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 bridging словарей;

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 наиболее текстов, наиболее близких по частотности употребления слов.

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. Метод WTMF-G

Метод WTMG-G решает задачу установления связей между твитами и новостными статьями, путём построения модели, которая учитывает неявные связи между текстами. Метод был предложен в статье Linking Tweets to News: A Framework to Enrich Short Text Data in Social Media [1].

Метод WTMF-G представляет собой доработанный метод WTMF, который позволяет хорошо моделировать семантику коротких текстов, но не учитывает 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.

В статье показано, что добавление информации о взаимосвязи текст-текст позволяет повысить качество установления связей между твитами и новостными статьями.

1.2.3. 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 Linking Online News and Social Media [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 большого 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.

1.2.4. Связывание твитов с новостями на основе bridging словарей

Метод связывания твитов с новостными статья, основанный на bridging словарях (множество слов, которые встречаются только в твитах и, соответственно, не встречаются в новостях), предложен в статье Bridging Vocabularies to Link Tweets and News [3]. 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 хештэги, как механизм для категоризации твитов. Но этот подход обладает рядом недостатков, таких как: не все записи содержат хештеги, хештег не содержит информацию о событии, хештег сформулирован в слишком общей форме, твит содержит несколько хештегов. Следовательно использование одних хештегов 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. К 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 Bridging словарь и служат «мостом» к другим твитам.

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

1.3. Используемый в работе подход

В качестве 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.

Ниже представлено описание теории, необходимой для реализации метода WTMF-G.

1.3.1. Построение набора данных

Для решения задачи автоматического связывания твитов и новостей 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 набора 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.

1.3.2. Метод WTMF

Метод WTMF 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 $K \times N$.

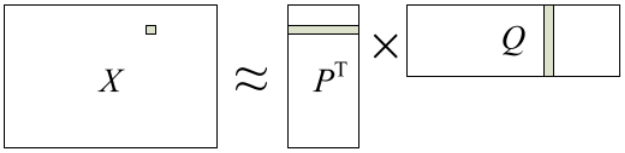

Рисунок 1 — Разложение TF-IDF матрицы (X) на произведение матриц P и Q

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 (λ - 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]):

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

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

Здесь $W'_i = diag(W_{i,\cdot})$ - диагональная матрица полученная из i-ой строчки матрицы W, аналогично $W'_j = diag(W_{\cdot,j})$ - 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.

1.3.3. Построение связей текст-текст

Построение связей текст-текст предполагает поиск потенциально семантически близких текстов. При построении связей текст-текст используются 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.

1.3.4. Метод WTMF-G

Метод WTMF-G (WTMF on Graphs) — представляет собой 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 δ 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.

1.4. Оценка качества

Решение задачи установления связей между твитами и новостными статьями неоднозначно. Как твиту может соответствовать несколько новостей, так и новостной статье может соотвествовать несколько твитов. Но в эталонном наборе данных для каждого твита существует связь с единственной новостью. В данном случае для оценки качества применимы метрики принятые в информационном поиске.

Мы рассматривает твит как запрос в терминологии информационного поиска, а список новостей как ответом. То есть для каждого твита мы получаем список новостей, ранжированный по мере убывания их схожести.

1.4.1. 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:

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

1.4.2. 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, TOP_{10}.

2. Реализация

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

1. сбор необходимой для решения задачи информации;
2. построение наборов данных;
3. применение к наборам данных методов машинного обучения;
4. получение рекомендаций новостей для произвольных твитов;
5. вариативность в выборе метода для построения рекомендаций;
6. возможность получить информацию о качестве используемого метода.

Программный комплекс реализуется с использование языка программирования Python 2.7.

2.1. Архитектура

Программный комплекс состоит из набора подсистем, которые выполняют набор функций:

1. получение данных из твиттера;
2. получение данных из новостной rss-ленты;
3. расшифровка коротких URL;
4. автоматическое построение набора данных;
5. построение набора данных на основе вручную построенных заготовок;
6. построение моделей для методов WTMF и WTMF-G;
7. построение рекомендаций для методов WTMF, WTMF-G и поиска схожести на основе частнотности употребления слов;
8. оценка качества рекомендаций;
9. получение результатов рекомендаций в пригодном для чтения формате;

подробное описание архитектуры системы приведённой на flowchart

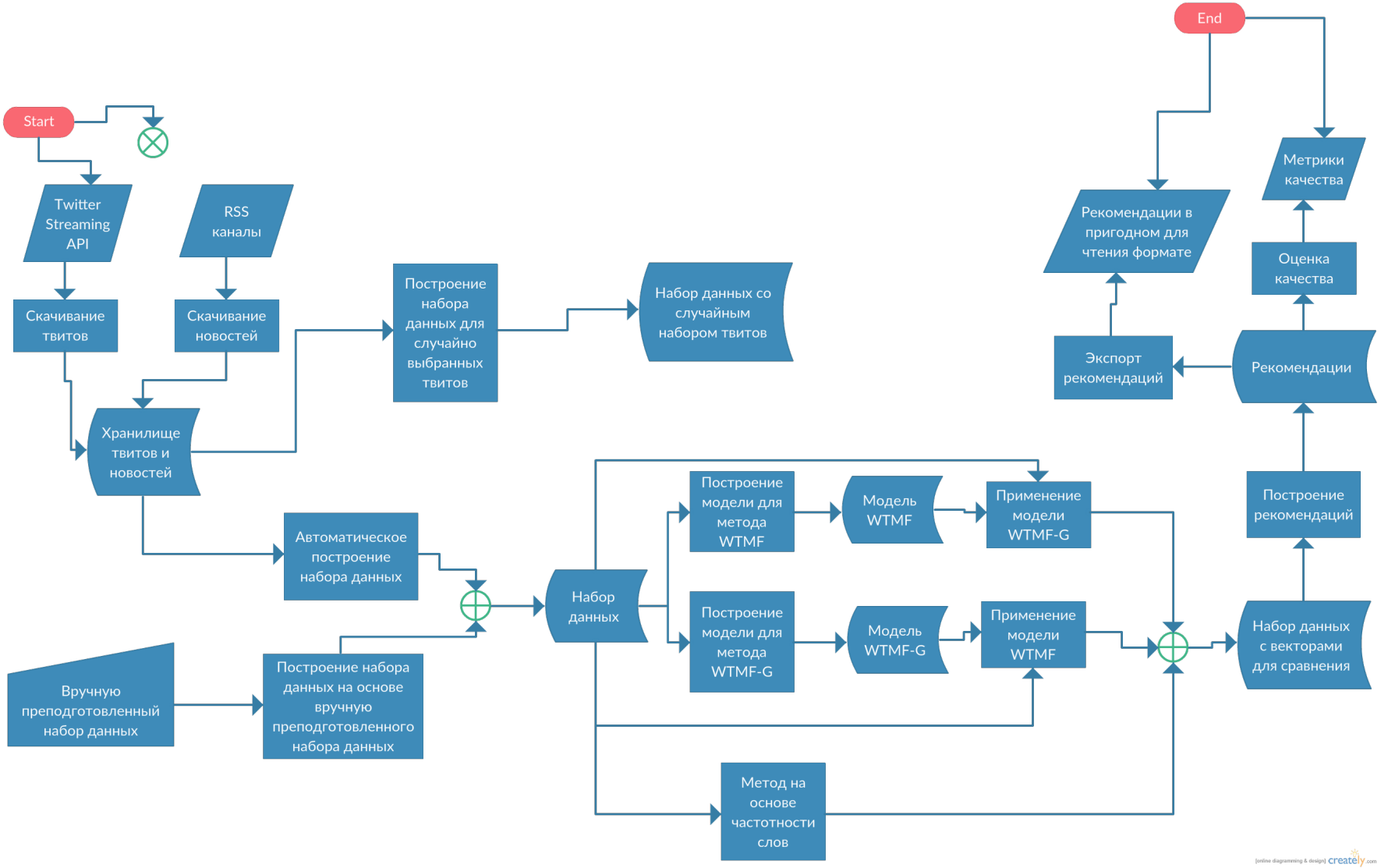

Рисунок 2 — flow chart

2.2. Получение данных

Для получения данных (твитов и новостей) необходимо:

1. реализовать скачивание твитов и новостей из разных новостных источников;
2. определить формат хранения скачиваемых данных;
3. скачать твиты за определённый промежуток времени;
4. расшифровать сокращённые URL;
5. определить список наиболее популярных новостных источников в твиттере;
6. в течение длительного времени собрать данные как с твиттера, так и с новостных источников.

2.2.1. Скачивание твитов

Для скачивания 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 необходимо на сайте https://apps.twitter.com/ зарегистрировать новое приложение и получить набор секретных ключей, которые требуются для авторизации. Для упрощения работы с Twitter Streaming API используется 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 используется информация публикации твитов. Каждое событие о создании твита описывается в виде большого количества параметров (несколько десятков), в работе используются 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 помещается в хранилище твитов. Хранилище твитов реализованно использованием библиотеки python shelve.

2.2.2. Скачивание новостных статей

Новостные статьи скачиваются в формате *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 потоками RSS используется 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 используются 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 библиотеки python shelve.

2.2.3. 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 2.4 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.

2.2.4. 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 очень сложно собрать и в дальнейшем обрабатывать эталонный набор новостей.

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

Как видно из таблицы 1 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 2.

Как видно из таблицы 2 среди твитов, собранных на довольно большом промежутке времени (неделя), популярные новостные источники составляют лишь малую долю от общего числа употребляемых ссылок (3%).

Было принято решение одновременно использовать 5 самых популярных новостных источников, а именно: `ria.ru`, `lifenews.ru`, `lenta.ru`, `russian.rt.com`, `www.gazeta.ru`.

Таблица 1: 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 2: 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

2.3. 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 решаются 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.

2.4. Формирование набора данных

Набор данных формируется на основе заранее собранной и предподготовленной информации. Информация собирается определённый, фиксированный отрезок времени. Для формирования набора данных использовалась информация собранная с 06.04.2016 по 17.04.2016. Было получено множество, основные характеристики которого приведены в таблице 3. В дальнейшей под собранными данными будет подразумеваться описанное в таблице 3 множество.

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

Таблица 3: 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

1. автоматическая разметка набора данных;

2. ручная разметка набора данных.

В результате разметки будет получаться некоторое количество пар твит-новость, в которых твит, практически полностью будет совпадать с заголовком новости. Будем в дальнейшем называть такие связи твит-новость, в которых менее половины слов из твита не встречаются в заголовке статьи *тривиальными*.

2.4.1. Автоматическое разметка набора данных

Автоматически построенный набор данных состоит из всех собранных новостей и твитов, которые содержат ссылку на одну из собранных новостей. Получаем множество состоящее из 4324 твитов, 13711 новостей, а также 4324 связей между ними.

Проанализируем полученный набор данных, нас интересует насколько в парах твит-новость, твит отличается от соответсутющей новости. Для этого для каждого твита берём его текст, для каждой новости берём заголовок. Все тексты поэтапно преобразовали 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 3 изображена зависимость количества пар твит-новость от длины пересечения слов твита и слов новости, нормализованной по длине новости. Как

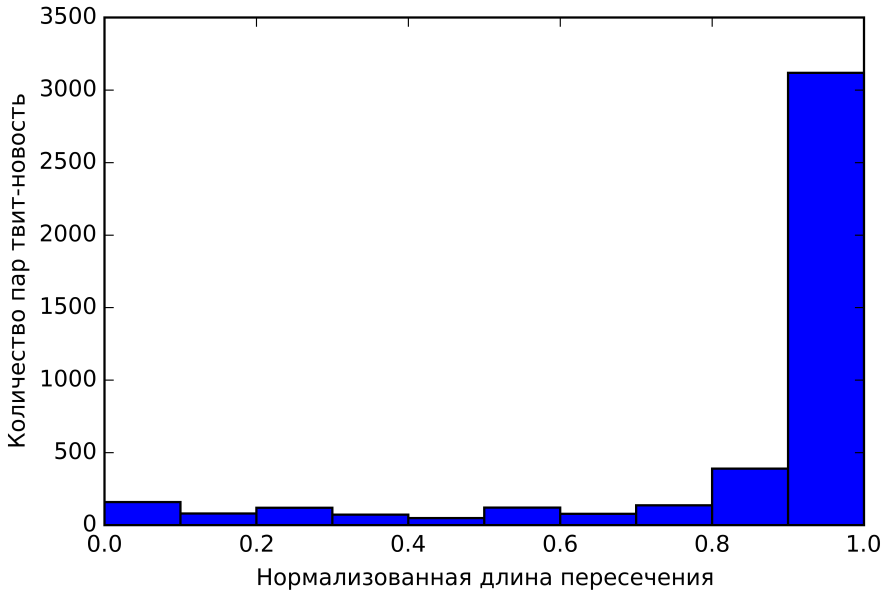

Рисунок 3 — Зависимость количества пар твит-новости от нормализованный длины пересечения множества слов (автоматически размеченный набор данных).

видно из рисунка 3 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 4. Как видно из рисунка 4 количество твитов более с чем половиной уникальных слов достаточно мало.

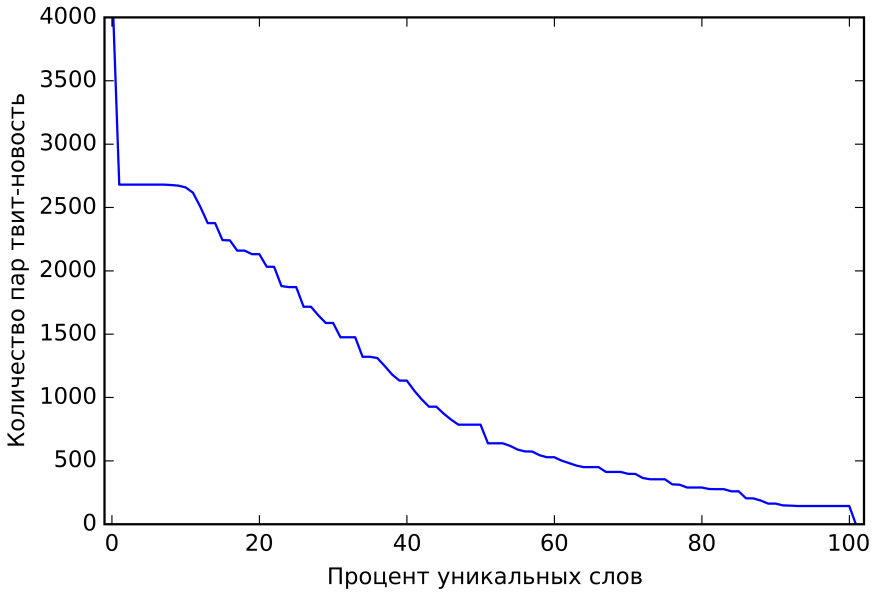

Рисунок 4 — Зависимость количества пар твит-новости от процента уникальных слов в твите (автоматически размеченный набор данных).

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.

2.4.2. Ручная разметка набор данных

Для получения большего количества нетривиальных пар твит-новость была предпринята ручная разметка набора данных. Для ручной разметки необходимо предподготовить 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 получается множества пар твит-ранжированный список новостей.

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

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

Для сравнения вручную построеннго 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 5 изображена зависимость количества пар твит-новость от длины пересечения слов твита и слов новости, нормализованная по длине новости. Как видно из рисунка 5 было получено распределение намного более близкое к равномерному, чем в случае автоматически размеченного набора данных.

На рисунке 6 изображена зависимость количества пар твит-новость от процента уникальных слов в твите. Как видно из рисунка 6 количество твитов более с чем половиной уникальных слов сравнимо с аналогическим 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 значение полученных метрик приведены в таблице 4. Как видно из таблицы 4 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).

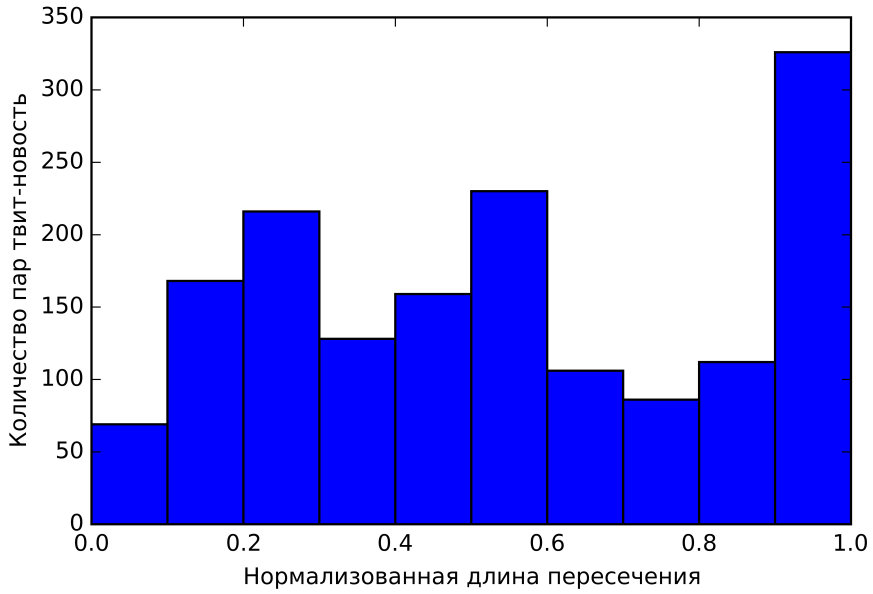

Рисунок 5 — Зависимость количества пар твит-новости от нормализованный длины пересечения множества слов (вручную размеченный набор данных).

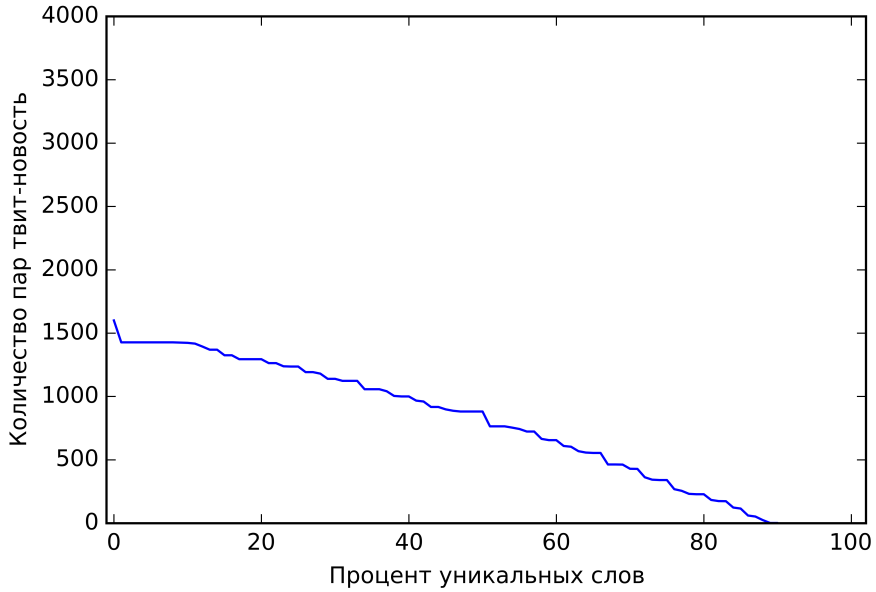

Рисунок 6 — Зависимость количества пар твит-новости от процента уникальных слов в твите (вручную размеченный набор данных).

2.4.3. Построение связей текст-текст

Реализация построения связей, описанного в главе 1.3.3, привела к получению избыточного количества ложных связей на используемом наборе данных. По этой

Таблица 4: 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

причине реализация была дополнена следующими эвристическими ограниченями:

1. удаление слишком популярных хэштегов (слишком популярным считаем хэштег, который встретился более чем в 10 твитах из обучающей выборки);
2. твиты, считаются связанными когда оба содержать не менее 2 одинаковых 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).

2.4.4. Сформированные набор 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 5.

Таблица 5: 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

2.5. Метод WTMF

Модель для метода WTMF строится на основе заранее построенного 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, полученных из набора данных, строится модель, пригодная для сериализации, которая состоит из матрицы P (здесь и далее используются обозначения введённые в главе 1.3.2). 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. λ — 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.

2.6. 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, полученных из набора данных, строится модель, пригодная для сериализации, которая состоит из матрицы 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. δ — коэффициент, задающий степень влияния связей вида текст-текст.

Применение полученной модели на множество твитов происходит 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:

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

В результате получаем вектора для сравнения твитов из заданного множества.

2.7. 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 задача по повышению эффективности работы с матрицами актуальна.

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

Оптимизируется формула получения строк матрицы P, используемая при построении моделей WTMF и WTMF-G. На каждой итерации построения модели происходит многократное выполнение формулы (количество 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 6.

Таблица 6: 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

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

3. Руководство пользователя

Необходимо переписать главу под текущие реалии

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

- twnews_consumer — консьюмер, который позволяет выкачивать твиты с твиттера и новости с rss каналов.
- twnews — пакет, позволяющий по твитам и новостям, произвести все необходимые преобразования данных и на основе полученных признаков произвести обучение и оценку модели.

Оба пакета ориентированы на работу в операционных системах из семейства linux. Для начала работы необходимо иметь установленный менеджер пакетов для языка Python — pip, а также установить setuptools:

```
$ pip install setuptools
```

Также необходимо выкачать git-репозиторий: https://github.com/art-vybor/twnews.git. Если установлен пакет git, то это можно сделать следующим образом:

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

Для корректной работы пакетов, все указываемые в конфигурации директории должны быть заранее созданы.

3.1. Пакет twnews_consumer

Пакет twnews_consumer располагается в папке consumer в корне репозитория. Он позволяет выкачивать и сохранять в формате, удобном для дальнейшей работы пакета twnews, твиты и новости.

Конфигурирование пакета производится в файле twnews_consumer/defaults.py. Описание задаваемых параметров находится в таблице 7.

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, время публикации и информация о том, является ли он ретвитом (ретвит — твит, представляющий собой, ссылку на ранее созданный твит).

Таблица 7: 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/avyboro v/twnews_data/’	Путь до директории, в которую будут сохранены данные
RSS_FEEDS	{’ria’: {’rss_url’: ’http://ria.ru/export/rss2/index.xml’}, ’lifenews’: {’rss_url’: ’http://lifenews.ru/xml/feed.xml’}}	Новостные источники, которые требуется выкачать
TWEETS_LANGUAGES	[’ru’]	Список языков, твиты с использованием которых выкачиваются из твиттера

3.1.1. Установка

Для установки, необходимо зайти в папку consumer, находящуюся в корне репозитория и выполнить команду:

```
$ make install
```

Во время установки, нужно будет ввести пароль, для распаковки секретного ключа, который необходим для работы с API твиттера.

3.1.2. Использование

Для того, чтобы начать выкачивать новости, необходимо запустить команду:

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

Для того, чтобы начать выкачивать сообщения твиттера, необходимо запустить команду:

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

Узнать информацию о работе программы можно из файла лога. 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))
```

3.2. Пакет twnews

Пакет twnews располагается в папке core в корне репозитория. Он позволяет обрабатывать данные полученные с помощью консьюмера с целью построения и оценки качества модели WTMF.

Конфигурирование пакета производится в файле twnews/defaults.py. Описание задаваемых параметров находится в таблице 8.

Результатом работы пакета является построенная модель WTMF, для которой измерено её качество. Перед построением модели, необходимо выполнить команду, которая разрешает ссылки (может занять длительное время).

Таблица 8: 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/avyboxnov/twnews_data/’	Путь до рабочей директории в которой лежат выкаченные с помощью консьюмера данные
DATASET_FRACTION	1.0	Часть датасета, которая будет использована для обучения модели
TMP_FILE_DIRECTORY	’/tmp/twnews/’	Путь до директории в которую будут сохранены временные данные

3.2.1. Установка

Для установки, необходимо зайти в папку core, находящуюся в корне репозитория и выполнить команду:

```
$ make install
```

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

3.2.2. Использование

Для того, чтобы разрешить ссылки, необходимо запустить команду:

```
$ twnews_consumer --resolve
```

Для того, чтобы посмотреть статистику по упомянутым в коллекции твитов ссылкам нужно выполнить команду:

```
$ twnews_consumer download --analyse_urls
```

Для построения модели небходимо запустить команду:

```
$ twnews_consumer download --run_pipe
```

Узнать информацию о работе программы можно из файла лога. 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
```

4. Тестирование

Главное целью тестирования 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.

4.1. 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, λ, w_m. Параметры K и I влияют на время построения модели, а параметры λ 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: λ и w_m. Для этого фиксируются остальные параметры: $I = 1$, $K = 30$. Для начала находится оптимальный порядок значений начального приближения. Результаты занесены в таблицу 9.

Таблица 9: Качество работы алгоритма WMTF для различных значений λ и w_m при фиксированных значениях $I = 1$, $K = 30$.

λ/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 9 в целом получена достаточно однородная картина для всех порядков λ и w_m. Заметное снижение качества происходит при большом порядке w_m и малом порядке λ. 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 10.

Таблица 10: Качество работы алгоритма WMTF для различных значений λ и w_m при фиксированных значениях $I = 1$, $K = 30$.

λ/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 10 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 11.

Таблица 11: Качество работы алгоритма WMTF для различных значений K и I при фиксированных значениях $\lambda = 0.95$, $w_m = 1.95$.

K/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 11 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 12.

Из таблицы 12 получаем оптимальные значения коэффициента $K = 90$

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

Таблица 12: Качество работы алгоритма WMTF для различных значений K при фиксированных значениях $I = 1$, $\lambda = 0.95$, $w_m = 1.95$.

K	**Значение метрики RR**
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

4.2. Оптимизация качества WTMF-G, путём варьирования параметров

Оптимизация параметров ещё не завершена, существующая и очень, очень грубая оценка приведена ниже

Оптимизация параметров модели для метода WTMF-G будет производится на наборе данных auto_cleared, используя метрику MRR. Модель WTMF зависит от четырёх параметров: K, I, δ, w_m. Параметры K и I влияют на время построения модели, а параметры λ и w_m не влияют на время построения модели.

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

Оптимизируется параметр δ. Для этого фиксируются остальные параметры: $K = 90$, $I = 1$, $w_m = 1.95$. Для начала находится оптимальный порядок значений начального приближения. Результаты занесены в таблицу 13. Как видно из таблицы 13

Таблица 13: Качество работы алгоритма WMTF-G для различных значений δ при фиксированных значениях $K = 90$, $I = 1$, $w_m = 1.95$.

δ	**Значение метрики RR**
0.001	0.5508
0.01	0.5307
0.1	0.5695
1	0.5311
10	0.5303
100	0.5203

максимальное значение метрики получено при $\delta = 0.1$. Для уточнения значения коэффициента δ, производится исследование качества работы алгоритма в окрестностях максимального значения метрики. Результаты приведены в таблице 14. Из

Таблица 14: Качество работы алгоритма WMTF-G для различных значений δ при фиксированных значениях $K = 90$, $I = 1$, $w_m = 1.95$.

δ	**Значение метрики RR**
0.05	0.5340
0.1	0.5695
0.15	0.5380
0.25	0.5533
0.3	0.5195
0.35	0.5329

таблицы 14 получаем оптимальные значения коэффициента $\delta = 0.1$.

В итоге оптимизации качества рекомендаций на основе алгоритма WMTF-G были получены оптимальные параметры: $K = 90$, $I = 1$, $\delta = 0.1$, $w_m = 1.95$.

4.3. Сравнительные результаты

Для выявления влияния добавления связей текст-текст на результаты работы метода WMTF-G производится сравнительное тестирование алгоритма WTMF и WTMF-G. Результаты тестирования приведены в таблице 15.

Таблица 15: 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.	0.	0.	0.	0.
auto	0.8640	0.	0.	0.	0.	0.
total	0.8196	0.	0.	0.	0.	0.
cutted	0.8630	0.	0.	0.	0.	0.
manual_nt	0.6194	0.	0.	0.	0.	0.
auto_nt	0.5297	0.5695	0.	0.	0.	0.
total_nt	0.5729	0.	0.	0.	0.	0.
cutted_nt	0.6495	0.	0.	0.	0.	0.

Как видно из таблицы 15 ...

Сравним метод основанный на частотности употребления слов и WTMF-G. Метод основанный на частотности употребления слов обозначим как TF-IDF. Результаты тестирования приведены в таблице 16.

Таблица 16: Сравнительное тестирование алгоритмов TF-IDF и WTMF-G.

Набор данных	**Метрика MRR**		**Метрика** TOP_1		**Метрика** TOP_3	
	TF-IDF	**WTMF-G**	**TF-IDF**	**WTMF-G**	**TF-IDF**	**WTMF-G**
manual	0.8336	0.	0.	0.	0.	0.
auto	0.8817	0.	0.	0.	0.	0.
total	0.8610	0.	0.	0.	0.	0.
cutted	0.9075	0.	0.	0.	0.	0.
manual_nt	0.7565	0.	0.	0.	0.	0.
auto_nt	0.6048	0.5695	0.	0.	0.	0.
total_nt	0.6914	0.	0.	0.	0.	0.
cutted_nt	0.7485	0.	0.	0.	0.	0.

Как видно из таблицы 16 ...

объяснение влияния различных датасетов, специфики русского языка и сравнение с результатами статьи.

5. 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 А* – 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.

5.1. Трудоемкость разработки программной продукции

Трудоемкость разработки программной продукции (τ_{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 τ_{TZ} — трудоемкость разработки технического задания на создание программного продукта; τ_{EP} — трудоемкость разработки эскизного проекта программного продукта; τ_{TP} — трудоемкость разработки технического проекта программного продукта; τ_{RP} — трудоемкость разработки рабочего проекта программного продукта; τ_V — трудоемкость внедрения разработанного программного продукта.

5.1.1. Трудоемкость разработки технического задания

Расчёт трудоёмкости разработки технического задания (τ_{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:

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

5.1.2. Трудоемкость разработки эскизного проекта

Расчёт трудоёмкости разработки эскизного проекта (τ_{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 форму-

лам:

$$T_{RZ}^{E} = t_E \cdot K_{RZ}^{E},$$

$$T_{RP}^{E} = t_E \cdot K_{RP}^{E},$$

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_{RZ}^{E} — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ЭП. В нашем случае (совместная разработка с разработчиком 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:

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

5.1.3. Трудоемкость разработки технического проекта

Трудоёмкость разработки технического проекта (τ_{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 разновид-

ностей 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 информации; 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:

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

5.1.4. Трудоемкость разработки рабочего проекта

Трудоёмкость разработки рабочего проекта (τ_{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)):

$$t^R_{RZ} = 15,$$

$$t^R_{RP} = 91.$$

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:

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

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:

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

5.1.5. Трудоемкость выполнения стадии «Внедрение»

Расчёт трудоёмкости разработки технического проекта (τ_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:

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

5.2. 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 17. Из таблицы получаем, что фонд рабочего времени

$$F = 96.$$

Таблица 17: 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.

5.3. Ленточный график выполнения работ

На основе рассчитанных в главах 5.1, 5.2 трудоёмкости и фонда рабочего времени найдём количество рабочих дней, требуемых для выполнения каждого этапа разработка. Результаты приведены в таблице 18.

Таблица 18: 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 18 в ленточный график (таблица 19), в ячейки столбца “продолжительности рабочих дней” заносятся времена, которые требуются на выполнение соответствующего этапа. Все исполнители работают одновременно.

Таблица 19: 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	

5.4. 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 20 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 20: 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 21.

Таблица 21: 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 $5611500 + 1750 + 77760 = 5691010$ рублей.

5.5. 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 ρ_N — норматив рентабельности, $\rho_N = 30\%$;

Получаем стоимость программного продукта:

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

5.6. 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 22 находится расчёт чистого дисконтированного дохода. График его изменения приведён на рисунке 7.

Таблица 22: 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

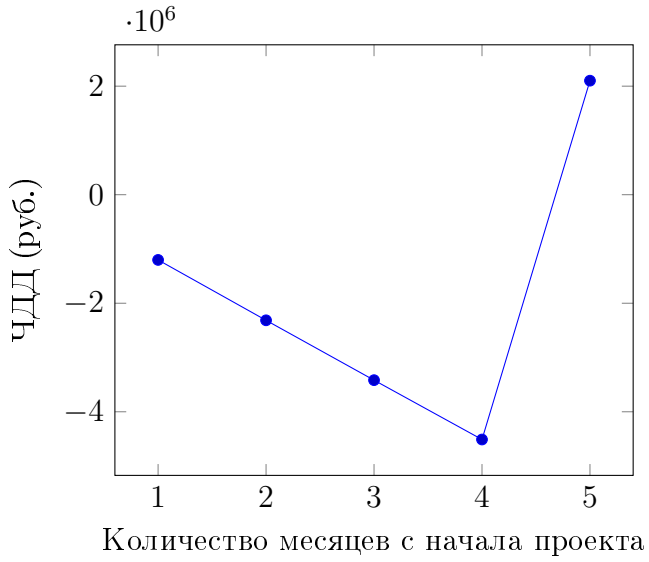

Рисунок 7 — График изменения чистого дисконтированного дохода

Согласно проведенным расчетам, проект является рентабельным. Разрабатываемый проект позволит превысить показатели качества существующих систем и сможет их заменить. Итоговый ЧДД составил: 2101032 рублей.

5.7. Результаты

В рамках организационно-экономической части был спланирован календарный график проведения работ по созданию подсистемы поддержки проведения диа-

гностики 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.

6. Заключение

Что получилось. Должно отображать задачи описанные во введении

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".