использование готовых программных модулей составляет около 30

$$K_Z = 0.7.$$

$K_{IA}$ — коэффициент учета вида используемой информации и сложности алгоритма программного продукта, его значение определяется по формуле:

$$K_I A = \frac{K'_P \cdot n_P + K'_{NS} \cdot n_{NS} + K'_B \cdot n_B}{n_P + n_{NS} + n_B},$$

где $K'_P$ — коэффициент учета сложности алгоритма ПП и вида используемой информации для переменной информации; $K'_{NS}$ — коэффициент учета сложности алгоритма ПП и вида используемой информации для нормативно-справочной информации; $K'_B$ — коэффициент учета сложности алгоритма ПП и вида используемой информации для баз данных. $n_P$ — количество наборов данных переменной информации; $n_{NS}$ — количество наборов данных нормативно-справочной информации; $n_B$ — количество баз данных. Коэффициенты находим по таблице (группа новизны - А):

$$K'_P = 2.02,$$

$$K'_{NS} = 1.21,$$

$$K'_B = 1.05.$$

Количество наборов данных, используемых в рамках задачи:

$$n_P = 3,$$

$$n_{NS} = 0,$$

$$n_B = 1.$$

Находим значение $K_{IA}$:

$$K_{IA} = \frac{2.02 \cdot 3 + 1.21 \cdot 0 + 1.05 \cdot 1}{3 + 0 + 1} = 1.7775.$$

Тогда:

$$\tau_{RP} = (15 + 91) \cdot 1.00 \cdot 1.75 \cdot 0.8 \cdot 0.7 \cdot 1.7775 = 185.$$

### 7.1.5. Трудоемкость выполнения стадии «Внедрение»

Расчёт трудоёмкости разработки технического проекта ($\tau_V$) [чел.-дни] производится по формуле:

$$\tau_V = (t^V_{RZ} + t^V_{RP}) \cdot K_K \cdot K_R \cdot K_Z,$$

где $t^V_{RZ}$ — норма времени, затрачиваемого разработчиком постановки задач на выполнение процедур внедрения программного продукта, [чел.-дни]; $t^V_{RP}$ — норма времени, затрачиваемого разработчиком программного обеспечения на выполнение процедур внедрения программного продукта, [чел.-дни]. По таблице принимаем (функциональное назначение — технико-экономическое планирование, количество разновидностей форм входной информации — 2 (твиты, новости), количество разновидностей форм выходной информации — 2 (набор связей твит-новости, оценка работы рекомендательной системы)):

$$t^V_{RZ} = 17,$$

$$t^V_{RP} = 19.$$

Коэффициент $K_K$ и $K_Z$ были найдены выше:

$$K_K = 1.00,$$

$$K_Z = 0.7.$$

$K_R$ — коэффициент учета режима обработки информации. По таблице принимаем (группа новизны — А, режим обработки информации — реальный масштаб времени):

$$K_R = 1.60.$$

Тогда:

$$\tau_V = (17 + 19) \cdot 1.00 \cdot 1.60 \cdot 0.7 = 40.$$

Общая трудоёмкость разработки ПП:

$$\tau_{PP} = 79 + 175 + 261 + 185 + 40 = 740.$$

## 7.2. Расчет количества исполнителей

Средняя численность исполнителей при реализации проекта разработки и внедрения ПО определяется соотношением:

$$N = \frac{t}{F},$$

где $t$ — затраты труда на выполнение проекта (разработка и внедрение ПО); $F$ — фонд рабочего времени. Разработка велась 5 месяцев с 1 января 2016 по 31 мая 2016. Количество рабочих дней по месяцам приведено в таблице 23. Из таблицы получаем, что фонд рабочего времени

$$F = 96.$$

Таблица 23: Количество рабочих дней по месяцам

| **Номер месяца** | **Интервал дней** | **Количество рабочих дней** |
|---|---|---|
| 1 | 01.01.2016 - 31.01.2016 | 15 |
| 3 | 01.02.2016 - 29.02.2016 | 20 |
| 4 | 01.03.2016 - 31.03.2016 | 21 |
| 5 | 01.04.2016 - 30.04.2016 | 21 |
| 6 | 01.05.2016 - 31.05.2016 | 19 |
| Итого | | 96 |

Получаем число исполнителей проекта:

$$N = \frac{740}{96} = 8$$

Для реализации проекта потребуются 3 старших инженеров и 5 простых инженеров.

## 7.3. Ленточный график выполнения работ

На основе рассчитанных в главах 7.1, 7.2 трудоёмкости и фонда рабочего времени найдём количество рабочих дней, требуемых для выполнения каждого этапа разработка. Результаты приведены в таблице 24.

Таблица 24: Трудоёмкость выполнения работы над проектом

| **Номер стадии** | **Название стадии** | **Трудоёмкость [чел.-дни]** | **Удельный вес [%]** | **Количество рабочих дней** |
|---|---|---|---|---|
| 1 | Техническое задание | 79 | 11 | 10 |
| 2 | Эскизный проект | 175 | 24 | 23 |
| 3 | Технический проект | 261 | 35 | 34 |
| 4 | Рабочий проект | 185 | 25 | 24 |
| 5 | Внедрение | 40 | 5 | 5 |
| Итого | | 740 | 100 | 96 |

Планирование и контроль хода выполнения разработки проводится по ленточному графику выполнения работ. По данным в таблице 24 в ленточный график (таблица 25), в ячейки столбца “продолжительности рабочих дней” заносятся времена, которые требуются на выполнение соответствующего этапа. Все исполнители работают одновременно.

Таблица 25: Ленточный график выполнения работ

| Номер стадии | Продолжительность [раб.-дни] | Календарные дни | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 01.01.2016 - 03.01.2016 | 04.01.2016 - 10.01.2016 | 11.01.2016 - 17.01.2016 | 18.01.2016 - 24.01.2016 | 25.01.2016 - 31.01.2016 | 01.02.2016 - 07.02.2016 | 08.02.2016 - 14.02.2016 | 15.02.2016 - 21.02.2016 | 22.02.2016 - 28.02.2016 | 29.02.2016 - 06.03.2016 | 07.03.2016 - 13.03.2016 | 14.03.2016 - 20.03.2016 | 21.03.2016 - 27.03.2016 | 28.03.2016 - 03.04.2016 | 04.04.2016 - 10.04.2016 | 11.04.2016 - 17.04.2016 | 18.04.2016 - 24.04.2016 | 25.04.2016 - 01.05.2016 | 02.05.2016 - 08.05.2016 | 08.05.2016 - 15.05.2016 | 16.05.2016 - 22.05.2016 | 23.05.2016 - 29.05.2016 | 30.05.2016 - 31.05.2016 |
| | | Количество рабочих дней | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0 | 0 | 5 | 5 | 5 | 5 | 5 | 6 | 3 | 5 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 3 | 4 | 5 | 5 | 2 | |
| 1 | 10 | | | 5 | 5 | | | | | | | | | | | | | | | | | | | | |
| 2 | 23 | | | | | 5 | 5 | 5 | 6 | 2 | | | | | | | | | | | | | | | |
| 3 | 34 | | | | | | | | | 1 | 5 | 3 | 5 | 5 | 5 | 5 | 5 | | | | | | | | |
| 4 | 24 | | | | | | | | | | | | | | | | | 5 | 5 | 3 | 4 | 5 | 2 | | |
| 5 | 5 | | | | | | | | | | | | | | | | | | | | | | 3 | 2 | |

## 7.4. Определение себестоимости программной продукции

Затраты, образующие себестоимость продукции (работ, услуг), состоят из затрат на заработную плату исполнителям, затрат на закупку или аренду оборудования, затрат на организацию рабочих мест, и затрат на накладные расходы.

В таблице 26 приведены затраты на заработную плату и отчисления на социальное страхование в пенсионный фонд, фонд занятости и фонд обязательного медицинского страхования (30.5 %). Для старшего инженера предполагается оклад в размере 120000 рублей в месяц, для инженера предполагается оклад в размере 100000 рублей в месяц.

Таблица 26: Затраты на зарплату и отчисления на социальное страхование

| **Должность** | **Зарплата в месяц** | **Рабочих месяцев** | **Суммарная зарплата** | **Затраты на социальные нужды** |
|---|---|---|---|---|
| Старший инженер | 120000 | 5 | 600000 | 183000 |
| Старший инженер | 120000 | 5 | 600000 | 183000 |
| Старший инженер | 120000 | 5 | 600000 | 183000 |
| Инженер | 100000 | 5 | 500000 | 152500 |
| Инженер | 100000 | 5 | 500000 | 152500 |
| Инженер | 100000 | 5 | 500000 | 152500 |
| Инженер | 100000 | 5 | 500000 | 152500 |
| Инженер | 100000 | 5 | 500000 | 152500 |
| Суммарные затраты | | | 5611500 | |

Расходы на материалы, необходимые для разработки программной продукции, указаны в таблице 27.

Таблица 27: Затраты на материалы

| **Наименование материала** | **Единица измерения** | **Кол-во** | **Цена за единицу, руб.** | **Сумма, руб.** |
|---|---|---|---|---|
| Бумага А4 | Пачка 400 л. | 2 | 200 | 400 |
| Картридж для принтера HP P10025 | Шт. | 3 | 450 | 1350 |
| Суммарные затраты | | | | 1750 |

В работе над проектом используется специальное оборудование — персональные электронно-вычислительные машины (ПЭВМ) в количестве 8 шт. Стоимость одной ПЭВМ составляет 90000 рублей. Месячная норма амортизации К = 2,7%. Тогда за 4 месяцев работы расходы на амортизацию составят $P = 90000 \cdot 8 \cdot 0.027 \cdot 4 = 77760$ рублей.

Общие затраты на разработку программного продукта (ПП) составят

$$5611500 + 1750 + 77760 = 5691010 \text{ рублей.}$$

## 7.5. Определение стоимости программной продукции

Для определения стоимости работ необходимо на основании плановых сроков выполнения работ и численности исполнителей рассчитать общую сумму затрат на разработку программного продукта. Если ПП рассматривается и создается как продукция производственно-технического назначения, допускающая многократное тиражирование и отчуждение от непосредственных разработчиков, то ее цена $P$ определяется по формуле:

$$P = K \cdot C + Pr,$$

где $C$ — затраты на разработку ПП (сметная себестоимость); $K$ — коэффициент учёта затрат на изготовление опытного образца ПП как продукции производственно-технического назначения ($K = 1.1$); $Pr$ — нормативная прибыль, рассчитываемая по формуле:

$$Pr = \frac{C \cdot \rho_N}{100},$$

где $\rho_N$ — норматив рентабельности, $\rho_N = 30\%$;

Получаем стоимость программного продукта:

$$P = 1.1 \cdot 5691010 + 5691010 \cdot 0.3 = 7967414 \text{ рублей.}$$

## 7.6. Расчет экономической эффективности

Основными показателями экономической эффективности является чистый дисконтированный доход (NPV) и срок окупаемости вложенных средств. Чистый дисконтированный доход определяется по формуле:

$$NPV = \sum_{t=0}^{T} (R_t - Z_t) \cdot \frac{1}{(1+E)^t},$$

где $T$ — горизонт расчета по месяцам; $t$ — период расчета; $R_t$ — результат, достигнутый на $t$ шаге (стоимость); $Z_t$ — текущие затраты (на шаге $t$); $E$ — приемлемая для инвестора норма прибыли на вложенный капитал.

На момент начала 2016 года, ставка рефинансирования 11% годовых (ЦБ РФ), что эквивалентно 0.87% в месяц. В виду особенности разрабатываемого продукта он может быть продан лишь однократно. Отсюда получаем

$$E = 0.0087.$$

В таблице 28 находится расчёт чистого дисконтированного дохода. График его изменения приведён на рисунке 9.

Согласно проведенным расчетам, проект является рентабельным. Разрабатываемый проект позволит превысить показатели качества существующих систем и сможет их заменить.

Таблица 28: Расчёт чистого дисконтированного дохода

| **Месяц** | **Текущие затраты, руб.** | **Затраты с начала года, руб.** | **Текущий доход, руб.** | **ЧДД, руб.** |
|---|---|---|---|---|
| Январь | 1201810 | 1201810 | 0 | -1201810 |
| Февраль | 1122300 | 2324110 | 0 | -2314430 |
| Март | 1122300 | 3446410 | 0 | -3417454 |
| Апрель | 1122300 | 4568710 | 0 | -4510964 |
| Мая | 1122300 | 5700730 | 7967414 | 2101032 |

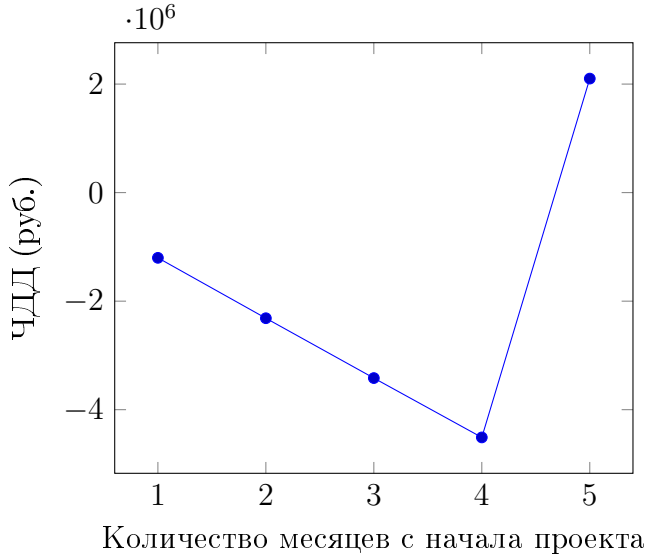


Рисунок 9 — График изменения чистого дисконтированного дохода

Итоговый ЧДД составил: 2101032 рублей.

## 7.7. Результаты

В рамках организационно-экономической части был спланирован календарный график проведения работ по созданию подсистемы поддержки проведения диагностики промышленных, а также были проведены расчеты по трудозатратам. Были исследованы и рассчитаны следующие статьи затрат: материальные затраты; заработная плата исполнителей; отчисления на социальное страхование; накладные расходы.

В результате расчетов было получено общее время выполнения проекта, которое составило 96 рабочих дней, получены данные по суммарным затратам на создание системы для автоматического сопоставления твитов и новостных статей, которые составили 5700730 рублей. Согласно проведенным расчетам, проект является рентабельным. Цена данного программного проекта составила 7967414 рублей, итоговый ЧДД составил 2101032 рублей.

# Заключение

В рамках дипломной работы было проведено исследование методов установления связей между твитами и новостными статьями. Было собрано несколько наборов данных и проанализированы различные подходы к их разметке. Реализован программный комплекс, позволяющий устанавить связи между твитами и новостными статьями в формате рекомендаций на основе различных подходов: методов WTMF, WTMF-G и метода, основанного на частотности слов (TF-IDF). На основе написанного программного комплекса произведено сравнение различных подходов. Как результат сравнения получено, что в для русского сегмента твиттера наиболее оптимальным подходом является метод, основанный на частотности слов (TF-IDF).

К сожалению, исследование показало, что наиболее простой метод, основанный на частотности слов, проявил наилучшее качество. Высокое качество метода TF-IDF является серьёзной проблемой, так как из этого следует, что рассматриваемые твиты, оказались «большими» текстами, очень похожими на заголовки новостей, то есть рассматриваемые твиты не репрезентативны. Основными причинами вызвавшими это являются как специфика собранных наборов данных, так и особенности русскоязычного сегмента твиттер.

Построение набора данных, состоящего из большего числа твитов, сильно отличающихся от заголовков новостей, позволит получить более реалистичную оценку качества. Что в свою очередь приведёт к возможности оценить преимущество методов, использующих дополнительные признаки, предоставляемые предметной областью.

# Список литературы


[1] W. Guo, H. Li, H. Ji, and M. T. Diab. Linking tweets to news: A framework to enrich short text data in social media. - ACL, pages 239–249, 2013.

[2] Manos Tsagkias, Maarten de Rijke, Wouter Weerkamp. Linking Online News and Social Media. - ISLA, University of Amsterdam.

[3] T. Hoang-Vu, A. Bessa, L. Barbosa and J. Freire. Bridging Vocabularies to Link Tweets and News. - International Workshop on the Web and Databases (WebDB 2014), Snowbird, Utah, US, 2014.

[4] Weiwei Guo and Mona Diab. 2012a. Modeling sentences in the latent space. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.

[5] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[6] Nathan Srebro and Tommi Jaakkola. 2003. Weighted low-rank approximations. In Proceedings of the Twentieth International Conference on Machine Learning.

[7] Eric Huns. Hunseblog on Wordpress: URL: https://hunseblog.wordpress.com/2014/09/15/installing-numpy-and-openblas/.

[8] Арсеньев В.В., Сажин Ю.Б. Методические указания к выполнению организационно-экономической части дипломных проектов по созданию программной продукции. М.: изд. МГТУ им. Баумана, 1994. 52 с. 2.

[9] Под ред. Смирнова С.В. Организационно-экономическая часть дипломных проектов исследовательского профиля. М.: изд. МГТУ им. Баумана, 1995. 100 с.

[10] ГОСТ 34.601 "АС. Стадии создания".

[11] ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.