Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра информационно-аналитический систем

Чуриков Никита Сергеевич

Суммаризация групп в социальных сетях

Выпускная квалификационная работа

Научный руководитель:
к. ф.-м. н., доцент Графеева Н. Г.

Рецензент:
Директор департамента управления информационными потоками Яковлев П. А.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Sub-Department of Analytical Information System

Nikita Churikov

News resources summarization in social networks

Graduation Thesis

Scientific supervisor:
Assistant Professor Natalia Grafeeva

Reviewer:
Director of information flow department Pavel Yakovlev

Saint-Petersburg
2019

Оглавление

Введение

В современном мире создается все больше и больше информации, которую мы можем потреблять. Новости, статьи, юмор постоянно меняются и создаются людьми. При таком потоке информации появляется потребность в инструментах, способных давать как можно больше информации с минимальными потерями.

При чтении новостей люди, как правило, не идут дальше новостных заголовков [6], для популярных технических статей создают краткие описания описывающие их достижения и основные моменты [30, 28], а визуальный контент нередко подчиняется единому шаблону.

В данной работе мы показываем, как используя современные достижения в области анализа данных можно извлекать полезную информацию из новостных ресурсов в социальной сети вконтакте [31], и приводим обоснование выбора решений, основываясь на соответствующих метриках.

1. Постановка задачи

Мы поставили перед собой задачу создать систему, которой бы можно было передавать ссылку на новостной ресурс в социальной сети вконтакте, а на выходе получать его краткое описание. В рамках работы мы ограничились новостными ресурсами с высоким содержанием текста.

На Рис. 1 мы показываем как работает наше решение "с высоты птичьего полета". Процесс имеет следующий вид, мы получаем ссылку на группу ВК, затем через API вконтакте получаем информацию о группе и оцениваем количество текста в ней. если содержание текста низкое, то мы говорим, что это ресурс с доминирующим медиа-контентом, если в группе мало предложений, но достаточно слов, то решается задача выделения ключевых слов, если в группе высокое содержание предложений, то мы решаем задачу генерации заголовков.

Таким образом, с алгоритмической точки зрения, задача суммаризации новостного ресурса была рассмотрена нами как две подзадачи:

1. Извлечение ключевых слов, присущих данному источнику информации;

2. Сжатие новостей, используя автоматическое создание заголовков.

Через извлечение данной информации мы хотим добиться эффекта "чтения по диагонали".

Для оценки качества наших алгоритмов, мы воспользовались открытыми датасетами для генерации заголовков и извлечению ключевых слов.

2. Обзор литературы

Задача сжатия текста с малой потерей смысла и сохранением возможности его прочтения имеет название задачи *суммаризации*. При этом, есть два концептуальных подхода к решению: экстрактивный, когда для создания краткого содержания извлекаются целые куски текста вплоть до предложений, и абстрактивная, где в кратком содержании могут быть слова, которых не было в исходном тексте.

Статья Браславского [4] дает хорошую ретроспективу о том, как подходили к задаче суммаризации в 2000-х годах. Помимо того, что суммаризация чаще называлась реферированием, виден тренд применения метода машинного обучения SVM, а задача суммаризации еще не была сведена к задаче генерации последовательности, а применялись попытки решать ее посредством прямого извлечения кусков текста.

Сегодня задача суммаризации рассматривается как задача перевода одной последовательности в другую. Долгое время применялись методы Seq2Seq с различными модификациями [18], поскольку алгоритмы на основе Seq2Seq плохо работают с длинными последовательностями.

Однако в последние несколько лет был придуман подход с вниманием, где выход кодировщика входной последовательности не только один накопленный вектор контекста, а также для декодирования обучается слой "внимания", который создает вектор контекста для текущего предсказания.

В данной работе мы рассматриваем подзадачу суммаризации, извлечение заголовков. При исследовании абстрактивной генерации заголовков, мы отталкивались от статьи Вконтакте, посвященной данной проблеме [7]. Ими предлагается применять нейронные сети с архитектурой Transformer и предобработкой Byte pair encoding (BPE) [22]. Однако в задаче абстрактивной генерации заголовков существуют дебаты на тему того, что использовать в качестве входа модели. Поскольку долгое время SOTA были модели с архитектурой encoder decoder, то было невозможно использовать длинные входные последовательности. Потому авторы статьи [18] исследуют различные подходы по предвари-

тельному извлечению "Topic sentence", которое нейронная сеть должна дальше обработать. Это предложение, как говорят авторы, в идеальном случае, должна отвечать на 5W1H. Но достаточно ответов на "что, кто, когда".

Несмотря на значительный прогрессы в различных областях за последние несколько лет, задача извлечения значимых *ключевых слов* все еще не решена, поскольку эффективность существующих алгоритмов все еще далека от удовлетворительной в отличии от многих других ключевых областей computer science. Большинство традиционных подходов используют обучение с учителем, которое в значительной степени зависит от наличия доступа к размеченным данным для обучения. Один из первых подходов для выделения ключевых слов был предложен Turney [24], который разработал специально разработанный алгоритм под названием GenEx.

Подавляющее большинство подходов, разработанных до сих пор основываются на обучении с учителем для выбора релевантных ключевых слов. Возможно, наиболее распространенной реализацией такого подхода является KEA [11], который использует алгоритм Naive Bayes для извлечения ключевых слов. Несмотря на их часто превосходящую эффективность, основным ограничением алгоритмов обучения с учителем является их относительно длительный процесс обучения.

Это отличает алгоритмы обучения без учителя, которые могут быстро применяться к документам на разных языках или или из разных предметных областей и выдавать результат за короткий промежуток времени. В статьях [23, 27] описывается YAKE, алгоритм извлечения ключевых слов, который основан на статистических признаках текста, извлеченных из одного документа, для идентификации и ранжирования наиболее важных ключевых слов. Ключевые аспекты YAKE! состоят в том, что не требуется обучение на конкретном наборе документов, и, следовательно, может быть легко применен к отдельным текстам, независимо от наличия корпуса, словаря или любой внешней коллекции. Алгоритм не использует ни извлечение именованных сущностей, ни разметку частей речи, что делает его независимым от языка, за

исключением использования различных, но статических списков стопслов для каждого языка. Данные статьи являются оригинальными статьями по данному алгоритму и описывают принцип его работы, что является крайне необходимым для создания собственных реализаций.

Помимо YAKE, state of the art решениями в задаче извлечения ключевых слов все еще считается TextRank, графовый алгоритм экстрактивной суммаризации, который строит граф расстояний между токенами текста. Ключевой статьей при ознакомлении с работой данного алгоритма является оригинальная статья Рады и Тарау [15]. Данная статья описывает PageRank, алгоритм Google для рекомендации веб страниц, который лег в основу TextRank, и объясняет, почему алгоритмы на основе графовых голосований подходят для решения задачи извлечения ключевых слов.

3. Система анализа групп

В этой секции описано то, что из себя представляет система анализа групп с точки зрения разработчика. Какие требования были ей были поставлены и какие технические решения были использованы.

При этом алгоритмическая часть будет рассмотрена в следующих секциях.

3.1. Актульность решения

Рассмотрим теперь то, насколько актуальна данная проблема и какое место занимает предлагаемое решение среди существующих систем анализа групп в социальных сетях.

На рынке уже существует большое множество сервисов дающих возможность проанализировать группу на предмет возможного размещения рекламного поста [29]. Однако несмотря на такое разнообразие сервисов, их функционал слабо отличается друг от друга. Как правило, предоставляется возможность следить за аудиторией (ее полом, численностью, возрастом), частотой выпускаемого материала, частотой выпуска материала и его типом. Однако никто из игроков не производит анализа содержания групп, что кажется упущением возможности выделиться на рынке.

Наличие анализа содержания группы могло бы повысить скорость создания рекламы за счет уменьшения времени, необходимого для того, чтобы вникнуть в идею группы.

Причинами того, что создатели подобных сервисов не спешат с созданием таких услуг мы видим следующее:

- Необходимо нанимать новых сотрудников, что влечет дополнительные расходы;
- Спрос на подобный функционал может быть низок из-за того, что в случае успеха подобного функционала, может снизиться спрос на специалистов в области SMM.

- Поскольку спрос низкий, то данное предложение не предлагается.

3.2. Описание системы

Имея ввиду причины отсутствия подобного функционала, мы считаем, что есть смысл создать его в виде черной коробки, которую можно будет подключать к уже существующим сервисам. В этой черной коробке использовать алгоритмы, которые показали отличные результаты на академических датасетах и рекомендовать их как базовые, однако предоставить возможность выбора алгоритмов разработчикам.

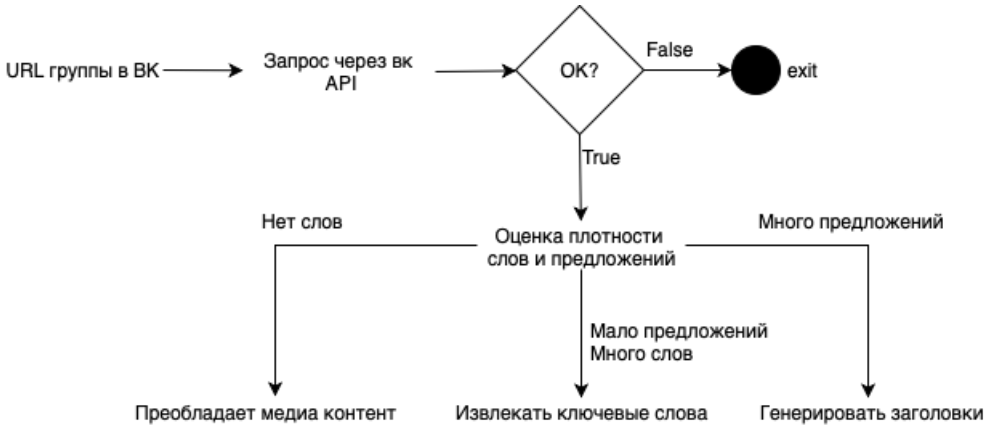

Рис. 1: Принцип работы системы.

Таким образом стояли следующие задачи:

- Проанализировать существующие алгоритмы выделения ключевой информации из текстов;
- Предоставить возможность воспользоваться этими решениями как черным ящиком, создав интерфейсы для разработчиков:
  - Библиотеку на python
  - REST API, чтобы была возможность использовать из других языков программирования

3.3. Реализация системы анализа групп

На Рис. 1 приведено описание того, как можно построить фронтенд для взаимодействия с нашей реализацией.

На Рис. 2 показано то, как выглядит сама техническая реализация. Мы реализовали эту систему в виде четырех микросервисов, где каждый из них решает свою задачу: выделяет ключевые слова, генерирует заголовки, производит подсчет статистик о группе, при этом, первый запрос о группе выполняет загрузку новых данных делая запрос во вконтакте и сохраняет новые данные базу данных.

Реализацию подсчета статистик и загрузки данных можно найти в нашем репозитории [1], реализация REST API для генерации заголовков использует API библиотеки OPENNMT [16], API выделения ключевых слов можно найти в нашей библиотеки keyverbum [2].

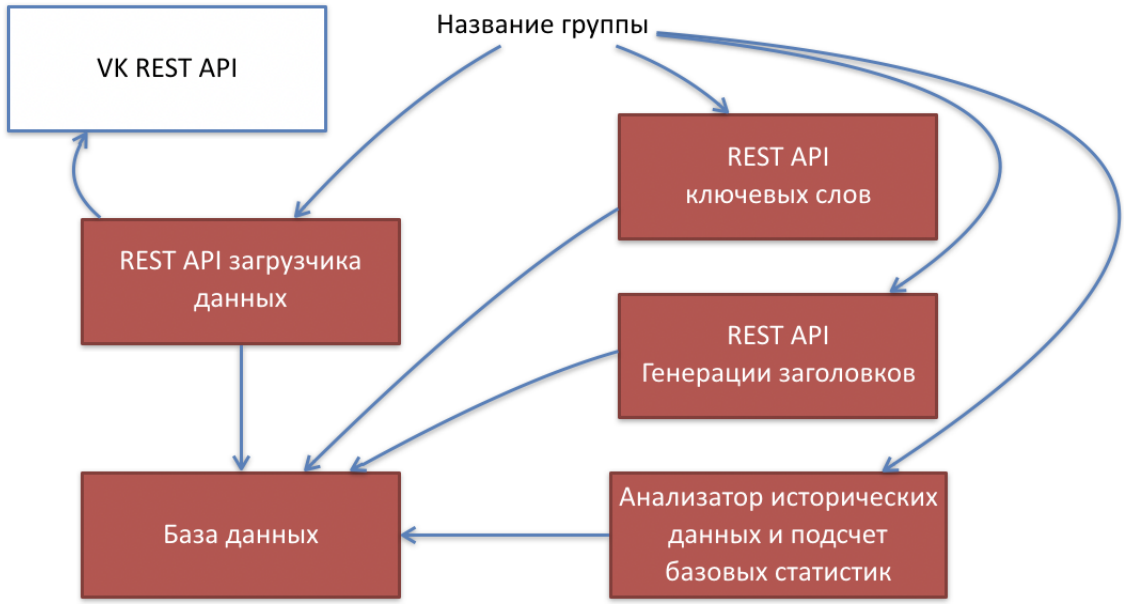

Рис. 2: Техническая визуализация системы.

Рассмотрим дальше использованные технические решения. В качестве базы данных мы использовали MongoDB [3], поскольку данная база

[1] https://github.com/imemedb/media_resources_summarization
[2] https://github.com/imemedb/keyverbum
[3] https://www.mongodb.org

данных не имеет фиксированной структуры данных и при этом способна возвращать результаты запросов из базы очень быстро [4]. Помимо простота настройки и использования делают ее отличным выбором для opensource решения.

В самой базе данных хранение информации о группах организованно следующим образом: существует только одна коллекция с постами из групп. Когда происходит запросы в vk api[5] о новых записях, но пришедшие записи сравниваются по Id. Если эта запись уже присутствует в базе, то она заменяется на более новую, поскольку информация о ней более актуальная и в ней содержится обновленная информация о количестве лайков и репостов, информация, которая нужна для анализа активности группы.

Для создания микросервисов с Рис. 2 мы использовали docker [6], поскольку это сейчас лучшее решения для создания независимых изолированных виртуальных окружений.

А анализаторе групп мы предоставляем достаточно базовый набор подсчета информации, вроде подсчета постов с момента начала слежки за группой, или количества лайков и расчета статистик над этой информацией, но полезный функционал, который мы предоставляем состоит в том, что мы даем возможность посчитать количество медиа информации в группе, количество используемых предложений и слов. Данная информация может быть полезна для дифференциации групп с целью дальнейшего извлечения либо ключевых слов либо создания заголовков. Рис 3

[4]https://stackoverflow.com/questions/49886279/mongo-query-take-a-long-time-how-make-it-more-fast
[5]https://vk.com/dev/methods
[6]https://www.docker.com

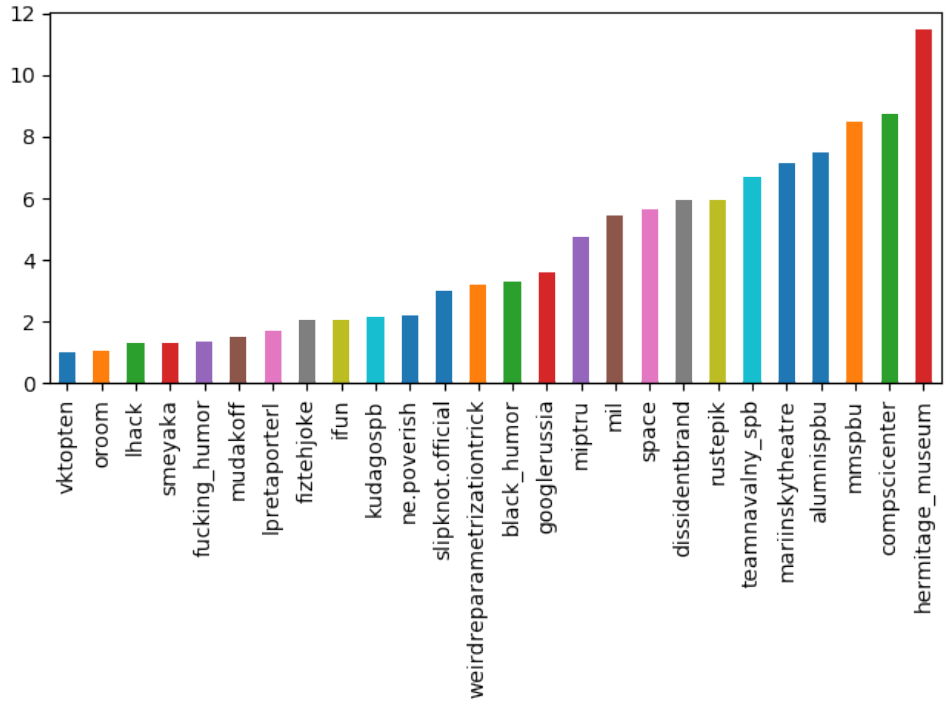

Рис. 3: Количество предложений в различных группах.

4. Алгоритмы, использованные в работе

Нами были использованы как классические подходы, так и новые, основанные на нейронных сетях. В следующих секциях мы опишем их основные принципы, а также приведем ссылки на их реализации.

4.1. Суммаризация текста

Для суммаризации текста мы воспользовались алгоритмом экстрактивной суммаризации основанном на TextRank [26, 20, 15], и моделью трансформера [2], обученной на датасете РИА новостей [7]. Для предобработки данных модели трансформера мы использовали byte pair encoding [22]. Помимо этого мы извлекали первое предложение из новости. Для TextRank и извлечения первого предложения не требуется обучающая выборка, что делает их очень удобными в использовании. При этом, исследования показывают, что в задаче генерации заголовков, первое предложение в новости – это очень сильный бэйзлаин [7],

который трудно побить как экстрактивной, так и абстрактивной суммаризацией.

4.1.1. Baseline

В качестве бэйзлаина в задаче генерации заголовков используется первое предложение новости. Именно им мы и воспользовались и отталкивались от него.

4.1.2. TextRank

TextRank является является адаптацией идеи алгоритма PageRank [17] с задачи рекомендации страниц в интернете на задачу рекомендации лучшего предложения или набора слов в тексте. Сам алгоритм состоит в том, что мы текст превращаем в граф, где узлы – это предложения, а для каждого ребра подсчитывается вес, где вес определяется по количеству совпавших слов в двух предложениях.

Таким образом, получается, что можно выбрать предложение с самыми тяжелыми ребрами в качестве предложения, которое описывало бы исходный текст.

4.1.3. Byte pair encoding

Мы используем byte pair encoding (BPE), технику, предложенную Сеннрич для задачи машинного перевода в [22]. BPE – это метод сжатия данных, в котором часто встречающиеся пары байтов заменяются дополнительными символами алфавита. В случае текстов, как в области машинного перевода, наиболее часто встречающиеся слова сохраняются в словаре, а менее часто встречающиеся слова заменяются последовательностью (обычно двумя) токенами. Например, для морфологически богатых языков окончания слов могут быть отделены, поскольку каждая форма слова определенно реже, чем ее основа. Кодирование BPE позволяет нам представлять все слова, включая те, что не встречаются по время обучения, с фиксированным словарным запасом.

4.1.4. Universal transformer network

В то время как рекуррентные нейронные сети могут быть легко использованы для определения модели Encoder-Decoder, тренировка таких моделей очень дорого с точки зрения вычислений. Другой недостаток состоит в том, что они используют только локальную информацию, опуская последовательность скрытых состояний $H = h_1, \ldots, h_N$. То есть любые два вектора из скрытого состояния h_i и h_j связаны с вычислениями $j - i$ RNN, что затрудняет улавливание всех зависимостей в них из-за ограниченной емкости. Чтобы обучить богатую модель, которая изучила бы сложную текстовую структуру, мы должны определить модель, которая опирается на нелокальные зависимости в данных. В этой работе мы принимаем архитектуру модели Universal Transformer [25], которая является модифицированной версией Transformer [2]. Этот подход имеет несколько преимуществ по сравнению с RNN. Прежде всего, его можно тренировать параллельно. Кроме того, все входные векторы связаны друг с другом через механизм Attention. Это подразумевает, что архитектура Transformer учитывает нелокальные зависимости между токенами независимо от расстояния между ними, и, таким образом, она может выучить более сложное представление текста в статье, что оказывается необходимым для эффективного решения задачи суммаризации.

4.2. Выделение ключевых слов

Статья Boudin, Florian [3] рассматривает наиболее сильные подходы к выделению ключевых слов. И основная суть их статьи в том, что они предлагают реализации основных алгоритмов выделения ключевых слов, однако проблема, с которой мы столкнулись состоит в том, что в ядре их библиотеки заложена библиотека spacy [9], которая не работает с русским языком

Таким образом, для данной работы мы реализовали TopicRank, TfIdf и YAKE и TextRank поскольку эти алгоритмы считаются лучшими алгоритмами для выделения ключевых слов. Мы выложили эти реализа-

ции в качестве библиотеки на python [7], реализовав интерфейсы библиотеки Scikit-Learn [21, 1].

4.2.1. Keyverbum

Написанная нами библиотека содержит в себе четыре популярных алгоритма выделения ключевых слов: TopicRank, TfIdf и YAKE и TextRank. Для для того, чтобы их использовать в других проектах, связанных с машинным обучением, они были реализованы как наследники ClassifierMixin, что означает, что у них должны быть реализованы методы fit и predict.

При этом, было принято решение вынести этап чистки данных и разбиение на токены за пределы экстракторов ключевых слов. Таким образом, алгоритмы извлечения ключевых слов ожидают на вход уже готовый список токенов.

В результате, использование этих алгоритмов выглядит следующим образом:

```
from keyverbum.keywords import (Textrank,
                                preprocessing_pipeline)

text = ''some text''
prep_text = preprocessing_pipeline.transform(text)

extractor = Textrank(n_keywords=10)

keywords = extractor.fit_predict(prep_text)
```

Помимо этого, был реализован REST API, который можно развернуть на своем сервере. Таким данными алгоритмами можно воспользоваться в других приложениях, написанных не только на Python.

[7] https://github.com/imemedb/keyverbum

4.2.2. TopicRank

TopicRank - это метод обучения без учителя, целью которого является извлечение ключевых фраз из наиболее важных тем документа. Темы определяются как кластеры похожих ключевых фраз-кандидатов. Извлечение ключевых фраз из документа состоит из следующих шагов, показанных на рисунке 4. Во-первых, документ предварительно обрабатывается (сегментация предложений, разметка слов и тегирование частей речи), а кандидаты в ключевые фразы группируются по темам. Затем темы ранжируются в соответствии с их важностью в документе, и ключевые фразы извлекаются путем выбора одного кандидата ключевой фразы для каждой из наиболее важных тем.

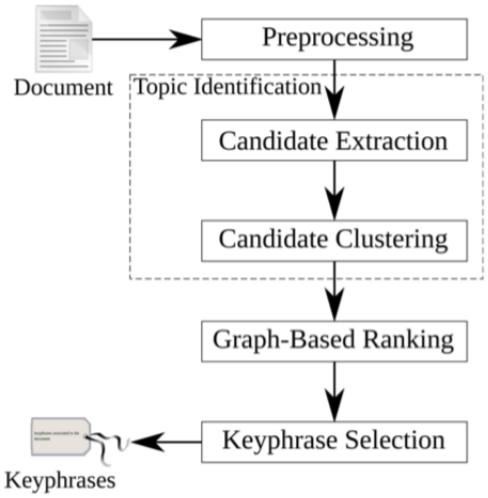

Рис. 4: Принцип работы TopicRank.

4.2.3. YAKE

Алгоритм состоит из 6 основных этапов: предварительная обработка текста, извлечение признаков, индивидуальная оценка токенов, формирование списка ключевых слов кандидатов, дедупликация данных и ранжирование.

Сначала применяется предварительная обработка, которая разбивает текст на отдельные токены на основе пробела или специального

символа (например, разрывы строк, скобки, запятая, точка и т. д.). Во-вторых, выделяется набор из пяти признаков, чтобы охватить характеристики каждого отдельного токена. Это регистр, позиция токена, частота токена, контекст токена и как часто токен появляется в различных предложениях.

Регистр учитывает то, написано ли слово с большой буквы. Оценка позиции слова учитывает то, где находится слово в документе, исходя из предположения, что релевантные ключевые слова часто имеют тенденцию концентрироваться больше в начале документа. Частота слова указывает на то, как часто встречается слово в тексте. Четвертной признак, контекст токенов, вычисляет количество различных токенов, встречающихся с левой (или правой) стороны слова-кандидата. Чем больше количество различных токенов, которые встречаются со словом-кандидатом (с обеих сторон), тем более бессмысленным будет слово-кандидат. Наконец, частота токена в различных предложениях количественно определяет, как часто слово-кандидат встречается в разных предложениях. Подобно частоте слов, частота токена в различных предложениях больше ценит те слова, которые часто встречаются в разных предложениях. Оба признака, тем не менее, сочетаются со контекстом слов, что означает, что чем чаще они встречаются в разных предложениях, тем лучше, если они не встречаются часто с разными словами.

На третьем этапе эти признаки эвристически объединяются одной величиной, так что каждому токену присваивается величина $S(w)$. Эта величина будет принимать участие в процессе генерации ключевых слов, который выполняется на четвертом шаге. Здесь мы рассмотрим скользящее окно из 3 граммов, таким образом генерируя непрерывную последовательность из 1, 2 и 3 граммов ключевых слов-кандидатов. Каждому ключевому слову-кандидату будет присвоен окончательный $S(kw)$, так что чем меньше оценка, тем более значимым будет ключевое

слово. Уравнение 1 описывает это:

$$S(kw) = \frac{\prod_{w \in kw} S(w)}{TF(kw) * (1 + \sum_{w \in kw} S(w))} \tag{1}$$

где $S(kw)$ - оценка ключевого слова-кандидата, определяемая путем умножения (в числителе) оценки $S(w)$ первого члена ключевого слова-кандидата на последующие оценки оставшихся токенов. Результат делится на сумму оценок $S(w)$ для усреднения по длине ключевого слова, так что более длинные n-граммы не получают выгоды только потому, что они имеют более высокое n. Результат далее делится на $TF(kw)$ - частоту использования ключевого слова - для уменьшения влияния менее частых кандидатов. На пятом этапе исключается аналогичные кандидаты из предыдущих шагов. Для этого используется расстояние Левенштейна [12]. Наконец, система выведет список релевантных ключевых слов, составленный из 1, 2, 3 грамм, так что чем ниже показатель S(kw), тем более важным будет ключевое слово.

4.3. Оценки качества

Для оценки качества выделения ключевых слов мы использовали среднюю абсолютную точность P, полнту R и их среднее гармоническое $F1$. Данные метрики являются стандартными при оценке качества моделей выделения ключевых слов. Формулы их рассчета приведены ниже, а их реализацию можно найти в нашей библиотеке keyverbum [8]

$$P_i = \frac{tp}{tp + fp} \quad R_i = \frac{tp}{tp + fn} \quad F1_i = 2\frac{P_i * R_i}{P_i + R_i} \tag{2}$$

На управлениях 2 tp, fp, fn обозначают количество правильно выбранных ключевых слов, ложно правильно выбранных ключевых слов и ложно неправильно, соответственно. При этом, P_i, R_i и $F1_i$ считаются для каждого примера, и имея эти метрики для каждого примера, в результате считается среднее.

Для оценки качества моделей генерации заголовков мы использова-

[8] https://github.com/imemedb/keyverbum/blob/master/keyverbum/evaluate.py

ли метрику ROUGE-1, 2, L P, R, $F1$ [13], где 1, 2, L обозначают размер использованных n-грамм, а P, R, $F1$ – это точность, полнота и их среднее гармоническое, почитанные по формулам 3. В этих формулах $number\_of\_overlaping\_words$ – это количество n-грамм, которые совпадают в предсказании и истинном заголовке, $total\_words\_in\_reference$ – это количество n-грамм в истинном заголовке, а $total\_words\_in\_prediction$ – это количество n-грамм в предсказнном заголовке.

$$\begin{aligned} P &= \frac{number\_of\_overlaping\_words}{total\_words\_in\_reference} \\ R &= \frac{number\_of\_overlaping\_words}{total\_words\_in\_prediction} \end{aligned} \tag{3}$$

Разберем на конкретном примере, как используются эти метрики. Представим, что истинным текстом является "кошка была найдена под кроватью", а некоторым предсказанием является "кошка под кроватью". Тогда для 1-грамм, точность или ROUGE-1 $P = \frac{3}{5} = 0.6$, а полнота или ROUGE-1 $R = \frac{3}{3} = 1.0$.

В случае ROUGE-2 в качестве токенов берутся последовательно биграммы, а при подсчете ROUGE-L количество пересекающихся токенов заменяется на поиск длинны максимальной подпоследовательности и точность и полнота тогда считаются по формуле 4, где LCS – длинна самой длинной пересекающейся подпоследовательности.

$$\begin{aligned} P &= \frac{LCS(reference, prediction)}{total\_words\_in\_reference} \\ R &= \frac{LCS(reference, prediction)}{total\_words\_in\_prediction} \end{aligned} \tag{4}$$

Оценка алгоритмов генерции заголовков, с использованием данных метрик, производилась на датасете РИА новостей [7]. Помимо этого, на основе этого датасета проводилось соревнование по генерации заголовков, где автором было получено 3 место, а описание результатов соревнования было принято в качестве статьи на конференцию "Диалог".

5. Анализ использованных данных

В работе были использованы несколько датасетов с целью выбора алгоритмов под соответствующие задачи. В частности, мы воспользовались датасетом оценки качества выделения ключевых слов [14], недавно опубликованным датасетом Риа новостей [7], а также в нашей работе мы используем данные из 25 групп вконтакте с разным целевым материалом: от медиа контента до полноценных длинных текстов.

5.1. Данные выделения ключевых слов

Датасет	**Тип**	**Количество документов**	**Среднее количество токенов**
Киберленика	Абстракты	4072	5.27
Habr	Блоги	3990	5.03
Независимая газета	Новости	1987	6.11
Россия сегодня	Новости	7217	10.07

Для анализа качества использованных алгоритмов для выделения ключевых слов мы воспользовались датасетом с разнообразными статьями на русском языке [14]. В наборе присутствуют научные статьи из "журнала киберленика" [5], технического блога "хабрахабр" [8] и новостных ресурсов "Независимая газета" [10] и "Россия сегодня" [19].

5.2. Данные для генерации заголовков

Для обучения и выбора алгоритмов генерации заголовков, мы воспользовались недавно опубликованным датасетом Риа новостей. Этот датасет содержит новости с января 2010 по декабрь 2014. В нем имеется 1003869 новостных статей со средним размером заголовка 9.5 слов и средней длинной текста 315.6 слов.

Эти датасет предоставлен в виде необработанных фрагментов оригинальных html-страниц. Это означает, что в данных присутствовали

различные HTML-теги и объекты. В итоге имеется необработанная новость и соответствующий заголовок.

```
<p> <strong> <\strong> <\p> \n <p> <strong> Moscow,
Dec 1   &mdash; RIA news. <\strong> a fire in  
one of the   workshops in  ...<\p>
```

В результате первым делом мы попытались очистить данные от ненужной информации. И так, мы создали препроцессор, который удаляет все html-теги и сущности.

Кроме того, мы обнаружили, что иногда в данных отсутствует текст, поскольку исходные новости представлены в виде изображений (например, снимок экрана Twitter), а новости чисто польские. Это все выбросы, с которых мы очистили данные.

6. Эксперименты

В данной секции мы приводим технические детали, параметры моделей и оценки качества созданных моделей на датасетах описанных выше.

6.1. Генерация заголовка

На практике обычно используется метрика ROUGE [13] при оценке качества алгоритмов суммаризации. Конкретно используются так называемые ROUGE 1,2, L - точность, полнота и F1. Имена в ROUGE X - Y обозначают следующее: X - количество n-грамм, используемых для вычисления метрики Y. В случае n-грамм размера L рассматривается самая длинная общая подпоследовательность из предсказанной последовательности найденая в исходной последовательности. Метрики Y - классическая точность, полнота и их среднее гармоническое.

В случае алгоритма Textrank [26], алгоритма экстрактивной суммаризации, мы воспользовались реализацией из библиотеки gensim [20]. В случае этого алгоритма, выделяется не одно предложение, а подмножество текста определенного размера, потому мы установли параметр отвечающий за размер возвращаемого текста равным 20% от исходного.

Мы взяли state of the art реализацию алгоритма Universal Transformer из Open NMT [16] и обучили этот алгоритм на данных Риа новостей. Мы использовали 4 слоя кодировщика и декодеривщика, 8 heads of attention, вероятность дропаута была выставлена равной 0.3. Для оптимизации был использован алгоритм Adam с изменяющейся скоростью обучения, по правилу из оригинальной статьи о трансформере.

В качестве входа модели мы использовали первые 2000 BPE токенов.

Что касается обучения Byte Pair Encoder, то мы попробовали пердобученный на википедии токенизатор и обученный на датасете Риа новостей. В таблице 1 они обозначены Wiki BPE и Ria BPE, соответственно.

Также для отбора лучших кандидатов для заголовка мы использовали beam search размером 10.

Algorithm\Score	1F	1P	1R	2F	2P	2R	LF	LP	LR
First sentence	0.23	0.16	0.44	0.10	0.07	0.21	0.16	0.15	0.40
Wiki BPE transformer	0.37	0.39	0.36	0.20	0.21	0.19	0.34	0.37	0.34
Ria BPE transformer	0.36	0.37	0.35	0.18	0.20	0.18	0.33	0.36	0.33
Textrank summarization	0.14	0.09	0.41	0.05	0.03	0.17	0.09	0.08	0.38

Таблица 1: ROUGE-1,2,F1, precision and recall scores.

Score	Example
0.00	**Text**: 8 декабря 1991 года россия, белоруссия и украина подписали соглашение о создании содружества независимых государств (снг). **Ground truth**: встреча в беловежской пуще **Prediction**: заявил
0.00	**Text**: более 70 международных художников и арт-групп примут участие в основном проекте ”больше света” 5-й московской биеннале современного искусства,который будет показан в цвз ”манеж” с 20 сентября по 20 октября, сообщили риа новости в пресс-службе проекта. **Ground truth**: стали известны все участники основного проекта 5-й московской биеннале **Prediction**: более 00 художников представят проект ”больше света” в ”манеже”
0.99	**Text**: фонд ”сколково” и корпорация intel подписали в четверг соглашение о сотрудничестве, передает корреспондент риа новости. **Ground truth**: ”сколково” и intel подписали соглашение о сотрудничестве **Prediction**: ”сколково” и intel подписали соглашение о сотрудничестве

Таблица 2: Лучшие и худшие результаты предсказания модели. Score среднее от ROUGE-1,2,L F1.

В Таблице 2 среднее от ROUGE-1,2, L F1 используется в качестве метрики. Мы показываем два примера: лучшее и худшее возможное предсказание нашего лучшего алгоритма – Wiki transformer-a.

6.2. Выделение ключевых слов

Как видно из Таблицы 3, лучшие результаты показывает TfIdf. Однако, как было сказано в самом начале, задача извлечения ключевых слов еще не имеет универсального решения, потому, несмотря на такой отрыв TfIdf, есть смысл пробовать другие алгоритмы на своих задачах.

Датасет	Независимая газета			Россия Сегодня			Habr		
Алгоритм	P	R	F1	P	R	F1	P	R	F1
TfIdf	0.13	0.07	0.08	0.08	0.08	0.08	0.16	0.08	0.10
TopicRank	0.05	0.02	0.03	0.03	0.03	0.03	0.05	0.02	0.03
TextRank	0.09	0.04	0.06	0.06	0.06	0.06	0.12	0.6	0.07
YAKE	0.06	0.03	0.04	0.05	0.05	0.04	0.13	0.06	0.08

Таблица 3: Средняя абсолютная точность (P), полнота (R), и F1 мера.

Заключение

В данной работе мы предложили решение задачи краткого описания новостного ресурса. Решение включает в себя путь того, как с этой системой будет взаимодействовать пользователь, техническую архитектуру системы, обоснование выбранных алгоритмов, отталкиваясь от соответствующих решаемых подзадач.

Как результат, с алгоритмической точки зрения, система была разбита на две подзадачи: выделения ключевых слов и генерации заголовков. Нами были предложены лучшие алгоритмы, для решения соответствующих задач. Также в этой работе мы заложили фундамент для дальнейших исследований в этой области, предоставив интерфейс для разработчиков и людей.

Список литературы

[1] API design for machine learning software: experiences from the scikit-learn project / Lars Buitinck, Gilles Louppe, Mathieu Blondel et al. // ECML PKDD Workshop: Languages for Data Mining and Machine Learning. — 2013. — P. 108–122.

[2] Attention Is All You Need / Ashish Vaswani, Noam Shazeer, Niki Parmar et al. // CoRR. — 2017. — Vol. abs/1706.03762.

[3] Boudin Florian. pke: an open source python-based keyphrase extraction toolkit // Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations. — Osaka, Japan, 2016. — December. — P. 69–73. — URL: http://aclweb.org/anthology/C16-2015.

[4] Braslavski, Gustelev. The system of automatic summarization of news messages based on machine learning. — URL: http://rcdl2007.pereslavl.ru/papers/paper_54_v1.pdf.

[5] CyberLenica. — URL: https://cyberleninka.ru/.

[6] DeMers Jayson. 59 Percent Of You Will Share This Article Without Even Reading It. — 2016. — aug. — URL: https://www.forbes.com/sites/jaysondemers/2016/08/08/59-percent-of-you-will-share-this-article-without-even-reading-

[7] Gavrilov Daniil, Kalaidin Pavel, Malykh Valentin. Self-Attentive Model for Headline Generation // Proceedings of the 41st European Conference on Information Retrieval. — 2019.

[8] Habr. — URL: https://habr.com/ru/.

[9] Honnibal Matthew, Johnson Mark. An Improved Non-monotonic Transition System for Dependency Parsing // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. — Lisbon, Portugal : Association for Computational

Linguistics, 2015. — September. — P. 1373–1378. — URL: https://aclweb.org/anthology/D/D15/D15-1162.

[10] Independent Journal. — URL: http://www.ng.ru/.

[11] KEA: Practical Automatic Keyphrase Extraction / Ian H. Witten, Gordon W. Paynter, Eibe Frank et al. // Proceedings of the Fourth ACM Conference on Digital Libraries. — DL '99. — New York, NY, USA : ACM, 1999. — P. 254–255. — URL: http://doi.acm.org/10.1145/313238.313437.

[12] Levenshtein Vladimir Iosifovich. Binary codes capable of correcting deletions, insertions and reversals. // Soviet Physics Doklady. — 1966. — feb. — no. 8. — P. 707–710. — Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[13] Lin Chin-Yew. ROUGE: A Package for Automatic Evaluation of summaries. — 2004. — P. 10.

[14] Mannefedov. ru-kw-eval-datasets. — 2019. — Jan. — URL: https://github.com/mannefedov/ru_kw_eval_datasets.

[15] Mihalcea Rada. Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization. — 2004. — 01. — Vol. 170-173.

[16] OpenNMT: Open-Source Toolkit for Neural Machine Translation / G. Klein, Y. Kim, Y. Deng et al. // ArXiv e-prints. — 1701.02810.

[17] Page Larry, Brin Sergey, Motwani R., Winograd T. The PageRank Citation Ranking: Bringing Order to the Web. — 1998.

[18] Putra Jan Wira Gotama, Kobayashi Hayato, Shimizu Nobuyuki. Incorporating Topic Sentence on Neural News Headline Generation. — 2018.

[19] RT in russian, latest news in the world and Russia. — URL: https://russian.rt.com/.

[20] Řehůřek Radim, Sojka Petr. Software Framework for Topic Modelling with Large Corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. — 2010. — P. 45–50.

[21] Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825–2830.

[22] Sennrich Rico, Haddow Barry, Birch Alexandra. Neural Machine Translation of Rare Words with Subword Units // CoRR. — 2015. — Vol. abs/1508.07909.

[23] A Text Feature Based Automatic Keyword Extraction Method for Single Documents / Ricardo Campos, Vítor Mangaravite, Arian Pasquali et al. // Advances in Information Retrieval / Ed. by Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, Allan Hanbury. — Cham : Springer International Publishing, 2018. — P. 684–691.

[24] Turney Peter D. Learning Algorithms for Keyphrase Extraction // Information Retrieval. — 2000. — Vol. 2. — P. 303–336.

[25] Universal Transformers / Mostafa Dehghani, Stephan Gouws, Oriol Vinyals et al. // CoRR. — 2018. — Vol. abs/1807.03819. — 1807.03819.

[26] Variations of the Similarity Function of TextRank for Automated Summarization / Federico Barrios, Federico López, Luis Argerich, Rosa Wachenchauzer // CoRR. — 2016. — Vol. abs/1602.03606.

[27] YAKE! Collection-Independent Automatic Keyword Extractor / Ricardo Campos, Vítor Mangaravite, Arian Pasquali et al. // Advances in Information Retrieval / Ed. by Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, Allan Hanbury. — Cham : Springer International Publishing, 2018. — P. 806–810.

[28] article essence. Article essence. — 2019. — feb. — URL: https://opendatascience.slack.com/messages/C5VQ222UX.

[29] smm pub. 20 websites for vk groups analysis. — 2019. — may. — URL: https://vk.com/page-43503600_49056860.

[30] tldr arxiv. tldr arxiv. — 2019. — feb. — URL: https://t.me/tldr_arxiv.

[31] vkontakte. vkontakte. — 2019. — feb. — URL: https://vk.com/feed.