Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н. Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э.Баумана)

Факультет «Информатика и системы управления»
Кафедра «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ НА ТЕМУ:

Метод ранжирования информационных источников по степени доверия

Студент ИУ7-81 _____ П. Ю. Павелко

Руководитель ВКР _____ Д. Е. Бекасов

Нормоконтролер _____ _____

Москва, 2017

РЕФЕРАТ

Расчетно-пояснительная записка 46 стр., 17 рис., 2 табл., 22 ист.

НОВОСТНЫЕ ИСТОЧНИКИ, РАНЖИРОВАНИЕ, КЛАСТЕРИЗАЦИЯ, ДОВЕРИЕ

Объектом исследования является рейтинг источников по степени доверия к ним. Объект разработки — программное обеспечение для мониторинга новостей с последующим ранжированием источников.

Цель работы — разработка и реализация метода ранжирования новостных источников по степени доверия.

Задачи, решаемые в работе:

— Анализ предметной области;
— Разработка метода ранжирования новостных источников;
— Реализация выбранного метода и исследование его применимости к задаче.

В первой части работы описываются существующие подходы к решению задачи, проводится анализ возможных решений. Во второй части описываются выбранные методы и внесенные в них модификации. В третьей части описываются технологии, примененные при реализации метода и тестирование. В четвертой части проведены экспериментальные исследования характеристик метода и применимости к решаемой задаче.

Предлагаемые направления развития:

— Построение тематического рейтинга источников;
— Ранжирование экспертов;
— Нахождение дубликатов и первоисточников новостей.

Поставленная цель была достигнута: метод ранжирования новостных источников по степени доверия. Были рассмотрены существующие недостатки решения и предложены пути дальнейшего развития.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном пространстве сформировалось огромное количество информационных источников. К примеру, новостной агрегатор «Яндекс.Новости» только в России получает более 100 тысяч новостей ежедневно от почти 7000 информационных источников.

Проблема выбора новостного источника усугубляется тем, что СМИ периодически публикуют недостоверную информацию. Существуют различные проекты по проверке новостей на достоверность, такие как stopfake.org. Поскольку такая проверка осуществляется вручную, эксперты сильно ограничены во времени и могут проверять только наиболее известные информационные источники, оставляя без внимания более мелкие, но при этом всё равно читаемые новостные сайты.

Для решения данной проблемы необходим метод распространения оценки экспертов на схожие новости менее популярных источников новостей с последующим ранжированием источников по степени доверия к ним.

Целью данной работы является разработка такого метода для ранжирования информационных источников и реализация системы мониторинга новостей с последующим ранжированием источников по степени доверия.

Целью данной работы является разработка и реализация информационной системы для сбора информации в сети Интернет и последующего поиска по ней.

В рамках работы необходимо решить следующие задачи:

а) Проанализировать предметную область;
б) Разработать метод ранжирование источников;
в) Разработать программное обеспечение;
г) Исследовать применимость разработанного метода.

1 Аналитический раздел

В указанной постановке задача не решалась, однако можно очертить некоторые этапы, которые так или иначе должны присутствовать в решении задачи:

— Сбор новостей из открытых информационных источников;
— Кластеризация новостей по сюжетам;
— Ранжирование источников.

Перед анализом существующих методов каждого из этапов рассмотрим сущности предметной области (рис. 1.1):

— Источники — поставщики новостей, идентифицируются адресом;
— Лента — новостные ленты, содержащие список новостей за последнее время, предоставляются источниками;
— Новость — непосредственно текст новости, опубликованной источником;
— Сюжет — объединение различных новостей, относящихся к одному и тому же событию.

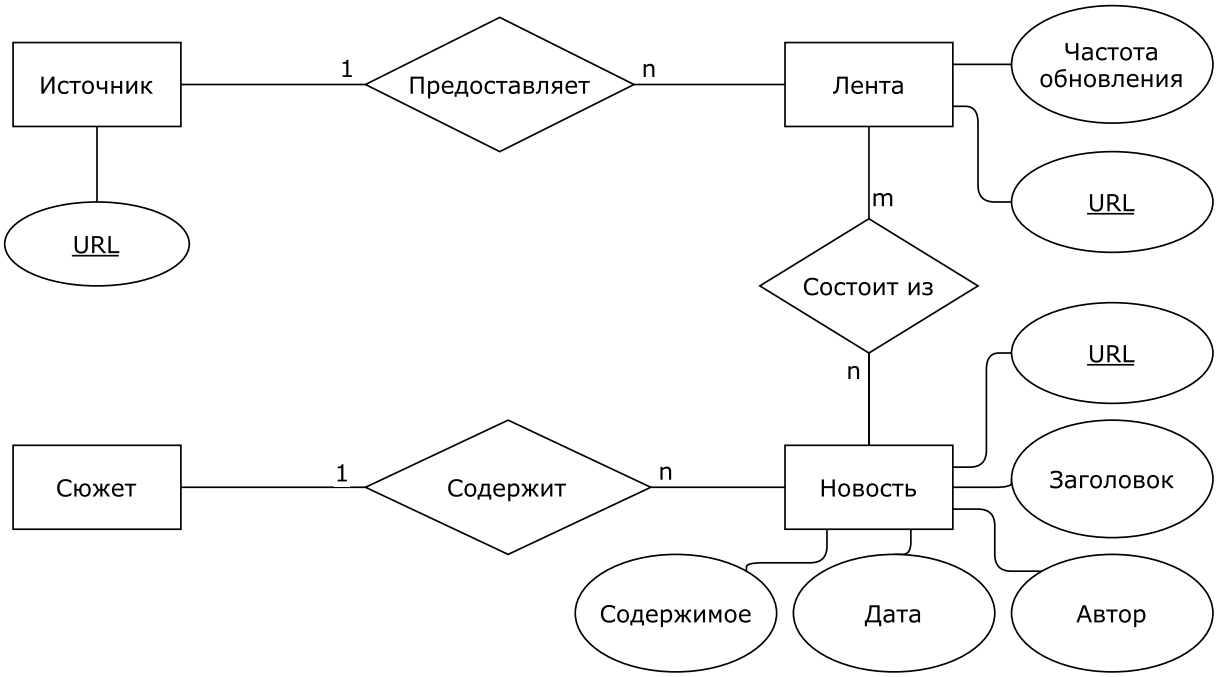


Рисунок 1.1 — Сущности предметной области.

1.1 Сбор новостей

Сбор актуальных новостей является важной задачей в разрабатываемой системе, потому что качество и актуальность собранной информации влияет на выделение сюжетов, что, в свою очередь, виляет на ранжирование источников.

С данной задачей связано большое количество проблем: огромное число новостных источников, ограничительная пропускная способность каналов, зашумлённость страниц второстепенной информацией и так далее.

1.1.1 Новостные ленты

Большинство новостных источников, включая новостные сайты и популярные группы в социальных сетях, предоставляют rss- и atom-ленты — небольшие xml-документы, в которых описываются последние новости и ссылки на них. Такие ленты стараются поддерживать актуальными, поскольку они используются многими новостными агрегаторами, которые активно используются пользователями.

Важно отметить, что в самих лентах не хранится непосредственно текст новостей, а лишь ссылки на html-страницы, расположенные на сайте источников, поэтому после получения ленты необходимо выгружать страницы с сайта источника.

1.1.2 Фильтрация посещённых страниц

Один источник может предоставлять несколько лент, например по ленте на категорию или тематику. Поэтому одна и та же новость может быть получена из нескольких лент, что будет создавать дополнительную нагрузку на систему. Поэтому имеет смысл отбрасывать уже посещённые новостные страницы. Очевидное на первый взгляд решение проблемы — использование ассоциативных массивов — довольно требовательно к памяти, поэтому имеет смысл рассмотреть использование веро-

ятностных фильтров. Наиболее популярным фильтром подобного рода является фильтр Блума.

1.1.2.1 **Фильтр Блума**

Фильтр Блума — вероятностная структура данных, позволяющая компактно хранить множество элементов и проверять принадлежность заданного элемента к множеству. При этом существует вероятность получить ложноположительный результат, то есть ситуацию, когда элемента в множестве нет, но фильтр сообщает, что элемент есть.

Фильтр Блума может использовать любой объём памяти, заранее заданный пользователем, причём чем он больше, тем меньше вероятность ложного срабатывания. Поддерживается операция добавления новых элементов в множество, но не удаления существующих (если только не используется модификация со счётчиками).

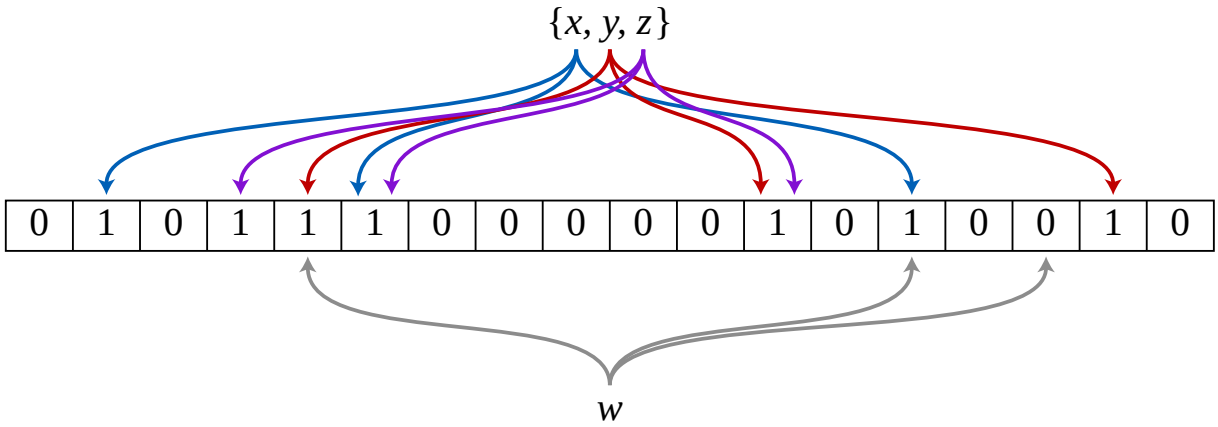


Рисунок 1.2 — Пример фильтра Блума с $m = 18$ и $k = 3$.

Фильтр Блума представляет собой битовый массив из m бит. Изначально, когда структура данных хранит пустое множество, все m бит обнулены. Пользователь должен определить k независимых хеш-функций h_i, отображающих каждый элемент в одну из m позиций битового массива достаточно равномерным образом.

Для добавления элемента e необходимо записать единицы на каждую из позиций $h_i(e)$ битового массива.

Для проверки принадлежности элемента e к множеству хранимых элементов, необходимо проверить состояние битов h_i. Если хотя бы один из них равен нулю, элемент не может принадлежать множеству (иначе бы при его добавлении все эти биты были установлены). Если все они равны единице, то структура данных сообщает, что e принадлежит множеству. При этом может возникнуть две ситуации: либо элемент действительно принадлежит к множеству, либо все эти биты оказались установлены по случайности при добавлении других элементов, что и является источником ложных срабатываний в этой структуре данных.

Согласно [1], для заданных n — числа ожидаемых элементов в множестве и p — максимальной вероятности ложноположительного срабатывания возможно вычислить оптимальный размер m как

$$m = -\frac{n \ln p}{(\ln 2)^2}, \tag{1.1}$$

и оптимальное количество хеш-функций как

$$k = \frac{m}{n} \ln 2. \tag{1.2}$$

1.1.3 Стандарт исключений для роботов

Следующая проблема, с которой нам предстоит столкнуться, заключается в том, что после получения новостной ленты нам необходимо сделать множество запросов к одному серверу, что обычно не нравится владельцам сайтов.

Когда владельцы сайта хотят ограничить доступ к определённым страницам для ботов, они используют для этого файл `robots.txt`, находящийся в корне сайта (то есть по пути `/robots.txt`). Это может быть полезно для ограничения частоты запросов со стороны роботов, запрета индексирования динамический и служебных страниц.

Формат файла имеет следующий вид:

```
<поле>:<необязательный пробел><значение><необязательный пробел>
```

Стоит заметить, что вместо пробела могут использоваться также другие пробельные символы, а `поле` является регистронезависимым.

Основные используемые поля:

— `user-agent`: стандартное поле, используется для задания имени бота, которому адресованы последующие правила, либо `*`, если всем;

— `disallow`: стандартное поле, используется для задания пути, индексация которого запрещена. В пути разрешены два спец-символа: `*` (любое количество любых символов) и `$` (конец пути). Причём путь, к примеру, `/path/to/file` действует аналогично `/path/to/file*`;

— `crawl-delay`: нестандартное поле, используется для задания минимальной задержки в секундах между запросами со стороны робота.

1.1.4 Разбор страницы

Далеко не вся информация на странице ценна для поставленной задачи. Например, информация, размещённая в подвале и по сторонам страницы несёт обычно служебную информацию, которая может быть полезна при обходе страниц (так как может содержать полезные ссылки), но бесполезна конечному пользователю и не содержит информации, относящейся к новости. Поэтому ставится вопрос о выделении основного содержимого, то есть непосредственно текста новости, из html-страницы. Для этого необходимы методы выделения такой информации, которые в основном базируются на эвристики и неплохо работают во многих случаях [2].

1.1.4.1 Выделение основного содержимого

Для получения основного содержимого во многих случаях достаточно сохранять только текстовые элементы HTML, то есть блоки текста, которые не прерывались разметкой, которые имеют более чем десяток слов. Люди выбирают один из двух типов текста для двух различных мотивов написания текста: «навигационного» и «информационного».

Для элементов навигации обычно применяется текст, состоящий из нескольких слов (например, «STOP», «Прочтите это», «Нажмите

здесь»), в то время как для основного содержимого используется много слов.

В то время как это разделение работает во множестве случаев, все становится сложнее с заголовками, короткими предложениями, отказами от ответственности, авторскими правами и другими колонтитулами.

Есть более сложные стратегии, а также функции, которые помогают отделять основное содержание от шаблонного [3]:

а) Рассмотренная выше плотность HTML-тегов;

б) Ссылочная плотность (количество слов внутри ссылок по сравнению с общим количеством слов в блоке);

в) Текстовые характеристики: количество запятых, длина текста и другие;

г) Связь текущего блока с контекстом (с предыдущими и следующими блоками);

д) DOM-структура документа (`<article>`, `<section>` и другие);

е) Визуальное изображение страницы (например, большие изображения, окружённые текстом);

Результирующий алгоритм, используя комбинацию этих факторов, рекурсивно оценивает все узлы документа, находя наиболее похожие на основное содержимое.

1.1.4.2 **Декодирование мнемоник HTML**

Мнемоника — это конструкция SGML, которая ссылается на символ из набора символов текстового файла. В HTML предопределено большое количество спецсимволов. Чтобы вставить определённый символ в разметку, нужно вставить определенную ссылку-мнемонику в HTML-структуру.

Мнемоника имеет вид `&...;`. Так, например, буква «q» может представляться как `&#113;`, а «—» как `&mdash;`.

1.1.4.3 **URL нормализация**

Различные URL могут указывать на одну и ту же страницу, но отличаться при этом незначительно.

URL нормализация — процесс приведения URL к каноничному виду, путём применения различных правил трансляции адресов. Не все правила дают строго эквивалентные URL, однако на практике эти эвристики работают неплохо [4].

Ключ страницы — своего рода хеш, получаемый в результате агрессивной нормализации, то есть применения всех правил, указанных в данном разделе.

Ключ страницы может служит идентификатором новости, ленты и источника.

Относительно безопасные преобразования:

а) Конвертация в нижний регистр компонентов схемы и хоста: `HTTP://www.Example.com/` в `http://www.example.com/`;

б) Удаление относительных каталогов, сегментов-точек: `http://example.com/../a/b/../c/./d` в `http://example.com/a/c/d`;

в) Удаление фрагментов: `http://example.com/bar.html#section1` в `http://example.com/bar.html`;

г) Удаление конечного слеша: `http://example.com/foo/` в `https://example.com/foo`;

д) Удаление порта по умолчанию: 80 для http и 443 для https `http://example.com:80` в `http://example.com`;

е) Перевод IDN в unicode: `http://xn--e1afmkfd.xn--80akhbyknj4f/` в `http://пример.испытание/`.

Преобразования, которые можно использовать для проверки эквивалентности двух страниц, но не при запросе:

а) Удаление головного индекса (`index.html`, `default.aspx` и другие):

`http://example.com/index.html` в `http://www.example.com/`;

б) Удаление дублированных слешей:

`http://example.com/foo//bar.html` в `http://example.com/foo/bar.html`;

в) Удаление `www.`:

`http://www.example.com/` в `http://example.com/`;

г) Сокращение идентификаторов протокола:

`https://example.com` в `http://example.com`;

д) Конвертация в нижний регистр всего URL:

`HTTP://example.COM/TEST.html` в `http://example.com/test.html`;

Такие преобразование применимы для построения ключа страницы, но не для упрощения адреса ссылки.

1.2 Кластеризация новостей

1.2.1 Векторное представление новости

Чтобы иметь возможность группировать новости по сюжетами нам необходим метод, позволяющий сравнивать новостные документы. К сожалению, две новости не могут сравниваться непосредственно, но могут быть представлены в векторной форме, допускающей такие сравнения.

Наиболее популярной формой такого представления является представление «мешком слов» — то есть вектором $\mathbf{x}$, где каждая компонента соответствует определённому слову, входящему в данную новость x [5].

Однако же у такой модели есть недостаток: представляя текст как «мешок слов», мы теряем контекст. Например, документы «кролик быстрее, чем черепаха» и «черепаха быстрее, чем кролик» имеют одинаковое векторное представление.

Простейшей характеристикой слова, которую можно использовать в данном представлении, является частота вхождения слова (от

англ. TF — term frequency) в новость:

$$tf_{w,x} = \frac{n_{w,x}}{|x|}, \tag{1.3}$$

где $|x|$ — длина новости x (количество слов в ней);

$n_{w,x}$ — количество вхождений слова w в новость x.

Однако данная характеристика обладает существенным недостатком: популярные слова имеют больший вес по сравнению с менее популярными словами, что понижает качество кластеризации. Для решения этой проблемы вводят обратную частоту документа.

Обратная частота документа (от англ. IDF — inverse document frequency) — инверсия частоты, с которой слово встречается в документах. Учёт IDF уменьшает вес широкоупотребительных слов.

$$idf_w = \log \frac{N}{N_w}, \tag{1.4}$$

где N — количество новостей в коллекции;

N_w — количество новостей, содержащих слово w, $N_w = |\{x : x \ni w\}|$.

Объединяя (1.3) и (1.4), получаем TF-IDF характеристику:

$$tf\text{-}idf_{w,x} = tf_{w,x} \cdot idf_w \tag{1.5}$$

Векторное представление для новости x в данном случае принимает вид:

$$\mathbf{x} = (tf\text{-}idf_{w_1,x}, \quad tf\text{-}idf_{w_2,x}, \quad ...) \tag{1.6}$$

Теперь вес некоторого слова пропорционален количеству употребления этого слова в новости, и обратно пропорционален частоте употребления слова в других новостях коллекции.

1.2.2 **Предобработка**

1.2.2.1 **Стоп-слова**

Существование заведомо высокочастотных слов (таких как «an», «and», «и», «как» и др.) ведёт к раздуванию векторного представления.

При этом вклад таких слов в семантику новости минимален и не влияет на кластеризацию по сюжетам. Поэтому разумно фильтровать данные слова при переводе новости в соответствующую векторную форму.

1.2.2.2 Лемматизация и стемминг

При построении векторного представления в виде «мешка слов» нет смысла различать формы (склонения, спряжения) одного и того же слова. Это приведёт к неоправданному разрастанию словаря, дроблению статистики, увеличению ресурсоёмкости и снижению качества модели кластеризации.

Лемматизация — это приведение каждого слова в документе к его нормальной форме. В русском языке нормальными формами считаются: для существительных — именительный падеж, единственное число; для прилагательных — именительный падеж, единственное число, мужской род; для глаголов, причастий, деепричастий — глагол в инфинитиве.

Стемминг — это более простая технология, которая состоит в отбрасывании изменяемых частей слов, главным образом, окончаний. Она не требует хранения словаря всех слов и основана на правилах морфологии языка. Недостатком стемминга является большее число ошибок.

1.2.3 Меры схожести новостей

1.2.3.1 Эвклидова метрика

Эвклидова метрика — «классическое» расстояние между двумя точками в пространстве [6]:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{1.7}$$

где n — размерность пространства.

Эвклидова метрика является наиболее просто мерой сходства между двумя векторами, однако плохо себя показывает относительно

других мер при работе с текстами [7] и требует вычисления квадратного корня.

1.2.3.2 **Косинусная мера**

Косинусная мера основана предположении о том, что направление векторов имеет большее значение, чем длина каждого вектора и расстояние между ними [8].

Данный метод, как следует из названия, основывается на косинусе угла между векторами:

$$sim_c(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \tag{1.8}$$

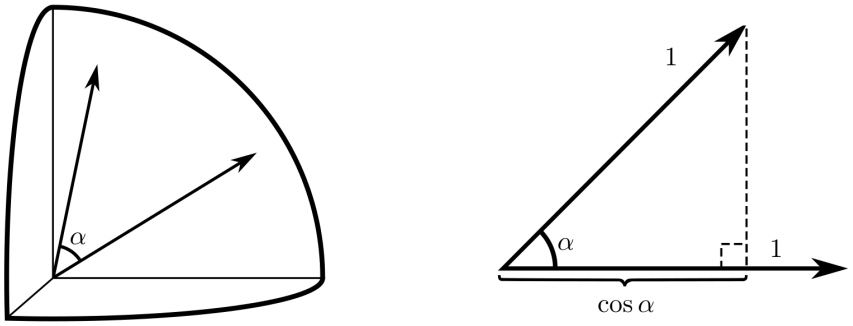


Рисунок 1.3 — Косинусная мера двух единичных векторов.

В качестве длины вектора берётся эвклидова норма:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}} \tag{1.9}$$

Так как компоненты векторов неотрицательны исходя из выбранного представления, то значение данной меры лежит в диапазоне [0,1], где значение 1 указывает на одинаковое направление векторов, а 0 на отсутствие общих слов.

Так как в данной мере используется только направление векторов, то достаточно работать с единичными векторами:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \tag{1.10}$$

Таким образом, приходим к нормализованной версии косинусной меры:

$$sim_c(\hat{\mathbf{x}},\hat{\mathbf{y}}) = \hat{\mathbf{x}} \cdot \hat{\mathbf{y}} \tag{1.11}$$

Что даёт возможность сильно упростить вычисления.

1.2.3.3 **Мера Жаккара**

Для лучшего понимания меры Жаккара сначала рассмотрим бинарный коэффициент Жаккара [9]:

$$sim_{bj} = \frac{|S_x \cap S_y|}{|S_x \cup S_y|}, \tag{1.12}$$

где S_x и S_y — множество слов в документе x и y соответственно.

Мера Жаккара определяется аналогично бинарному коэффициенту для случаев, когда вектора имеют не бинарные компоненты, а неотрицательные действительные значения [10]:

$$sim_j(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \mathbf{x} \cdot \mathbf{y}} \tag{1.13}$$

Данная мера показывает вес пересекающихся слов из обоих документов против веса слов, представленных только в одном из двух документов.

Мера Жаккара, аналогично косинусной мере, принимает значение в диапазоне [0,1], где 1 означает идентичность векторов.

1.2.3.4 **Эвристики**

В контексте задачи объединения новостей в сюжеты могут быть применены некоторые эвристики, влияющие на определение схожести новостей.

Во-первых, редко, когда новостной источник публикует две или больше новостей об одном и том же событии со схожей информацией, то есть фактически дублирует новость. Основываясь на данном предположении, можно добавлять штраф к схожести двух новостей от одного источника.

Во-вторых, новостные источники стараются максимально быстро опубликовать новость после соответствующего события, поэтому новости, образующие сюжет, публикуются приблизительно в одно время. Из этого следует, что имеет смысл добавлять штраф к схожести новостей пропорционально времени между ними.

Теперь штраф можно использовать с любой мерой схожести:

$$sim'(x,y) = sim(\mathbf{x},\mathbf{y}) \cdot (1 - penalty(x,y)), \tag{1.14}$$

где $sim(\cdot,\cdot)$ — мера схожести в диапазоне [0,1];
$penalty(\cdot,\cdot)$ — суммарный штраф в диапазоне [0,1].

1.2.4 Меры схожести кластеров

Во многих алгоритмах кластеризации предполагается объединение нескольких кластеров в один. Для этого необходимо ввести меру схожести кластеров.

1.2.4.1 По ближайшему соседу

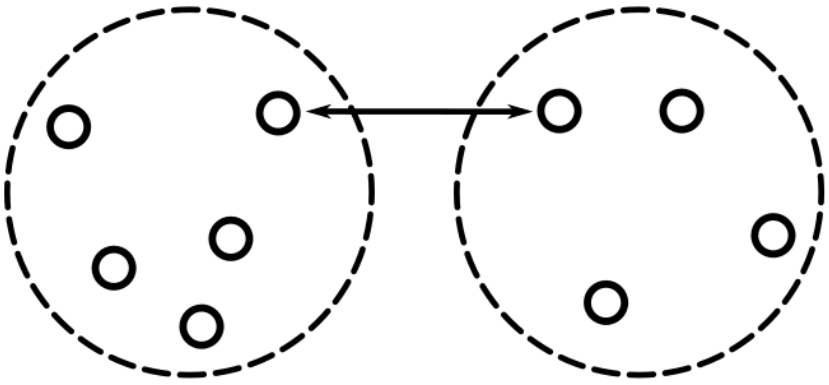


Рисунок 1.4 — Мера схожести кластеров по ближайшему соседу.

Схожесть кластеров определяется как схожесть наиболее близкой пары новостей:

$$sim_{near}(C_i,C_j) = \max_{x \in C_i, y \in C_j} sim(\mathbf{x},\mathbf{y}), \tag{1.15}$$

Однако данная мера обладает существенным недостатком: создаётся цепочка связанных кластеров, каждая связанная пара который схожа по данной мере, в отличие от крайних кластеров:

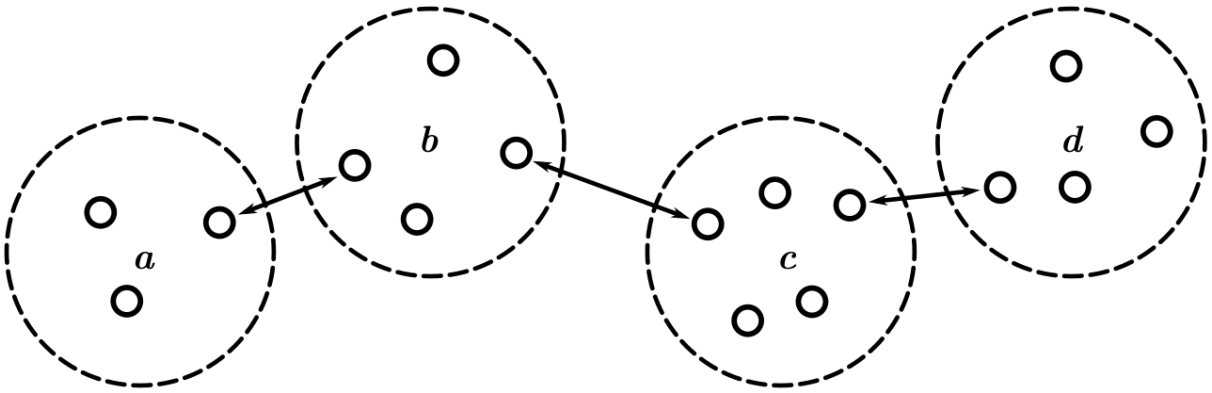


Рисунок 1.5 — Цепочка схожих кластеров.

1.2.4.2 По дальнему соседу

Аналогичным образом определяется мера по самому дальнему соседу:

$$sim_{far}(C_i,C_j) = \min_{x \in C_i, y \in C_j} sim(\mathbf{x},\mathbf{y}), \tag{1.16}$$

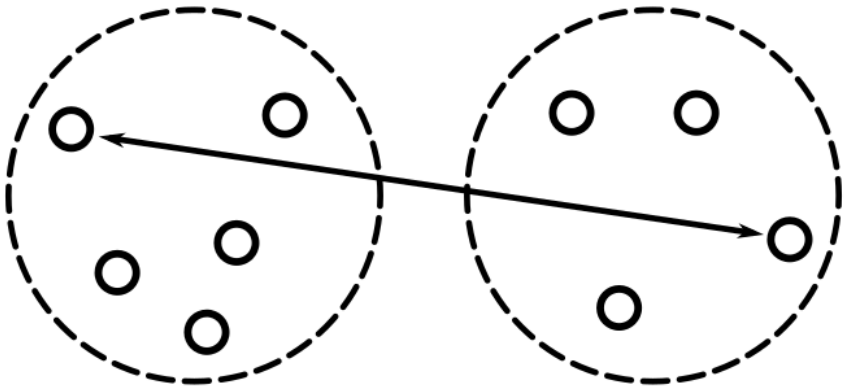


Рисунок 1.6 — Мера схожести кластеров по дальнему соседу.

Очевидным недостатком данной меры является то, что выбросы внутри кластера сильно влияют на конечный исход кластеризации [11].

1.2.4.3 По средней схожести

Мера схожести кластеров можно определять как среднюю схожесть между новостями (Unweighted Pair Group Method with Arithmetic Mean, UPGMA):

$$sim_{upgma}(C_i,C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{y}\in C_j} sim(\mathbf{x},\mathbf{y}) \tag{1.17}$$

1.2.4.4 По среднему в группе

Данная мера известна как Group-Average Agglomerative Clustering (GAAC) и похожа на меру по средней схожести в том смысле, что определяется среднюю схожесть между новостями. Однако здесь учитываются все пары новостей, включая входящие в один кластер [11]:

$$sim_{gaac}(C_i,C_j) = \frac{1}{(|C_i| + |C_j|)(|C_i| + |C_j| - 1)} \sum_{x\in C} \sum_{\substack{y\in C,\\ x\neq y}} \mathbf{x} \cdot \mathbf{y}, \tag{1.18}$$

где $C = C_i \cup C_j$.

1.2.5 Методы кластеризации

1.2.5.1 k-средних

Метод k-средних — наиболее популярный метод кластеризации. Данный алгоритм требует не только новости, но ещё и количество кластеров k.

Основная идея заключается в том, что на каждой итерации вычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем новости разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике к векторному представлению новости:

$$\mathbf{C}(\Omega) = \frac{1}{|\Omega|} \sum_{x\in\Omega} \mathbf{x}, \tag{1.19}$$

где $\mathbf{C}$ — новый центр кластера;

Ω — множество новостей, входящих в кластер.

Алгоритм завершается, когда достигается критерий останова. Обычно в качестве такого критерия берут момент, когда между итерациями состав кластеров не поменялся. При этом можно показать, что процесс завершится за конечное число итерация и зацикливание невозможно [10].

1.2.5.2 HAC

Hierarchical Agglomerative Clustering (HAC) — алгоритм иерархической кластеризации с построением иерархии снизу вверх [10]. Первоначально создаётся по кластеру на каждую новость, после чего на каждой итерации объединяются два наиболее схожих кластеров в один. Такая процедура повторяется до тех пор, пока не останется один единственный кластер, включающий в себя все остальные. В результате получаем дерево кластеров.

1.2.5.3 Разделяющий метод k-средних

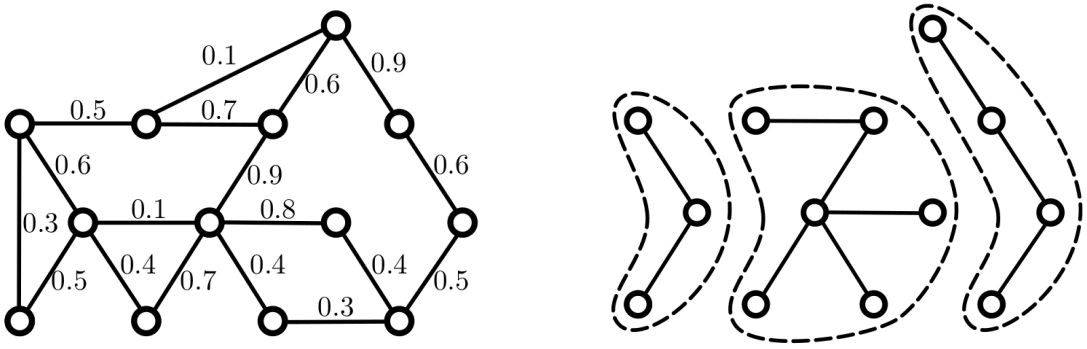
Данный метод использует два типа кластеризации: иерархическую (сверху вниз) и разделяемую [12]. Алгоритм начинает работу с создания одного кластера, включающего в себя все новости. Далее каждый кластер рекурсивно делится на k кластеров методом k-средних, пока не будет достигнуто заданное количество кластеров или схожесть новостей внутри кластера не достигнет заданной константы.

1.2.5.4 DHCA

Развитием алгоритма HAC является Algorithm (DHCA) [13]. Данные алгоритм является онлайновым, то есть обладает способностью обрабатывать новости по мере их поступления. Стоит отметить, что результат работы алгоритма не зависит от порядка обработки новостей.

Работа алгоритма начинается с создания для каждой новости кластера, после чего строится граф β-схожести (рис. 1.7а), где существу-

ют рёбра только межу кластерами схожих более, чем на β. После чего для каждой вершины оставляется только связи с максимальной схожестью. Образованные компоненты связности и будут новыми кластерами (рис. 1.7b). После чего процесс повторяется снова, причём в качестве меры схожести между кластерами используется UPGMA (1.17).



(a) Граф β-схожести кластеров.

(b) Определение кластеров.

Рисунок 1.7 — Пример работы DHCA.

1.2.5.5 **ICA**

Incremental Clustering Algorithm (ICA) — простой алгоритм кластеризации, позволяющий кластеризировать поток новостей [14].

Для каждой поступающей новости находится ближайший кластер по методу UPGMA (1.17). Если схожесть с ближайшим кластером превышает заданный порог, то новость добавляется в кластер, иначе — создаётся новый кластер, содержащий только данную новость.

Существуют модификации данного алгоритма, позволяющие значительно сократить количество вычислений схожести кластеров [14] путём разбиения пространства кластеров.

1.2.5.6 **Выбор алгоритма кластеризации**

Рассмотрим основные требования к алгоритму кластеризации согласно поставленной задачи:

а) Алгоритм должен быть онлайновым, так как мы имеем дело с потоком новостей;

б) В алгоритме не должно фиксироваться количество кластеров, так как каждая приходящая новость может образовывать новый сюжет, а значит и новый кластер;

в) Должны существовать модификации алгоритма для оптимизации вычисления схожести кластеров.

	k-ср.	разд. k-ср.	HAC	DHCA	ICA
Адаптивное кол-во кластеров	−	+	+	+	+
Иерархические кластеры	−	+	+	+	−
Онлайновый алгоритм	−	−	−	+	+
Возможность оптимизаций	+	−	−	−	+

Таблица 1.1 — Сравнение алгоритмов кластеризации.

Исходя из требований к задаче и оценки алгоритмов (1.1) имеет смысл выбрать ICA.

1.3 Ранжирование источников

Задача ранжирования источников по степени доверии к ним заключается в определении рейтинга источника по имеющейся оценке недостоверности новостей, публикуемых источником.

Рассмотрим несколько методов такого ранжирования.

1.3.1 Простое ранжирование

Поскольку публикация недостоверной новости должно вести к уменьшения рейтинга источника, то можно начислять штрафные очки каждый раз, когда источник публикует недостоверную новость, причём количество начисляемых очков пропорционально степени уверенности в недостоверности новости.

Таким образом, рейтинг источника S за промежуток времени t можно определить как

$$score_S(t) = \sum_{x \in S(t)} p_x, \tag{1.20}$$

где p_x — уверенность в недостоверности новости.

При таком определении менее достоверные источники получают выше рейтинг, чем более достоверные.

1.3.2 Ссылочное ранжирование

Для более сложного ранжирования можно использовать информацию о связи источника с другими источниками через новости. Для данного подхода недостаточно только информации о сюжетах, необходимо так же определять дубликаты и первоисточники.

Рассмотрим граф новостей (рис. 1.8). В результате обнаружения дубликатов и объединения новостей в сюжеты, получаем множество связей между новостями, которые представляются тремя видами направленных рёбер:

а) отношение «дублирует»;
б) отношение «схожий сюжет»;
в) отношение «тематическое продолжение».

С каждым ребром ставится в соответствие некоторое число, означающее силу связи.

Поскольку каждая новость так же связана с источником, из которого получена данная новость, то мы можем перейти к схожему графу для источников. Теперь, если пометить некоторую новость как фальсифицированную, то мы можем, используя установленные связи, передать часть веса остальным новостям, связанных с данной, а значит и источнику, связанному с данной новостью.

В качестве алгоритма распространения веса могут быть использованы модификации PageRank или HITS алгоритмов [15].

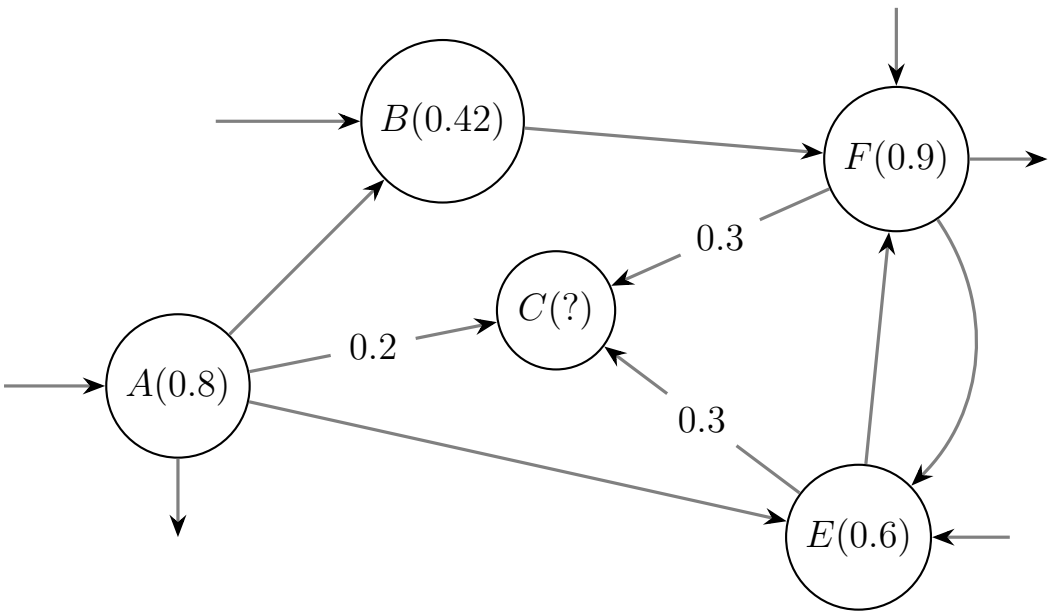


Рисунок 1.8 — Граф новостей

Из-за большей сложности (требуется так же определять дубликаты и первоисточники новостей) данный метод в работе не используется.

2 Конструкторский раздел

2.1 Декомпозиция задачи на верхнем уровне

На верхнем уровне (рис. 2.1) задача может быть сформулирована как задача ранжирования источников по степени доверия к новостям, публикуемых данными источниками. На вход разрабатываемого алгоритма поступает список новостных лент и экспертная оценка достоверности некоторых новостей. На выходе же ожидается рейтинги источников для данного временного среза новостей.

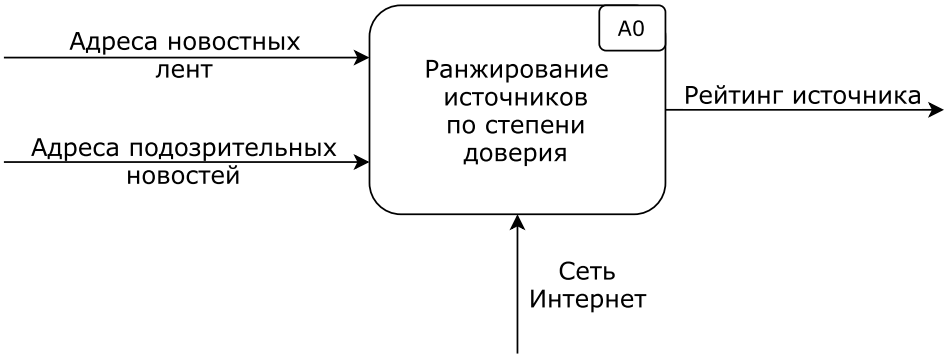


Рисунок 2.1 — Постановка задачи.

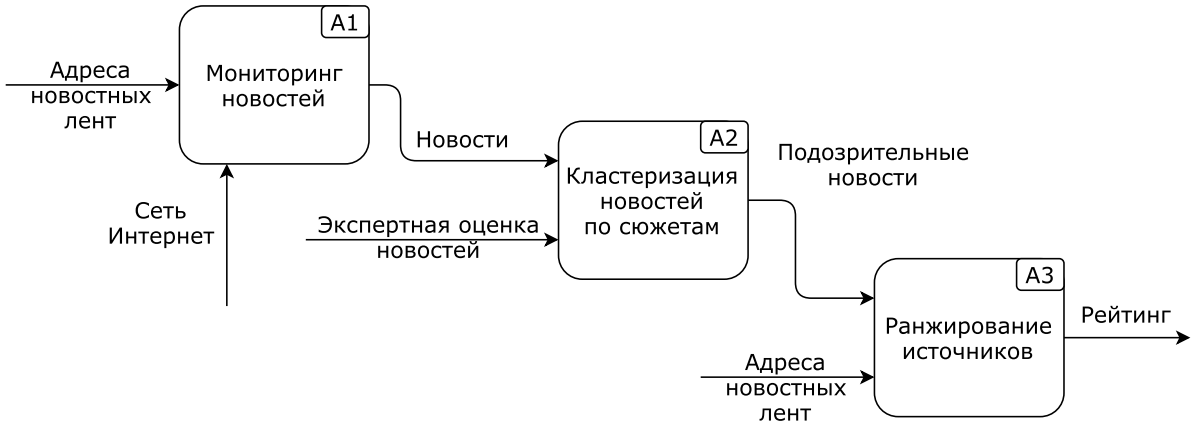


Рисунок 2.2 — Этапы решения задачи.

В результате декомпозиции выделяются три основных этапа решения задачи (рис. 2.2):

а) Мониторинг потока новостей в реальном времени. Разрабатываемая система должна регулярно проверять наличие новых новостей по

заданному набору rss/atom-лент, используя сетевой стек для доступа в глобальную сеть «Интернет»;

б) Кластеризация новостей с последующим распространением оценки экспертов (то есть информации о том, какие новости и с какой вероятностью признаются недостоверными) достоверности новостей на другие новости внутри сюжета;

в) Ранжирование источников, используя полученный список недостоверных новостей по каждому источнику.

Перейдём к рассмотрению потоков данных в системе для выделенных этапов (рис. 2.3).

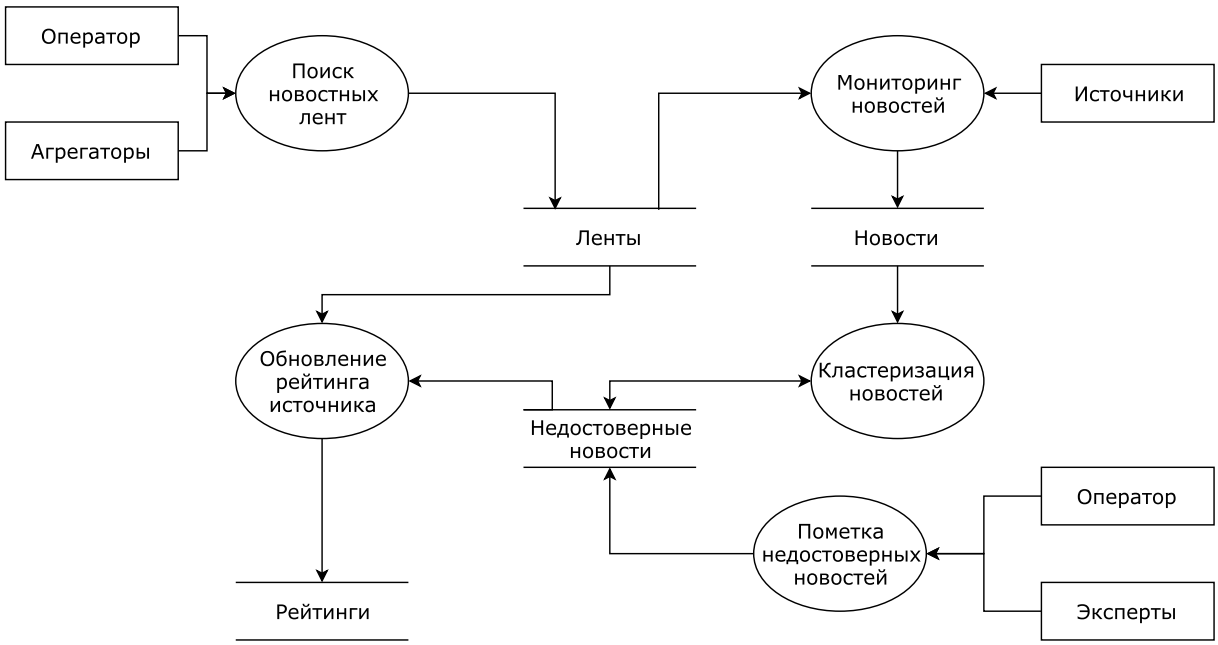


Рисунок 2.3 — Потоки данных в системе.

Типы данных в системе:

— Ленты — адреса rss/atom-лент и текущий период их обновления. В систему ленты попадают ручным добавлением со стороны оператора. Кроме того, периодически опрашиваются rss-агрегаторы (например feedly.com) на предмет появление новых популярных источников в заданных категориях (политика, экономика и другие);

— Новости — тексты новостей. В систему добавляются посредством регулярного опроса новостных лент и выкачивания новых статей со сайта источника;

— Недостоверные новости — адреса недостоверных новостей и оценки степени их недостоверности. В систему заносятся оператором и регулярно выкачиваются с экспертного сайта stopfake.org, а также в результате пометки схожих новостей;

— Рейтинги — рейтинги источников, обновляются при каждом появлении оценки очередной недостоверной новости.

2.2 Мониторинг новостей

Рассмотрим подробнее первый этап метода — мониторинг новостей (рис. 2.4) в реальном времени.

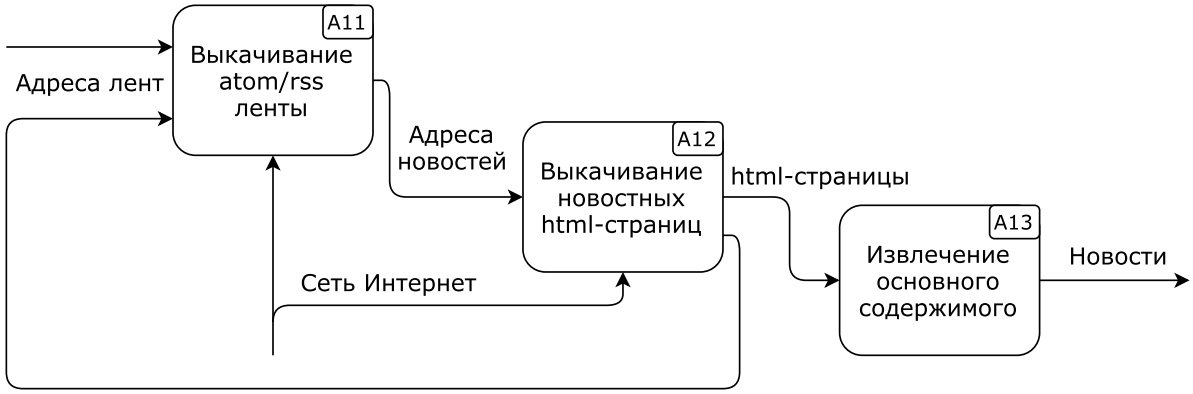


Рисунок 2.4 — Мониторинг новостей.

Задачей данного этапа является регулярное получение новых новостей по списку rss/atom-лент. Для решения данной задачи используются методы и оптимизации из раздела 1.1.

2.2.1 Выкачивание новостных лент

Вместе с новостной лентой в системе хранится информация о дате последней новости, полученной по данной ленте, а так же вычисленный текущий период обновления ленты.

Так как одна и та же новость может быть доступна из нескольких лент, то необходимо фильтровать посещённые новости. Для этого используется фильтр Блума, описанные в разделе 1.1.2.1.

Отдельной проблемой является определение периода обновления ленты, который зависит не только от частоты публикации новостей источником, но и политической ситуации и даже от времени суток.

Решить данную проблему можно следующим образом: при каждом получении ленты будем определять долю старых новостей в ней: s. Далее, определим некоторую константу p, определяющую идеальное значение такой доли. Если при очередном выкачивании ленты старых новостей стало больше, то необходимо увеличить период хождения за лентой, если же доля упала, то, наоборот, увеличить:

$$t_n = t_{n-1} \cdot (1 - trust) + \left(\frac{s}{p}\right) \cdot trust, \tag{2.1}$$

где t_n — период выкачивания ленты;

$trust$ — доверие к коэффициенту (пропорционально числу новостей).

2.2.2 **Выкачивание новостных страниц**

После выкачивания ленты и определения адресов новых новостей необходимо выкачать эти самые новости. Проблема заключается в том, что новость получается не напрямую, а только в составе html-страницы, содержащей мусор в виде рекламы, навигации и так далее.

При запросе к серверам источника учитываются ограничения, указываемые сайтом источника в стандарте исключений для роботов (см. 1.1.3).

2.2.3 **Извлечение новости из страницы**

На данном шаге из html-страницы получается непосредственно новость, используя алгоритм выделения основного содержимого со страницы, изложенный в 1.1.4.1, а так же декодирование мнемоник HTML

(см. 1.1.4.2). В результате получаем новости, которые можно использовать на следующем этапе — кластеризации.

Новости идентифицируются по ключу, получаемого в результате нормализации URL (см. 1.1.4.3).

2.3 Кластеризация

Задачей данного этапа является объединение новостей в сюжеты и распространение экспертной оценки внутри кластера (рис. 2.5).

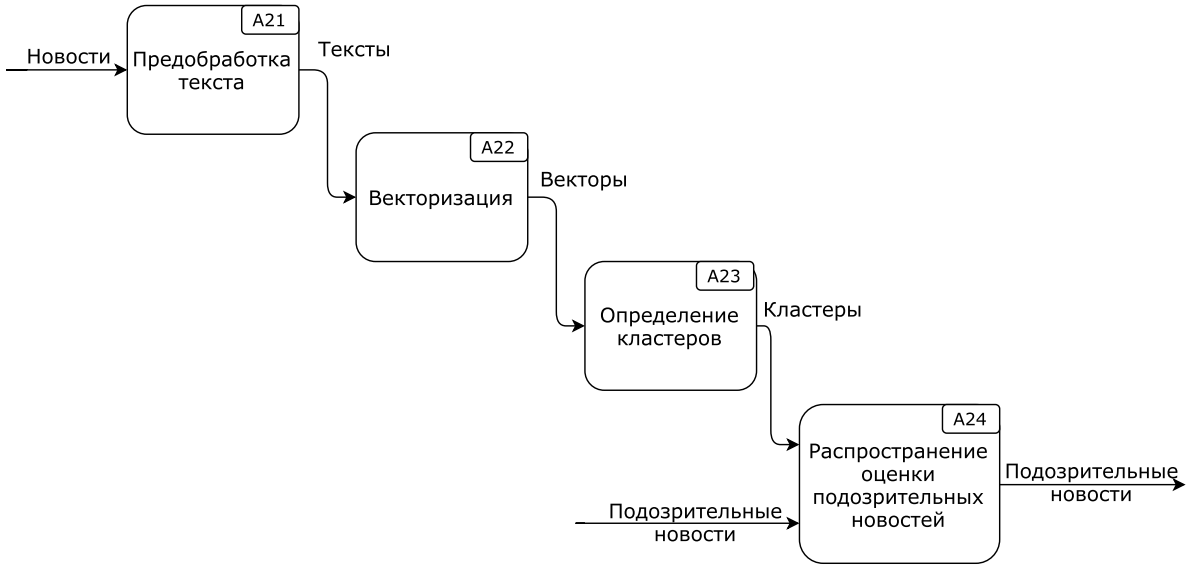


Рисунок 2.5 — Кластеризация.

Рассмотрим подробнее шаги данного этапа.

2.3.1 Предобработка

Шаг предобработки текста включает в себя удаление стоп-слов и стемминг.

2.3.1.1 Стоп-слова

Существование заведомо высокочастотных слов (таких как «an», «and», «и», «как» и др.) ведёт к раздуванию индекса и, как следствие, замедлению работы поиска. При этом их вклад в ранжирование минима-

лен. Поэтому разумно отказаться от индексирования таких слов вообще, путём занесения их в списки так называемых стоп-слов.

2.3.1.2 **Стемминг**

Стемминг —процесс нормализации слов путём выделения основы (см. 1.2.2.2). Это позволяет учитывать морфологически близкие слова как формы одно и того же слова (например, «connect» и «connected» или «чистый» и «чистая»).

Наиболее известен алгоритм стемминга Портера. Алгоритм не использует баз основ слов, а работает, последовательно применяя ряд правил отсечения окончаний и суффиксов [16].

Рассмотрим версию алгоритма для русского языка [17].

Во-первых, в слове выделяются три зоны:

RV — область слова после первой гласной. Может быть пустой, если гласных в слове нет;

R1 — область слова после первого сочетания «гласная-согласная»;

R2 — область R1 после первого сочетания «гласная-согласная».

Далее выделяются группы окончаний слов:

— Совершенного герундия («в», «вши», «вшись» после «а» или «я»; «ив», «ивши», «ившись», «ыв», «ывши», «ывшись»);

— Прилагательных («ее», «ие», «ые», «ое», «ими», «ыми», «ей», «ий», «ый», «ой», «ем», «им», «ым», «ом», «его», «ого», «ему», «ому», «их», «ых», «ую», «юю», «ая», «яя», «ою», «ею»);

— Причастных («ем», «нн», «вш», «ющ», «щ» после а и я; «ивш», «ывш», «ующ»);

— Возвратных («ся», «сь»);

— Глагольных («ла», «на», «ете», «йте», «ли», «й», «л», «ем», «н», «ло», «но», «ет», «ют», «ны», «ть», «ешь», «нно» после а или я; «ила», «ыла», «ена», «ейте», «уйте», «ите», «или», «ыли», «ей», «уй», «ил»,

«ыл», «им», «ым», «ен», «ило», «ыло», «ено», «ят», «ует», «уют», «ит», «ыт», «ены», «ить», «ыть», «ишь», «ую», «ю»);

— Существительных («а», «ев», «ов», «ие», «ье», «е», «иями», «ями», «ами», «еи», «ии», «и», «ией», «ей», «ой», «ий», «й», «иям», «ям», «ием», «ем», «ам», «ом», «о», «у», «ах», «иях», «ях», «ы», «ь», «ию», «ью», «ю», «ия», «ья», «я»);

— Превосходных («ейш», «ейше»);

— Словообразовательных («ост», «ость»);

— Адъективированных (определяется как прилагательное или причастие + прилагательное окончание).

При поиске окончания из всех возможных выбирается наиболее длинное. Например, в слове «величие» выбираем окончание «ие», а не «е».

Все проверки производятся над областью RV. Так, при проверке на совершенный герундий предшествующие буквы «а» и «я» также должны быть внутри RV. Буквы перед RV не участвуют в проверках вообще.

а) Найти окончание совершенного герундия. Если оно существует — удалить его и завершить этот шаг. Иначе, удаляем возвратное окончание, если существует. Затем по порядку удаляется адъективированное окончание, глагольное окончание, окончание существительного. Как только одно из них найдено — шаг завершается;

б) Если слово оканчивается на «и» — удалить «и»;

в) Если в R2 найдётся словообразоваельное окончание — удалить его.

г) Возможен один из трёх вариантов:

1) Если слово оканчивается на «нн» — удалить последнюю букву;

2) Если слово оканчивается на превосходное окончание — удаляем его и снова проверяем «нн»;

3) Если слово оканчивается на «ь» — удалить его.

Аналогично существуют версии алгоритма для многих европейских языков, однако в данной работе используется только русский и английский языки.

2.3.2 Векторизация

Следующим шагом в этапе кластеризации является векторизация, то есть получение векторного представление новости из текстового.

Для оставшихся слов после предобработки текста вычисляются значения характеристики $tf\text{-}idf$, согласно формулам, представленных в разделе 1.2.1.

2.3.3 Определение кластеров

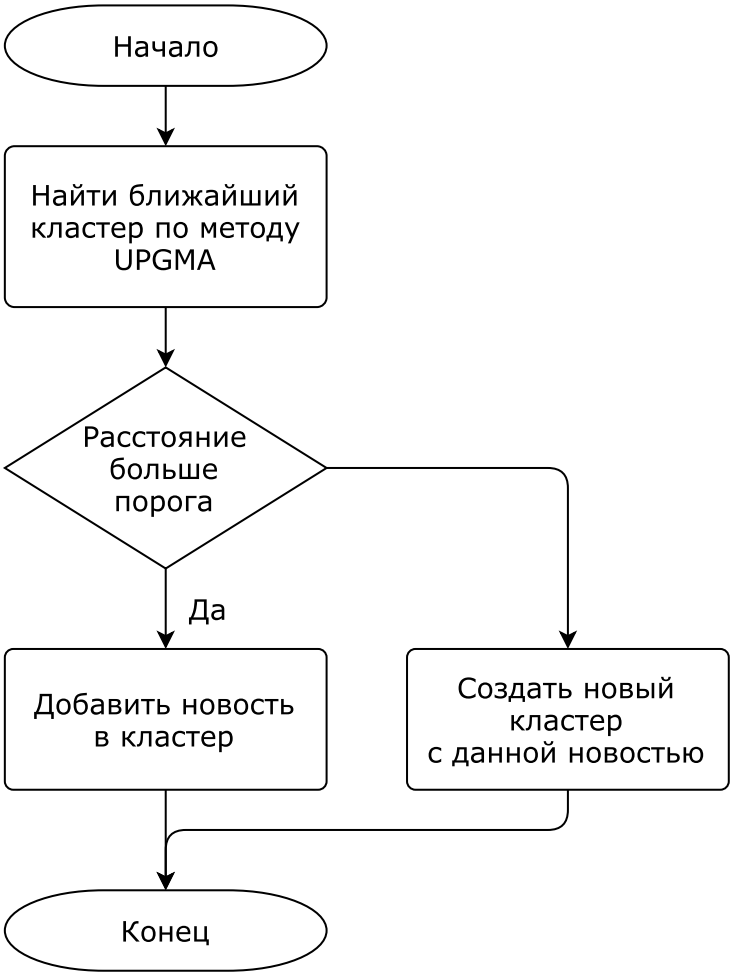


Рисунок 2.6 — Блок-схема ICA.

На данном шаге проводится непосредственно кластеризация — объединение новостей в кластеры. Для этого используется алгоритм ICA (рис. 2.6), выбранный в результате анализа, проведённого в разделе 1.2.5.6.

2.3.4 Распространение оценки экспертов

Последним шагом на данном этапе является распространение оценки экспертов. На входе шага имеем множество кластеров и множество экспертных оценок.

Для каждой экспертной оценки помечаем все новости в кластере как недостоверные с оценкой, пропорциональной расстоянию между экспертной новостью и рассматриваемой:

$$s_y = s_x \cdot sim(x,y), \tag{2.2}$$

где s_x и s_y — вероятностные оценки недостоверности новостей x и y, содержащихся в одном кластере.

2.4 Ранжирование

Заключительным этапом работы системы является ранжирование источников по степени доверия к новостям, публикуемым ими (рис. 2.7).

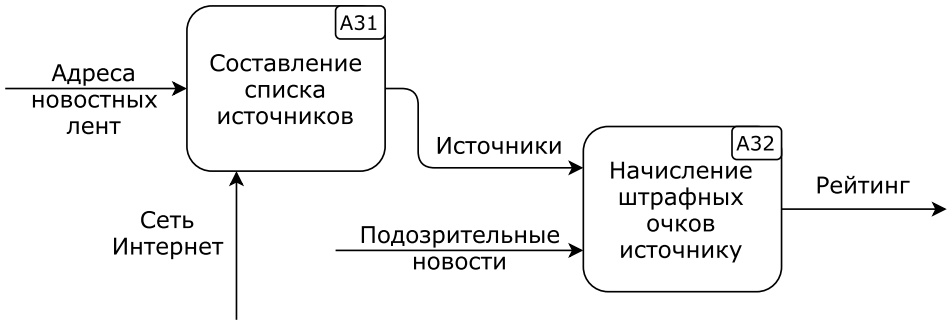


Рисунок 2.7 — Определение рейтинга.

2.4.1 **Составление списка источников**

Список источников может быть получен из новостной ленты, поскольку каждая rss/atom-лента обязана содержать ссылку на сайт источника.

2.4.2 **Начисление штрафных очков источнику**

Данный шаг получает на вход список источников и список всех недостоверных статей с их оценками, полученными системой (либо от экспертов, либо в результате распространения экспертной оценки на этапе кластеризации).

Для формирования рейтинга используется подход, описанный в разделе 1.3.1.

3 Технологический раздел

3.1 Подход к архитектуре системы

Если рассмотреть потоки данных в системе (рис. 2.3), то можно заметить несколько особенностей рассматриваемого решения:

— Все процессы связаны друг с другом через хранилища данных;
— Система должна обладать большой пропускной способностью и легко масштабироваться;
— Возможно появление новых процессов и потоков данных;

Для систем со схожими требованиями хорошо подходит микросервисная архитектура [18]: каждый процесс верхнего уровня представляется как отдельный микросервис, решающий поставленную задачу. В данную систему включены пять микросервисов:

— `scout` — поиск новостных лент с feedly.com;
— `raider` — мониторинг новостей;
— `soothsayer` — мониторинг экспертной оценки с сайта stopfake.org;
— `compounder` — кластеризация новостей;
— `rater` — ранжирование источников;

Микросервисы могут разрабатываться отдельными командами и с использованием разных технологий, в том числе разных языков программирования.

Для решения задачи коммуникации между микросервисами сегодня существуют три основных подхода [19]:

— Прямое взаимодействие посредством REST API или RPC;
— Совместно используемые базы данных;
— Использование брокера сообщений.

Поскольку процессы не связаны напрямую друг с другом, а совместно используемые базы данных сильно повышают связанность микросервисов, то подходящим решением является использование брокера

сообщений — специальной распределённой системы, занимающейся доставкой сообщений от поставщиков к подписчикам.

3.2 Брокер сообщений и персистентное хранилище

К сожалению, использование лишь брокера сообщений является недостаточным в данной системе, так как необходимо хранить новости на протяжении нескольких недель, чтобы иметь возможность запускать новые версии микросервисов (например, кластеризатора).

Кроме того, есть необходимость в постоянном хранении таких данных, как рейтинги источников и информации о лентах этих источников. То есть необходимо некоторое постоянное key-value хранилище.

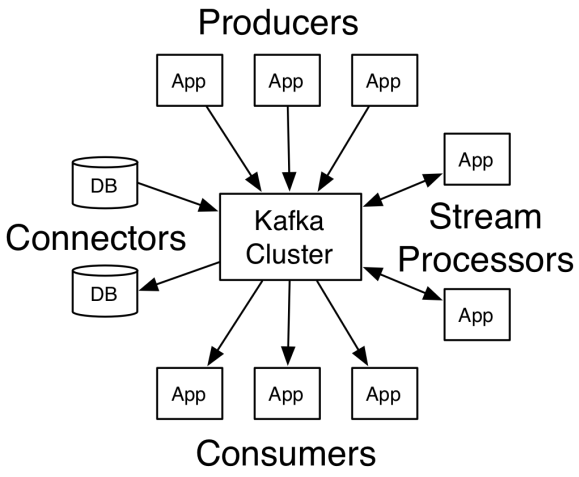
Классическими решением является использование баз данных для каждого микросервиса, который нуждается в этом. Однако в последнее время набирают оборот системы, сочетающие в себе функции брокера сообщений и персистентных хранилищ данных, например kafka и nsq.

В данной работе принято решение использовать kafka в качестве такой системы.

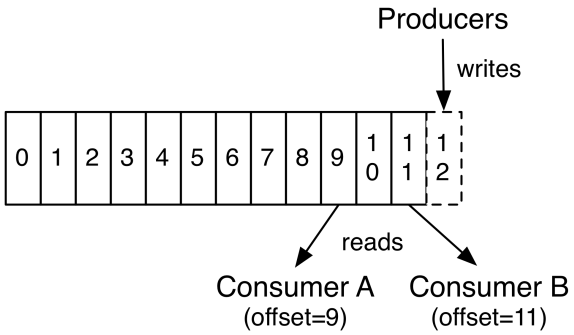
Рассмотрим подробнее принцип работы kafka. Кластер kafka обслуживает запросы от различных приложений, которые могут являться как потребителями сообщений, так и поставщиками (рис. 3.1a).

Все хранилища данных, представленные на диаграмме потоков данных системы (рис. 2.3) представляются как темы в терминах kafka. Каждая тема — буфер, хранящий все поступающие сообщения. Каждый потребитель получает сообщения по смещению в этом буфере (рис. 3.1b). Такой буфер полностью хранится на файловой системе, а производительность считывания достигается за счёт последовательного чтения и использования операционной системой файлового кеша.

Распределённость системы получается благодаря разделению буфера на несколько разделов [20], каждый из которых реплицируется внутри kafka кластера (рис. 3.2).



(a) Кластер kafka.



(b) Буфер темы с сообщениями.

Рисунок 3.1 — Apache Kafka.

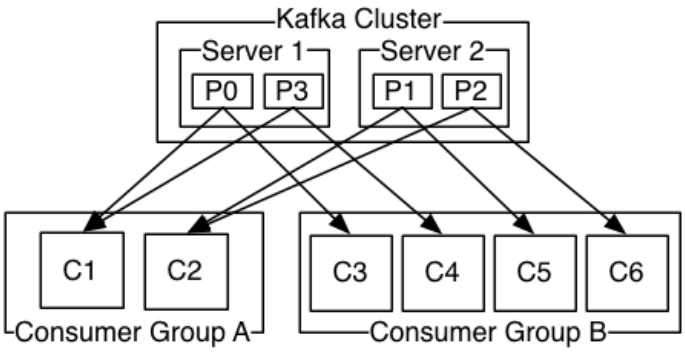


Рисунок 3.2 — Репликация буферов внутри кластера kafka.

3.3 Язык программирования и инструменты разработки

Поставленная задача требует частого взаимодействия с I/O: HTTP, работа с брокером сообщений. Из этого следует необходимость в асинхронной архитектуре микросервисов. Кроме того, основные микросервисы, для сбора новостей и кластеризации, требовательны к вычислительным ресурсам, поэтому необходимо использовать компилируемый язык программирования с производительностью сравнимой с Си. Также хотелось избежать накладных расходов на работу сборщика мусора. Поэтому в качестве языка программирования был выбран Rust.

В качестве редактора кода был использован NeoVim на ОС Arch Linux. Сборка проекта осуществляется пакетным менеджером cargo — стандартным инструментом для разработчиков на Rust. В качестве системы контроля версий использовался git.

Сама же система запущена на серверах, предоставляемых digitalocean.com.

3.4 **Тестирование**

Для наиболее сложных компонентов, входящих в систему, реализовано модульное тестирование — тестирование отдельного модуля для проверки корректности его работы в штатных и исключительных ситуациях. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии.

Наиболее требовательным к тестированию компонентом является модуль выделения основного содержимого из html-документа. Для тестирования данного модуля использовался набор системных тестов, разработанный сотрудниками mozilla для аналогичного модуля на js.

4 Экспериментальный раздел

4.1 Постановка эксперимента

Необходимо оценить влияние меры схожести (см. 1.2.3) при кластеризации новостей.

Для оценки качества кластеризации будем использовать набор данных, собранный с ресурса «Яндекс.Новости»[1] в категориях «Политика», «Финансы», «Экономика» и «В мире».

К сожалению, ресурс не предоставляет текст самих новостей, а только ссылки и небольшое описание, поэтому необходимо воспользоваться алгоритмом извлечения содержимого (см. 1.1.4.1).

Поскольку данный алгоритм не идеален и не всегда верно определяет основное содержимое, то из полученного набора данных были удалены все новостные статьи короче 150 символов, в результате чего был получен набор из 2005 новостей, относящихся к 263 кластерам.

Кластеры, собранные с ресурса «Яндекс.Новости» будем называть образцовыми и оценивать качество кластеризации относительно них. В качестве алгоритма кластеризации будем использовать ICA (см. 2.3.3).

4.2 Метрики качества кластеризации

В качестве метрик качества кластеризации применяют F-меру, чистоту и энтропию [21].

Для определения F-меры необходимо сначала определить понятие точности и полноты.

Точность определяется как

$$P(c,o) = \frac{|c \cap o|}{|c|}, \tag{4.1}$$

где c — множество полученных объектов;

o — множество релевантных объектов.

[1] https://news.yandex.ru/export.html

Полнота же определяется как

$$R(c,o) = \frac{|c \cap o|}{|o|} \tag{4.2}$$

Для объединения точности и полноты в одну метрику используют F_1-меру:

$$F_1(c,o) = \frac{2 \cdot P(c,o) \cdot R(c,o)}{P(c,o) + R(c,o)} \tag{4.3}$$

Так как мы имеем дело со множеством кластеров, необходимо объединить показатели данной метрики в результирующую [21]:

$$F = \sum_{o \in O} \frac{|o|}{n} \max_{c \in C} F_1(c,o), \tag{4.4}$$

где C — множество кластеров, сформированных системой;

O — образцовое множество кластеров;

n — количество кластеризуемых объектов.

Для оценки доли корректно кластеризованных объектов используют метрику чистоты [22]:

$$Purity = \frac{1}{n} \sum_{o \in O} \max_{c \in C} |o \cap c| \tag{4.5}$$

Значение чистоты близкое к 0 указывает на плохую кластеризацию, в то время как идеальная кластеризация имеет значение 1.

А для оценки распределения объектов внутри вычисленного кластера относительно образцовых используют метрику энтропии:

$$Entropy = -\frac{1}{\log k} \sum_{o \in O} \frac{1}{n} \sum_{c \in C} |o \cap c| \cdot \log P(c,o), \tag{4.6}$$

где k — количество кластеров в образцовом множестве.

Низкое значение энтропии показывает, что вычисленные системой кластеры не содержат объектов из множества разных образцовых кластеров.

4.3 **Результаты**

Сравним меры схожести новостей, представленные в 1.2.3.

Мера схожести	F-метрика	Чистота	Энтропия
Эвклидово расстояние	0.89	0.88	0.105
Косинусная мера	0.92	0.93	0.081
Мера Жаккара	0.91	0.92	0.082
Эвклидово расстояние со штрафами	0.89	0.89	0.104
Косинусная мера со штрафами	0.93	0.94	0.080
Мера Жаккара со штрафами	0.92	0.92	0.082

Таблица 4.1 — Сравнение качества кластеризации для разных мер схожести.

Исходя из результатов, представленных в таблице 4.1 можно сделать вывод, что лучшей мерой схожести новостей среди представленных является косинусная мера со штрафами.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был реализован программный продукт, полностью отвечающий требованиям, изложенным в техническом задании, а именно система мониторинга новостей в реальном времени с последующим ранжированием источников по степени доверия к статьям, публикуемым ими.

В течение выполнения проекта была проанализирована предметная область, разработан метод ранжирования источников, проведён анализ существующих методов для каждого этапа разработанного метода, позволяющих поставленные задачи, на основе проведённого анализа выбраны те методы, которые наиболее подходят для решения задачи, исследована применимость разработанного метода, разработано программное обеспечение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Broder Andrei, Mitzenmacher Michael, Mitzenmacher Andrei Broder I Michael. Network Applications of Bloom Filters: A Survey // Internet Mathematics. — 2002. — P. 636–646.

2. Pomikálek Jan. Removing boilerplate and duplicate content from web corpora : Ph. D. thesis / Jan Pomikálek ; Masaryk university, Faculty of informatics, Brno, Czech Republic. — 2011.

3. Kohlschütter Christian, Fankhauser Peter, Nejdl Wolfgang. Boilerplate Detection Using Shallow Text Features // Proceedings of the Third ACM International Conference on Web Search and Data Mining. — ACM, 2010. — P. 441–450.

4. Pant Gautam, Srinivasan Padmini, Menczer Filippo. Crawling the Web // In Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Edited by M. Levene and A. Poulovassilis. — Springer-Verlag, 2004. — P. 153–178.

5. Frakes William B, Baeza-Yates Ricardo. Information retrieval: data structures and algorithms. — 1992.

6. Huang Anna. Similarity measures for text document clustering // Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand. — 2008. — P. 49–56.

7. Strehl Alexander, Ghosh Joydeep, Mooney Raymond. Impact of similarity measures on web-page clustering // Workshop on artificial intelligence for web search (AAAI 2000). — Vol. 58. — 2000. — P. 64.

8. Zhong Shi. Efficient online spherical k-means clustering // Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on / IEEE. — Vol. 5. — 2005. — P. 3180–3185.

9. Strehl A, Ghosh J. Relationship-based clustering and cluster ensembles for high-dim. data : Ph. D. thesis / A Strehl, J Ghosh ; PhD thesis (May 2002).

10. Introduction to information retrieval / Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze et al. — Cambridge university press Cambridge, 2008. — Vol. 1.

11. Zhao Ying, Karypis George, Fayyad Usama. Hierarchical clustering algorithms for document datasets // Data mining and knowledge discovery. — 2005. — Vol. 10, no. 2. — P. 141–168.

12. A comparison of document clustering techniques / Michael Steinbach, George Karypis, Vipin Kumar et al. // KDD workshop on text mining / Boston. — Vol. 400. — 2000. — P. 525–526.

13. Gil-García Reynaldo, Badía-Contelles José M, Pons-Porrata Aurora. Dynamic hierarchical compact clustering algorithm // Iberoamerican Congress on Pattern Recognition / Springer. — 2005. — P. 302–310.

14. Lönnberg Marcus, Yregård Love. Large scale news article clustering. — 2013.

15. Segaran Toby. Programming Collective Intelligence. — O'Reilly, 2007. — ISBN: 9780596529321.

16. Porter M. F. Readings in Information Retrieval. — Morgan Kaufmann Publishers Inc., 1997. — P. 313–316.

17. Porter M. F. Russian stemming algorithm. — 2007. — Access mode: `http://snowball.tartarus.org/algorithms/russian/stemmer.html`.

18. Kleppmann Martin. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. — " O'Reilly Media, Inc.", 2017.

19. Neuman Sam. Building Microservices: Designing Fine-Grained Systems. — O'Reilly Media, 2015.

20. Wang Chengwei, Rayan Infantdani Abel, Schwan Karsten. Faster, larger, easier: reining real-time big data processing in cloud // Proceedings of the Posters and Demo Track / ACM. — 2012. — P. 4.

21. Recent developments in document clustering : Rep. / Technical report, Computer Science, Virginia Tech ; Executor: Nicholas O An-

drews, Edward A Fox : 2007.

22. Deepa M, Revathy P, Student PG. Validation of Document Clustering based on Purity and Entropy measures // International Journal of Advanced Research in Computer and Communication Engineering. — 2012. — Vol. 1, no. 3. — P. 147–152.