

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ
ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН**

Направление 09.03.01 - Информатика и вычислительная техника
(код и наименование направления)

**Профиль 02 – Программное обеспечение вычислительной техники и
автоматизированных систем**

Допускаю к защите
Заведующий кафедрой ЭВМ
_____ / Страбыкин Д.А. /
(подпись) (Ф.И.О.)

**Разработка программы генерации подписи к
цифровому изображению**

Пояснительная записка выпускной квалификационной работы
ТПЖА 090301.02.087 ПЗ

Разработал: студент гр.ИВТб-4302-02-00 _____ / Пивоваров И.А. / _____

Руководитель: к.т.н., доцент _____ / Чистяков Г.А. / _____

Нормоконтролер: к.т.н., доцент _____ / Скворцов А.А. / _____
(подпись) (Ф.И.О.) (дата)

Киров 2020

Реферат

Пивоваров И.А. Разработка программы генерации подписи к цифровому изображению: ТПЖА.090301.087 ПЗ: ВКР / ВятГУ, каф. ЭВМ; рук. Чистяков Г.А. – Киров, 2020. – Гр. ч. 8 л. ф. А1, ПЗ 93 с, 27 рис., 10 табл., 10 источников, 5 прил.

СОПОСТАВЛЕНИЕ ИЗОБРАЖЕНИЯ И ТЕКСТА, СБОР И АНАЛИЗ ДАННЫХ, ВЕКТОРИЗАЦИЯ ТЕКСТА, МАШИННОЕ ОБУЧЕНИЕ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, PYTHON, TENSORFLOW, KERAS

Объект выпускной квалификационной работ – программа генерации подписи к цифровому изображению.

Целью данной выпускной квалификационной работы является создание программы, которая способна генерировать подписи на основе цифрового изображения.

В рамках выпускной квалификационной работы представлено решение задачи сопоставления изображения и текста. Разработаны модели для сбора данных, для векторизации текста. Построена модель сверточной нейронной сети и обучена на собранных данных. Результатом является программа, способная к изображению подбирать текст.

Разработанная программа имеет перспективы развития в качестве программы для генерации уникальных подписей для различных изображений.

Содержание

Введение.....	3
1 Обзор предметной области	5
1.1 Постановка задач	6
1.2 Техническое задание	7
2 Сбор данных	10
2.1 Модель решения задачи сбора данных	10
2.2 Анализ целевых ресурсов	11
3 Анализ текстово-графических данных	19
3.1 Методы векторизации текстовой информации	19
3.2 Математическая модель для векторизации текста.....	21
3.3 Методы анализа графической информации.....	23
3.4 Нейронные сети: обоснование применения.....	23
3.5 Математическая модель сверточной нейронной сети	35
4 Проектирование и реализация программы.....	40
4.1 Разработка общей структуры программы	40
4.2 Средства реализации модулей системы.....	41
4.3 Сбор и подготовка данных для обучения нейронной сети	43
4.4 Разработка инструмента для векторизации текста	51
4.5 Построение модели сверточной нейронной сети. Обучение нейронной сети	54
4.6 Результаты обучения сверточной нейронной сети	58
5 Результаты экспериментальной апробации.....	64
Заключение	70
Приложение А. Руководство по использованию	71
Приложение Б. Листинг программы для сбора данных.....	72
Приложение В. Листинг программы модели нейронной сети	77
Приложение Г. Авторская справка.....	91
Приложение Д. Библиографический список	92

					ТПЖА 090301.02.087 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Разработка программы генерации подписи к цифровому изображению			Лист	Лист	Листов	
Разраб.	Пивоваров И.А.										
Провер.	Чистяков Г.А.								2	93	
								Кафедра ЭВМ Группа ИВТ-42			
Н. Контр.	Скворцов А.А.										
Утверд.	Страдыкин Д.А.										

ВВЕДЕНИЕ

Интернет – это место, где каждый человек пытается привлечь внимание окружающих людей к проблемам общества. Отсюда появляется вопрос: как лучше преподнести информацию людям?

Раньше люди знали, что самый простой способ зафиксировать информацию – это рисунок. Рисунки уточняют, подкрепляют, а то и заменяют наши слова. Кроме того, изображения понятны всем вне зависимости от возраста и владения языком, некоторые из них стали стандартом международного общения.

Со временем люди научились обмениваться информацией не только с помощью рисунков или устной речи, но и с помощью письменности. Ни для кого не секрет как написать какую-нибудь статью на очень важную проблему, но в этот момент появляется другая проблема: как привлечь читателя к потреблению неизвестной ему информации, как пробудить в нем интерес к чужому творчеству?

К сожалению, современном мире очень сложно привлечь внимание аудитории. Для того, чтобы читатель начал читать вашу статью, ему необходимо выбрать именно её из большого списка конкурирующих статей. Первым, что увидит читатель, пролистывая миллионы однотипных статей, будет заголовок. Всего одно предложение может привлечь большую аудиторию. Поэтому ваш заголовок должен быть наиболее привлекательным, название статьи должно цеплять читателя, побуждая его к прочтению.

Выбрав неудачный заголовок, вы делаете свою статью невидимой для большей части аудитории. Заголовок является важнейшим элементом интернет-страниц, рекламных объявлений, видеоматериалов. Он привлекает внимание аудитории к контенту. Материалы остаются незамеченными, если вы используете серый и невзрачный заголовок.

В рамках дипломного проекта будет рассматриваться проблема привлечения аудитории посредством создания некоторого привлекательного заголовка к статье, опираясь при этом на сопутствующие изображения.

					ТПЖА 090301.02.087 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Для человека не возникает большого труда придумать какое-нибудь высказывание для уже готового текста и пары картинок, но в большинстве случаев этот придуманный заголовок будет не новым и не актуальным или же скучным и неказистым. Можно предположить, что спустя некоторое времени придет свежая мысль и новый заголовок будет придуман, но он как первый будет не слишком привлекательным.

Рассмотрев проблему в вакууме, можно прийти к выводу, что без подручных средств ее не решить. Конечно, можно привлечь сторонних специалистов, которые ознакомившись с материалами помогут составить яркий заголовок, но в современном мире существует и другое решение данной проблемы – это обратиться к электронной вычислительной технике.

Обратившись к научной литературе можно увидеть, что машины уже научились писать картины по их словесному описанию, что возможно распознавать рукописный текст, но что если развернуть ситуацию наоборот, что если необходимо написать законченное предложение об изображении, которое еще нигде не было написано или озвучено, совершенно новое, никем не придуманное?

К сожалению, на сегодняшний день нет никаких открытых исследований по этому вопросу, возможно их еще не проводили, и эта проблема лишь сейчас набирает оборот.

Поскольку исследований по обозначенной проблематике мало, изучение данной проблемы будет актуально для научного сообщества, поэтому цель дипломного проектирования – создание программы, способной генерировать подписи на основе цифрового изображения.

1.1 Постановка задач

Под разработкой программы генерации подписи к цифровому изображению будет пониматься решение задачи сопоставления текста и изображения.

После того как была поставлена цель, необходимо продумать пути решения. Из современных научных работ можно узнать, что уже найдено решение таких задач, как распознавание образов и генерация изображения по тексту, и все это благодаря обширному набору различных методов машинного обучения.

Машинное обучение позволяет найти не прямое решение задачи, а обучить некоторую модель на основе решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

Задача распознавание образов очень схожа с задачей сопоставления изображения и текста. И в той, и другой задаче необходимо обрабатывать изображения, но целью данного проекта является получение осмысленного текста, а не один из заранее predetermined образов. Поэтому для решения задачи сопоставления изображения и текста будет применяться методы решения задачи распознавания образов.

Проанализировав научную литературу по решению задачи распознавания образов и учитывая особенности задачи сопоставления изображения можно определить следующие задачи, реализуемые в рамках проекта.

1.1.1 Сбор данных

Необходимо подготовить материал для исследования зависимостей между заголовками и изображениями.

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

6

Необходимо собрать обширную коллекцию данных, состоящую из заголовков статей и изображений. Объем коллекции должен превышать 100.000 уникальных записей.

1.1.2 Определить методы для анализа текстово-графических данных

Необходимо рассмотреть различные способы обработки текстово-графической информации и на основе полученных зависимостей построить информационные модели.

1.1.3 Реализация моделей

Необходимо реализовать модели программно, выполнить их настройку и непосредственно корректировку в практических условиях. Составить подробное описание их совместной работы и получить некоторый результат.

1.1.4 Анализ результатов

Подвергнуть практические результаты детальному анализу и оценке, рассмотреть возможные перспективы развития и дальнейшей разработки.

1.2 Техническое задание

Задача сопоставления изображения и текста является перспективной для изучения и исследований. Весь дипломный проект будет посвящен решению этой задачи, а разработанная программа послужит для демонстрации полученных результатов.

В ходе анализа предметной области было составлено техническое задание на основе уже имеющихся методов решения похожих задач.

1.2.1 Назначение и область применения

Первоначально задумывалось применение разрабатываемого программного изделия в качестве помощника редактора при крупных новостных веб-ресурсах. Программа позволила бы автоматизировать процесс подбора заголовков к новостным статьям, это упростило бы работу редакторов, так как в их обязанности входят придумать цепляющий и информативный новостной заголовок.

Кроме того, программа также может быть применена в других областях, где необходимо придумать описание заранее известного изображения.

1.2.2 Функциональные требования

Функциональные требования будут выдвигаться к каждой отдельно поставленной задаче.

При решении задачи сбора данных необходимо реализовать инструмент способный:

- 1) сделать запрос к веб-ресурсу;
- 2) найти необходимую информацию в полученном ответе с веб-ресурса;
- 3) сохранить собранную информацию.

Решение задачи сопоставления изображения и текста будет основываться на одном из методов машинного обучения – искусственных нейронных сетях.

При решении задачи обучения нейронной сети необходимо:

- 1) реализовать инструмент для представления текста в виде вектора;
- 2) построить модель нейронной сети, пригодной для распознавания образов на изображении;
- 3) обучить модель нейронной сети используя собранную коллекцию данных.

1.2.3 Системные требования

При обучении нейронной сети необходимо большое количество вычислительных ресурсов, поэтому предполагается использовать стороннюю аппаратуру. Многие крупные мировые компании разрешают использовать их оборудования в научных исследованиях. В приоритете получить возможность работать с тензорными процессорами (TPU) или с графическими ускорителями (GPU), либо с производительными центральными процессорами (CPU).

Общие требования к техническому обеспечению:

- 1) наличие современным TPU или GPU;
- 2) наличие 15-25Гб оперативной памяти;
- 3) наличие 100Гб свободного пространства на диске.

1.2.4 Требования к входным и выходным данным:

К входным и выходным данным программы предъявляются следующие требования:

- 1) на вход программа получает изображение;
- 2) на выходе программа выдает текст.

В данном разделе был произведен обзор предметной области, на основе которого поставлена цель проектной деятельности, определены задачи, а также разработано техническое задание на основе уже имеющихся методов решения похожих задач.

2 СБОР ДАННЫХ

Данный раздел посвящается первой поставленной задаче – процессу создания коллекции данных.

В разделе будет рассмотрена формальная модель задачи сбора данных, которая в дальнейшем получит программную реализацию в виде инструмента, будет произведен анализ веб-ресурсов, с которых возможно собрать необходимые данные, и будет подобрана структура для хранения коллекции данных.

2.1 Модель решения задачи сбора данных

Модель решения задачи сбора данных можно представить в виде кортежа:

$$I = \langle M_{ijk}, Z \rangle, \quad (1)$$

где M_{ijk} – массив данных ($i = \overline{1, n}$, $j = \overline{1, m}$, $k = \overline{1, p}$, n – количество новостных статей на k -ом новостном веб-ресурсе ($k \in Z$), m – количество данных, собранных по i -ой статье, p – количество новостных веб-ресурсов ($\overline{1, p} \in Z$); Z – массив веб-ресурсов.

Эффективность сбора данных будет определяться качеством выборки:

$$\mathcal{E}_\tau = f(K_\tau), \quad (2)$$

где f – функция пороговой активации, зависящая от K_τ , K_τ – качество данных, хранящихся на веб-ресурсе ($\tau \in Z$).

$$f(x) = \begin{cases} 0, & x < 85\% \\ 1, & x \geq 85\% \end{cases} \quad (3)$$

$$K_\tau = \frac{C_\tau}{n} * 100\%, \quad (4)$$

где C_τ – количество новостных статей с изображения для τ новостного ресурса, n – объем выборки новостных статей с τ новостного ресурса ($\tau \in Z$).

Согласно теории вероятностей, рассчитано, что при доверительной вероятности 95%, доверительном интервале $\pm 5\%$ и генеральной совокупности более 100000 объектов, объем выборки n составляет 383 объекта.

2.2 Анализ целевых ресурсов

Согласно поставленной задаче необходимо проанализировать новостные веб-ресурсы (далее – сайты) и выделить целевые сайты, сбор данных с которых будет максимально эффективен. Необходимо рассматривать только крупные российские новостные сайты. Далее будут рассмотрены следующие новостные сайты:

- 1) новостной канал «RT на русском»;
- 2) новостное агентство «РИА Новости»;
- 3) новостное агентство «РБК»;
- 4) интернет-издание «Lenta.ru».

2.2.1 Анализ качества извлекаемых данных с целевых ресурсов

Анализ качества контента с целевых сайтов начинается с определения процента контента высокого качества.

При описании модели было оговорено, что рассматриваемые сайты должны иметь контент с высоким качеством, который должен соответствовать метрике: $\mathcal{E}_t = f(K_t)$. Согласно формуле для оценки эффективности, подсчитаем количество статей с изображением для первых 383 статей из раздела «Политика» (на некоторых сайтах раздел «Мир»), при этом не должно совершаться дополнительных переходов (запросов) к содержанию статьи для уточнения каких-то необходимых данных, вся информация должна располагаться в одном месте.

$$K_{\text{«RT на русском»}} = \frac{C_{\text{«RT на русском»}}}{383} * 100\% = \frac{372}{383} * 100\% = 97\%$$

$$K_{\text{«РИА Новости»}} = \frac{C_{\text{«РИА Новости»}}}{383} * 100\% = \frac{378}{383} * 100\% = 99\%$$

$$K_{\text{«РБК»}} = \frac{C_{\text{«РБК»}}}{383} * 100\% = \frac{130}{383} * 100\% = 34\%$$

$$K_{\text{«Lenta.ru»}} = \frac{C_{\text{«Lenta.ru»}}}{383} * 100\% = \frac{138}{383} * 100\% = 36\%$$

После проведенного анализа можно убрать из рассмотрения новостные сайты «РБК» и «Lenta.ru», так как они имеют нулевой показатель эффективности \mathcal{E}_t по сравнению с сайтами «РТ на русском» и «РИА Новости».

Для оставшихся двух новостных сайтов проведем объективную оценку эффективности извлечения данных, а также оценим общее количество данных на новостном сайте:

- количество данных извлекаемых за один запрос;

Для новостного сайта «РТ на русском» при первом запросе раздела «Мир» выдается 12 новостей, и при каждом последующем еще 6.

Для новостного сайта «РИА Новости» при первом запросе раздела «Политика» выдается 20 новостей, и при каждом последующем еще 20.

- общее количество данных на новостных сайтах;

Для новостного сайта «РТ на русском» для раздела «Мир» не удалось выяснить общее количество статей для данного раздела.

Для новостного сайта «РИА Новости» при первом запросе всех статей за все время из раздела «Политика» выдается чуть больше 197 тысяч.

- поверхностное сравнение DOM-разметки.

Далее будет рассмотрено сравнение HTML-разметок для новостного сайта «РТ на русском» и новостного сайта «РИА Новости».

Как видно на рисунке 1, HTML-разметка на новостном сайте «РТ на русском» выглядит разрозненно, используется большая вложенность со сложными классами и самое важное: адрес прикрепленного изображения

хранится в тэге div → атрибуте style → background-image → url(), что потребует дополнительные затраты на разбор и извлечение адреса.

```

▼<li class="listing_column listing_column_sections
listing_column_sections_1 ">
  ▼<div class="listing_card listing_card_sections ">
    ▼<div class="card card_sections card_sections_1 ">
      ▶<div class="card_trend card_trend_sections
card_trend_sections_1 card_trend_with-label">...</div>
      ▼<div class="card_heading
card_heading_sections card_heading_sections_1
"
      >
        ▼<a class="link link_color" href="/world/article/699833-
shotlandiya-referendum-nezavisimost">
          "
          В ожидании брексита: к чему приведёт новая
          попытка Шотландии добиться независимости
          "
        </a>
      </div>
      ▼<div class="card_date-author card_date-author_sections
card_date-author_sections_1 ">
        <div class="card_date card_date_sections
card_date_sections_1 ">
          19:16
        </div>
        ▶<div class="card_author card_author_sections
card_author_sections_1 ">...</div>
      </div>
      ▼<div class="card_cover card_cover_sections
card_cover_sections_1 ">
        ▼<div class="cover cover_sections cover_sections_1 ">
          ▶<div class="cover_media cover_media_ratio
cover_media_bgi cover_media_sections_1 " style=
"background-image: url(https://cdni.rt.com/russian/images/
2019.12/thumbnail/5dfb51abae5ac9319f723851.jpg)">...</div>
        </div>
      </div>
      ▶<div class="card_summary card_summary_sections
card_summary_sections_1 ">...
      </div>
      ▶<div class="card_spot-im-count card_spot-im-count_sections
card_spot-im-count_sections">...</div>
    </div>
  </li>
  ▶<li class="listing_column listing_column_sections
listing_column_sections_2 ">...</li>
  ▶<li class="listing_column listing_column_sections
listing_column_sections_3 ">...</li>

```

Рисунок 1 - HTML-разметка на новостном сайте «RT на русском»

Как видно на рисунке 2, HTML-разметка на новостном сайте «РИА Новости» похожа на разметку с новостного сайта «РТ на русском», также используется большая вложенность, но уже с простыми классами и адрес прикрепленного изображения хранится в теге source → атрибуте srcset, что потребует меньше затраты на разбор и извлечение адреса.

```

▼<div class="list-item">
  ▼<div class="list-item__content">
    ▼<a href="/20191220/1562664298.html" class="list-item__image">
      ▼<picture>
        <source media="(min-width: 480px)" media-type="ar16x9" srcset=
          "https://cdn23.img.ria.ru/images/155214/54/1552145477_0:0:2564:
          1442_436x0_80_0_0_3da1a49dc09e40cb6a5f952c16f41fd3.jpg">
        <source media="(min-width: 375px)" media-type="ar4x3" srcset=
          "https://cdn21.img.ria.ru/images/155214/54/1552145477_451:0:2459:
          1506_186x0_80_0_0_39318aee04251f4f37ce99d0a9d3b8c2.jpg">
        <source media="(min-width: 0px)" media-type="ar1x1" srcset=
          "https://cdn23.img.ria.ru/images/155214/54/1552145477_702:0:2208:
          1506_140x0_80_0_0_86783e3ad35cdef131a1ef56572aa3e3.jpg">
        
      </picture>
    </a>
    <a href="/20191220/1562664298.html" class="list-item__title color-
      font-hover-only">В ГД и СФ считают важным обсуждение законопроектов о
      защите инвестиций </a>
    ::after
  </div>
  ▼<div class="list-item__info">
    <div class="list-item__date">20:28</div>
    ▶<div class="list-item__views">...</div>
  </div>
  ▼<div class="list-item__tags">
    ▶<div class="list-item__tags-more">...</div>
    ▼<ul>
      ▶<li class="list-tag active color-border color-font" data-sid=
        "politics">...</li>
      ▼<li class="list-tag" data-sid="economy">
        <span class="list-tag__text">Экономика</span>
        ▶<span class="list-tag__icon">...</span>
      </li>
    </ul>
  </div>
</div>
▶<div class="list-item">...</div>
▶<div class="list-item">...</div>

```

Рисунок 2 - HTML-разметка на новостном сайте «РИА Новости»

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

14

Как видно на рисунке 2, HTML-разметка на новостном сайте «РИА Новости» похожа на разметку с новостного сайта «RT на русском», также используется большая вложенность, но уже с простыми классами и адрес прикрепленного изображения хранится в теге source → атрибуте srcset, что потребует меньшие затраты на разбор и извлечение адреса.

Рассмотрев все вышеперечисленное, можно однозначно утверждать, что сбор данных с новостного сайта «РИА Новости» будет гораздо эффективнее, чем с других аналогичных сайтов.

2.2.2 Детальный анализ DOM-разметки целевого сайта

После определения целевого сайта необходимо более детально рассмотреть HTML-структуру выбранного новостного сайта.[4]

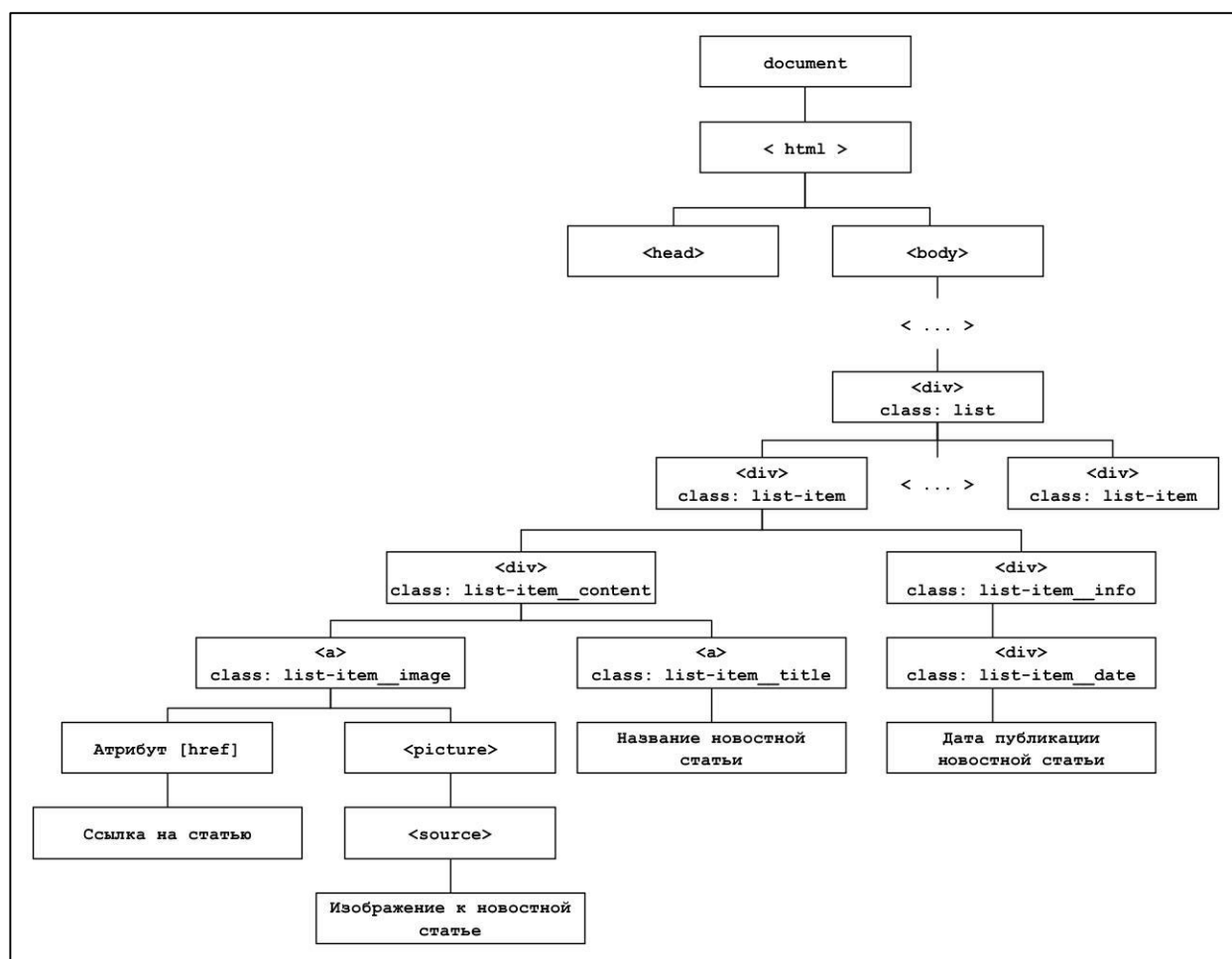


Рисунок 3 - HTML-структура сайта «РИА Новости»

Для новостного сайта «РИА Новости» HTML-структура (рисунок 3), как и для всех сайтов, начинается с тэгов <html>, <head>, <body>. Пропустим всю вложенную разметку, которая есть между началом документа и целевым участком, где хранятся интересные данные.

Список всех новостей начинается с тэга <div> класса «list», в него вложены другие контейнеры <div> с классами «list-item», каждый из которых содержит одну целевую единицу данных. Все <div> с классами «list-item» имеют одинаковую структуру, поэтому рассмотрим один самый первый контейнер.

Контейнер <div> с классом «list-item» включает:

- 1) ссылку на статью, которая хранится в атрибуте «href» тэга <a> класса «list-item__image», который принадлежит тэгу <div> класса «list-item__content»;
- 2) изображение к новостной статье, хранящееся в тэге <source> внутри тэга <picture> внутри тэга <a> класса «list-item__image», который принадлежит тэгу <div> класса «list-item__content»;
- 3) название новостной статьи, хранящееся в тэге <a> класса «list-item__title», который принадлежит тэгу <div> класса «list-item__content»;
- 4) дату публикации статьи, хранящуюся в тэге <div> класса «list-item__date», который принадлежит тэгу <div> класса «list-item__info».

2.2.3 Разработка структуры хранения данных

При сборе большого набора данных возникает задача эффективного размещения данного набора в памяти ЭВМ. В данном случае имеются таблицы данных и отдельные файлы-изображения.

Для размещения изображения в файловой системе целесообразно создать отдельное пространство (папку), которое необходимо связать с соответствующей таблицей, но для того чтобы связь структуры, хранящей в

таблице, и размещением изображения в файловой системе была однозначной, требуется дополнить таблицу специальным столбцом «Относительный путь к изображению».

Для эффективного хранения текстовой информации существуют различные форматы хранения. Далее будут рассмотрены два наиболее известных и эффективных: JSON/XML и CSV. Рассмотрим их достоинства и недостатки на примере хранения строки собранных данных:

CSV: Источник | Ссылка на источник | Новостной раздел | Название |
Относительный путь к файлу

JSON/XML: [{"Source": "Источник", "Link to source": "Ссылка на источник", "News section": "Новостной раздел", "Title": "Название", "Relative path": "Относительный путь к файлу"}, {...}]

Теперь сравним длину строки JSON/XML и CSV. У JSON/XML строка с информацией занимает в 2 раза больше места, чем у CSV.

Стоит отметить, что формат JSON/XML структурирован, поэтому способен описать любую, даже очень сложную структуру.

В то время, как формат CSV удобен тем, что очень компактен.

Рассмотрев оба формата хранения, для проекта лучше всего подойдет формат CSV, потому что целевая структура хранения данных не является сложной и не оправдывает высоких затрат на ее содержание. Исходя из данного заключения, определим структуру для CSV-файла для хранения таблицы с полученной информацией.

Источник информационной единицы	Ссылка на источник	Информационный раздел	Название	Относительный путь
---------------------------------	--------------------	-----------------------	----------	--------------------

Рисунок 4 - Структура хранения информации

Полученная структура включает следующие поля:

- 1) источник информационной единицы;

- 2) ссылка на источник;
- 3) информационный раздел;
- 4) название;
- 5) относительный путь.

В рассмотренном разделе была рассмотрена модель решения задачи сбора данных. Согласно метрике, описанной в модели, был отобран единственный веб-ресурс с качественным контентом, для которого выполнен анализ DOM-разметки, необходимой для последующего сбора данных. Для хранения коллекции данных была описана структура.

3 АНАЛИЗ ТЕКСТОВО-ГРАФИЧЕСКИХ ДАННЫХ

Данная глава посвящается второй поставленной задаче – анализу методов для работы с текстово-графическими данными.

В главе будут рассмотрены способы, с помощью которых можно решить задачу по анализу как текстовых, так и графических данных. После рассмотрения методов будут выбраны некоторые из них и по ним построены модели.

3.1 Методы векторизации текстовой информации

Под понятием векторизация следует понимать некую функцию или преобразование, которое позволяет представить входное множество данных в виде конечного вектора определенной длины.

Для векторизации используют специальные модели, рассмотрим наиболее популярные из существующих:

1) сумка слов (Bag of Words);

Bag of Words – детальная репрезентативная модель, учитывающая количество вхождения слова в корпус текста. Изначально модель не учитывает последовательность слов, следовательно, не распознает связь слов в тексте, и способна лишь определить количество вхождений отдельных слов в корпус текста. На практике модель реализуется как словарь, где ключ – слово, а значение – количество вхождений слова в корпус.

Чаще всего модель дорабатывается: вместо точно подсчета значений, учитывается факт вхождения слова в корпус. Если слово встретилось, то в словаре значение для него устанавливается в единицу и не меняется при нахождении другого такого слова.

К другому варианту доработки можно отнести использование N-грамм. N-грамма – это комбинация из N последовательных терминов. Модель на основе N-грамм способна уловить некоторую семантическую связь между словами. В

этом случае словарь представляется как набор различных N-грамм, и для каждой высчитывается либо точное количество вхождения в корпус, либо хотя бы одно.

2) модель на основе статистической меры TF-IDF;

TF-IDF – это статистическая мера, которая используется для оценки важности слова в контексте всего корпуса. Для каждого слова рассчитывается его вес внутри корпуса. Он пропорционален частоте употребления этого слова в отдельно взятом документе корпуса и обратно пропорционален частоте употребления слова во всем корпусе.

Модель на основе данной меры и мешка слов позволяет создать словарь из слов, значения частоты которых выше некоторого порога, и каждому слову будет присвоено некоторое значение на основе меры TF-IDF, что позволит сравнивать предложения в будущем.

3) Word2Vec.

Word2Vec – это современный инструмент для анализа семантики естественных языков, разработанный Google-ом.

Word2vec получает на вход текстовый корпус и сопоставляет каждому слову вектор. На выходе получается «координаты слова». Сначала он генерирует словарь корпуса, а затем, «обучаясь» на входных данных, вычисляет векторное значение слов. Векторное представление строится таким образом, что слова, встречающиеся в тексте рядом с одинаковыми словами, будут иметь близкие векторы.

На основе этих моделей существуют и другие, более сложные, например, модель, предложенная лабораторией компьютерной лингвистики Стенфордского университета, – GloVe.

В таблице 1 представлено сравнение моделей по наиболее значимым критериям.

После рассмотрения всех достоинств и недостатков моделей было принято решения остановиться на модели основанной на мере TF-IDF, так как данная модель просто реализуема и настраиваемая, а также просто векторизировать

текст. Но она не лишена недостатков, возникает сложность при добавлении в модель новых слов.

Таблица 1 - Сводная таблица достоинств и недостатков

Критерий сравнения	Модель Bag of Words	Модель на основе TF-IDF	Word2Vec
Учитывает семантическую близость слов	+/-	+/-	+
Качество получаемых представлений	-	+/-	+
Простота добавления новых слов в модель	-	-	+
Простота в реализации и настройке модели	+	+	-
Возможность векторизации предложения	+	+	-

3.2 Математическая модель для векторизации текста

Для решения задачи обработки текстовой информации необходимо разработать функцию, которая будет представить некоторый текст T в виде конечного вектора V размерностью q , где каждый элемент вектора отвечает за некоторый признак входного текста T :

$$V^q = f(T) \quad (5)$$

Перед началом векторизации текста необходимо предобработать текст:

- 1) необходимо убрать всю пунктуацию из текста T ;

$$T \setminus S_1 = \{ch \in T | ch \notin S_1\}, \quad (6)$$

где T – исходный текст,

S_1 – множество, включающее в себя все существующие знаки пунктуации,

ch – символ из множества T .

- 2) необходимо представить текст T как одномерный массив;

$$T = [t_0 \dots t_n], \quad (7)$$

где t_i – это отдельно взятое слово из текста T ,

n – число слов в тексте.

- 3) необходимо убрать незначащие элементы текста T , такие как стоп-слова, иностранные слова и числа;

$$T \setminus (S_2 | S_3 | C) = \{t_i \in T | t_i \notin S_2 | t_i \notin S_3 | t_i \notin C\}, i = \overline{0, n}, \quad (8)$$

где T – исходный текст.

S_2 – множество, включающее в себя все существующие стоп-слова русского алфавита,

S_3 – множество, включающее в себя иностранные слова,

C – множество комплексных чисел,

t_i – символ из множества T ,

n – число слов в тексте.

- 4) необходимо привести слова к их начальной форме – выполнить лемматизацию.

Для дальнейшей векторизации текста будет применяться мера TFIDF:

$$TFIDF = TF * IDF, \quad (9)$$

где TF – это численное значение вхождения заданного слова в текущем документе,

IDF – это численное значение, показывающее с какой частотой данное слово, встречается во всех исходных документах.

$$TF = \frac{t_i}{\sum t_k}, \quad (10)$$

где t_i – количество вхождений данного слова,

t_k – общее число слов в тексте.

$$IDF = \log \left(\frac{D}{d_i} \right), \quad (11)$$

где D – общее количество всех текстов,

d_i – тексты, в которых встречается данное слово.

3.3 Методы анализа графической информации

Человек и компьютер по-разному видят то, что происходит на изображении. Человек понимает семантику изображения, способен различать объекты и их характеристики, а компьютер различает лишь цвета пикселей на экране.

В современном мире при анализе изображений чаще всего задействуют машинное обучение. Искусственная нейронная сеть – это одно из наиболее популярных решений при анализе изображений. Такая сеть с лёгкостью способна распознавать объекты на изображениях, но пока не способна понимать, что происходит на изображении, например, скучно ли студентам на лекции.

Выделение и анализ графических признаков можно определить, как большую и самостоятельную задачу, для решения которой могут быть применены самые разнообразные подходы, и для данного проекта целесообразнее всего использовать методы машинного обучения при анализе изображений.

3.4 Нейронные сети: обоснование применения

Человеческий мозг – это сложное биологическое устройство, изучение которого ведется и по сей день. Он наделен великолепной нейронной сетью.

Биологическая нейронная сеть устроена довольно сложно, и до сих пор человеку не удалось узнать все ее тайны. Именно поэтому ее работа была смоделирована в виде искусственной нейронной сети, и все это для решения

единственного вопроса: как совокупность нейронов позволяет человеку думать, принимать решения и воспринимать окружающий мир?

3.4.1 Основные понятия

Искусственный нейрон – основной элемент искусственной нейронной сети, который моделирует работу биологического нейрона.

Рассмотрим строение биологического нейрона. Каждый нейрон имеет отростки двух видов – дендриты и аксон. Дендриты принимают входные импульсы от других соседних нейронов, а аксон передает импульс из самого нейрона другим. На конце аксона есть специальные волокна, которые взаимодействуют с дендритами других нейронов и при этом могут изменять величину импульса.

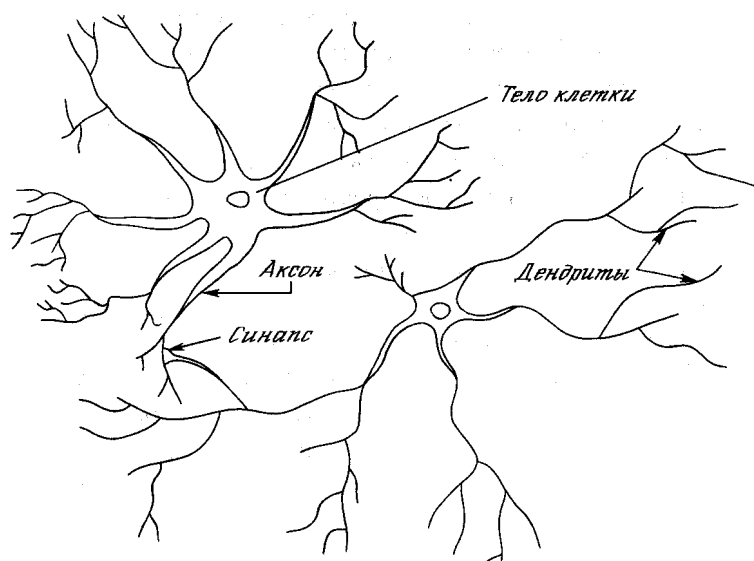


Рисунок 5 - Биологический нейрон

Весь секрет работы нейрона кроется внутри его ядра или тела. Благодаря внутренним электрохимическим реакциям, протекающим в ядре, нейрон может обрабатывать информацию.

Искусственный нейрон имитирует работу биологического нейрона. На вход искусственного нейрона поступают сигналы с других соседних нейронов.

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

24

Каждый вход имеет свой собственный вес. После некоторого преобразования внутри нейрона на его выходе появляется его собственный сигнал. Этот сигнал распространяется дальше к другим нейронам.

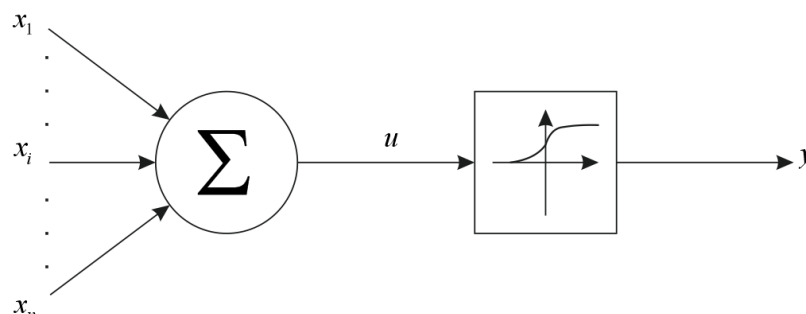


Рисунок 6 - Искусственный нейрон

Это модель закладывает лишь внешние принципы работы искусственного нейрона, но главный секрет его работы будет находится внутри ядра. В проектировании ядра кроется вся сложность – построение логики работы нейрона.

Искусственная нейронная сеть (ИНС) – это математическая модель. Элементом или ячейкой такой модели является искусственный нейрон. Сама же ИНС – система из множества отдельных нейронов, соединенных связями. Каждый нейрон – это черный ящик, в который поступает информация, обрабатывается и выходит наружу.

Если сравнивать «традиционные» модели и ИНС, то главным различием станет «способность к обучению». Под обучением понимается то, как модель будет реагировать на данные, которые раньше не вводились в модель. ИНС позволяют получать правильные ответы при ранее неизвестных данных.

ИНС может иметь реализацию в виде программного кода, либо же в виде устройства.

3.4.2 Архитектура нейронных сетей

Как уже говорилось ранее ИНС состоят из искусственных нейронов, но нигде не уточнялось как должны быть организованы эти нейроны, какую структуру поддерживать.

Рассмотрим основные классификации нейронных сетей, на основе которых определяется архитектура искусственных нейронных сетей.

- однослойная структура;

Такая структура включает в себя лишь один слой нейронов. Сигналы со входного слоя направляются на обрабатывающий выходной слой.

- многослойная структура.

Многослойная нейронная сеть имеет помимо входного и выходного слоя, некоторое количество скрытых слоев. Число этих слоёв зависит от степени сложности нейронной сети. С увеличением слоев растёт и сложность самой сети, но при этом в процессе обработки информации каждый промежуточный слой позволяет дополнительно обработать входные сигналы.

Кроме количества слоёв, ИНС можно классифицировать по направлению распределения информации по синапсам между нейронами.

- нейронные сети прямого распространения (однаправленные);

При такой организации входной сигнал движется только в одну сторону и однаправленно. Движение сигнала в обратном направлении не осуществляется и в принципе невозможно.

- рекуррентные нейронные сети (с обратными связями);

При такой организации входной сигнал движется и в прямом, и в обратном направлении, следовательно, результат выхода может иметь возможность вернуться на вход. Такие сети обладают свойством кратковременной памяти, сигналы способны восстанавливаться во время их обработки.

- радиально-базисные функции;

Радиально базисные функции – это вид нейронной сети, который имеет скрытый слой из радиальных элементов. Такие элементы используют радиальные базисные функции как функции активации.

- самоорганизующиеся карты.

Сеть принципиально отличаются от рассмотренных выше, поскольку используют неконтролируемое обучение, при таком обучении обучающее множество состоит лишь из значений входных переменных. При обучении нет эталонных значений, с которыми могли бы сравниваться выходы, такая сеть учится понимать структуру данных сама.

Нейронных сетей также различаются по способу обучения:

- 1) обучение с учителем – на каждый входной набор есть верный выходной набор данных;
- 2) обучение без учителя – задается только входной набор данных, на его основе необходимо определить некоторый выходной набор;
- 3) обучение с подкреплением – на каждый входной набор есть верный выходной набор данных, обучение происходит в некоторой среде.

Рассмотренные классификации позволяют выделить основные архитектурные решения. Некоторые из них будут рассмотрены далее.

3.4.3 Функции активации

В нейронных сетях функция активации определяет выходной сигнал. Эта функция относится к классу непрерывных функций и принимает на входе произвольное вещественное число, а на выходе дает число в некотором интервале, чаще всего от 0 до 1. Большие (по модулю) отрицательные числа превращаются в ноль, а большие положительные – в единицу.

Рассмотрим возможные функции активации:

- 1) ReLU – линейный выпрямитель;

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

27

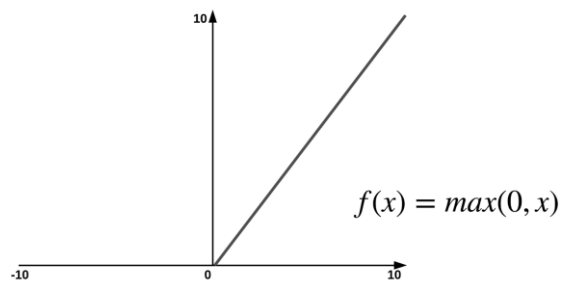


Рисунок 7 - Функция ReLU

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (12)$$

2) гиперболический тангенс;

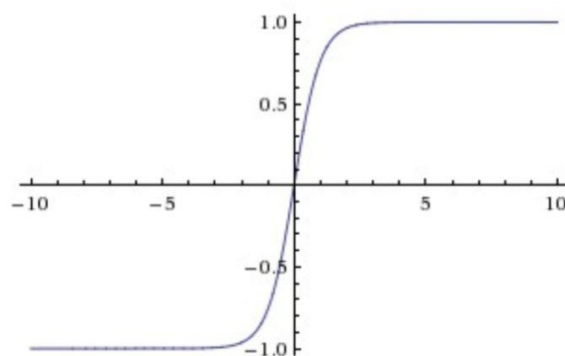


Рисунок 8 - Функция гиперболического тангенса

$$f(x) = th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (13)$$

3) сигмоидальная функция;

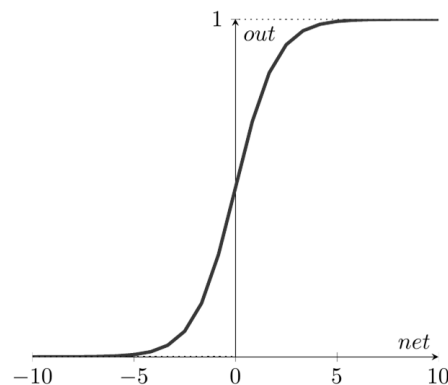


Рисунок 9 - Сигмоидальная функция

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \quad (14)$$

4) функция Softmax.

Softmax функция – это функция, которая принимает в качестве входных данных вектор и нормализует его в распределение вероятностей.

До применения Softmax функции некоторые элементы вектора могут быть как положительными, так и отрицательными, но после применения каждый элемент будет в интервале $[0,1]$ и в сумме будут равны 1, поэтому они могут быть интерпретированы как вероятности.

Кроме того, более крупные входные элементы вектора будут соответствовать большим вероятностям.

$$f(x) = \sigma(y)_i = \frac{e^{y_i}}{\sum_{j=1}^K e^{y_j}}, \text{ где } i = \overline{1, K}, Y^K = (y_1, \dots, y_K) \quad (15)$$

3.4.4 Классическая полносвязная нейронная сеть

Персептрон – это простейшая ИНС. Впервые он был предложен в 1957 году нейрофизиологом Фрэнком Розенблаттом, а уже в 1960 году успешно реализован и применен в нейροкомпьютере – «Марк-1».

По Розенблатту персептрон состоит из трёх типов элементов:

- сенсорный элемент (S-элемент) – чувствительный элемент, который при наличии определенного воздействия какой-либо энергии вырабатывает дискретный сигнал. Входной сигнал проходит через пороговую функцию и если он превышает порог, то элемент выдаёт сигнал «+1», иначе сигнал «0»;
- ассоциативный элемент (А-элемент) – логический решающий элемент. Все входные сигналы, поступающий на него, складываются и проходят через пороговую функцию. Если порог будет превышен, то элемент

считается активированным и сигнал устанавливается в «+1», иначе сигнал устанавливается в «0»;

- реагирующий элемент (R-элемент) – элемент, который выдаёт выходной сигнал «+1», если сумма всех входных сигналов является строго положительной, и сигнал «-1», если сумма всех входных сигналов – строго отрицательная. Если получается, что сумма входных сигналов равна нулю, то выход можно считать либо равным нулю, либо неопределённым.

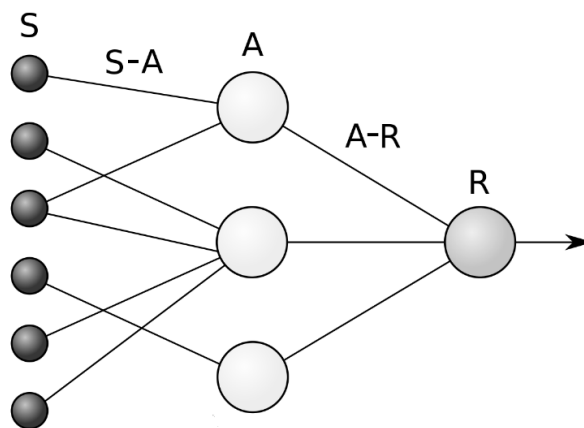


Рисунок 10 - Персептрон по Розенблатту

Детально работа такой модели выглядит следующим образом: для всех S-элементов определяется состояние, в котором они находятся. Они могут быть либо возбуждены (сигнал «1»), либо находиться в состоянии покоя (сигнал «0»). Сигналы с S-элементов поступают в A-элементы по S-A связям. Веса на S-A связях могут принимать одно из значений $\{-1, 0, +1\}$. A-элементы принимают сигналы от нескольких S-элементов и обрабатывают их: если сумма всех сигналов превышает определенный заданный порог, то этот A-элемент активируется и выдает сигнал «+1», иначе выдает сигнал «0». Сигналы с A-элементов поступают в R-элементы по A-R связям. R-элемент принимают сигналы от нескольких A-элементов и складывает их друг с другом: если

превышен определенный порог, то генерирует сигнал «+1», если порог не превышен, то сигнал «-1». R-элемент определяет выход персептрона в целом.

Однослойный персептрон по Розенблатту – персептрон, каждый S-элемент которого однозначно соответствует одному A-элементу, S-A связи всегда равны 1, а порог любого A-элемента равен 1.

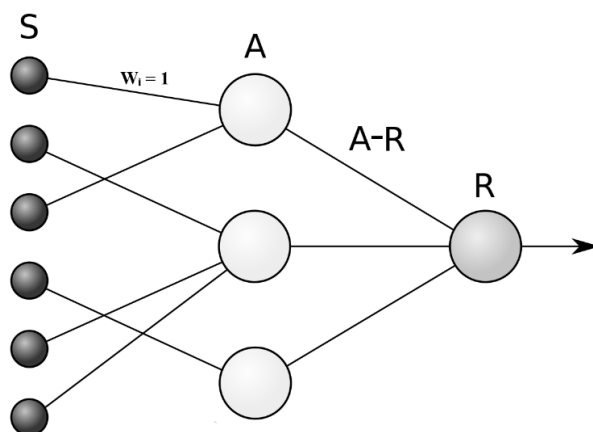


Рисунок 11 - Однослойный персептрон по Розенблатту

Если убрать S-элементы и S-A связи, то можно получить обычный искусственный нейрон с единственным отличием: у однослойного персептрона входные сигналы могут принимать только фиксированные значение – 0 или 1, а у искусственного нейрона – любые значения.

Многослойный персептрон по Розенблатту – персептрон, у которого имеется более 1 слоя A-элементов

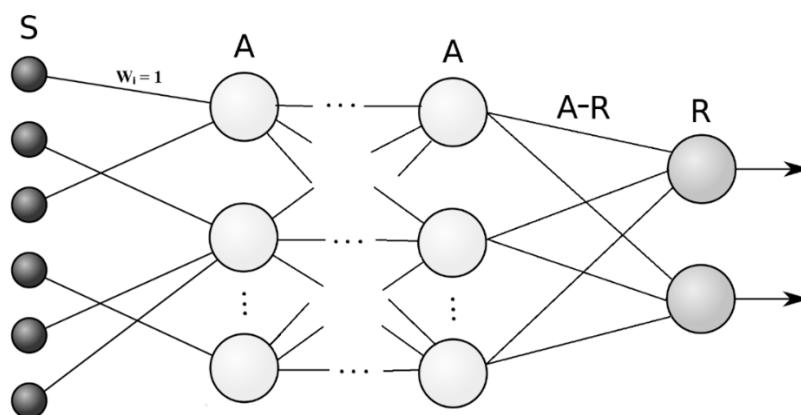


Рисунок 12 - Многослойный персептрон по Розенблатту

Полносвязные искусственные нейронные сети (ПИНС) прямого распространения хранят идеи персептрона, но имеют ряд некоторых отличий:

- искусственная нейронная сеть состоит из искусственных нейронов, а персептрон состоит из S-, A- и R-элементов;
- в персептроне S- и A-элементы могут принимать только фиксированные значения – 0 или 1, тогда как в искусственном нейроне ограничений на входные сигналы нет.

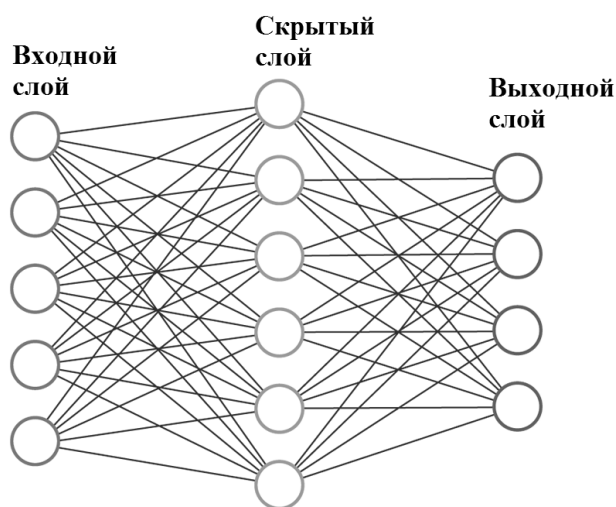


Рисунок 13 - Полносвязная искусственная нейронная сеть

3.4.5 Сверточная нейронная сеть

Сверточная нейронная сеть (СНС) состоит из трех основных видов слоев: сверточный слой, субдискретизирующий слой (слой подвыборки) и выходной слой (полносвязный слой).[6,7,8]



Рисунок 14 - Архитектура сверточной нейронной сети

В основе сверточного слоя лежит операции свертки изображению. Свертка представляется как двумерная матрица коэффициентов. Каждый фрагмент изображения поэлементно умножается на небольшую матрицу весов, результат подвергается операции суммирования. На вход сверточного слоя поступает фрагмент двумерного изображения, а на выходе – некоторое число, значение которого тем больше, чем больше фрагмент изображения похож на применяемый к нему фильтр. После обработки всех фрагментов изображения будет сформирована так называемая карта признаков.

Между двумя слоями свертки чаще всего применяют слой субдискретизации. Субдискретизация необходима тогда, когда на предыдущем сверточном слое были выявлены некоторые признаки, поэтому нет необходимости хранить изображение столь подробно и следует уплотнить его. Выбирается некоторая матрица (обычно размера 2×2) и уплотняется до одного пикселя. В большинстве случаев применяется функция максимума для матрицы.

Чередование слоёв позволяет составлять карты признаков из карт признаков. На каждом следующем слое карта уменьшается в размере, что позволяет распознавать сложные иерархии признаков. Обычно после прохождения нескольких слоёв карта признаков вырождается в вектор или даже в скаляр.

На выходе последнего сверточного слоя в сети обычно устанавливают несколько слоёв полносвязной нейронной сети, на вход которой подаются карты признаков.

3.4.6 Сравнение двух моделей

В предшествующих разделах были рассмотрены две наиболее популярные архитектуры ИНС – полносвязная искусственная нейронная сеть прямого распространения и сверточная нейронная сеть.[7]

В первой нейронной сети каждый нейрон связан с каждым нейроном последующего слоя. Сеть с такой архитектурой можно использовать для анализа изображений следующим образом: двумерную матрицу изображения представить в виде одномерного вектора и загрузить на вход, результатом будет некоторый вектор, представляющий текст.

Исходя из того, что изображение свернуто в вектор, сразу можно сформулировать две проблемы обработки изображения на ПНС:

- большое количество вычислений при прямом распространении: наличие большого входного вектора, и как правило, наличие еще большего последующего скрытого слоя;
- большое количество вычислений при обратном распространении ошибки из-за большого количества связей между нейронами.

Вторая архитектура – сверточная нейронная сеть. Такая архитектура нацелена на эффективное распознавание образов. Она включает в себя следующие слои:

- сверточный слой представляет набор матриц. Каждая матрица представляется как один нейрон, реализующий операцию свертки.;
- слой субдискретизации уплотняет (усредняет) матрицу, уменьшает ее размерность;
- полносвязный слой – классическая ПНС.

Сравнивая эти две архитектуры, можно сказать, что у сверточной нейронной сети гораздо меньшее количество настраиваемых весов, так как одно ядро весов используется целиком для всего изображения. А из-за того, количество настраиваемых весов уменьшилось, обучается сеть быстрее.

Из недостатков стоит отметить увеличение глобальных настраиваемых параметров сети. К таким параметрам можно отнести количество слоёв, размерность ядра свёртки на каждом слое, шаг сдвига ядра при обработке, наличие или отсутствие слоев субдискретизации, функция слоя

субдискретизации, наличие и параметры выходного полносвязного слоя. Все эти параметры влияют на результат, и выбираются исследователями индивидуально для каждой задачи.

При рассмотрении двух видов нейронных сетей однозначно стоит отметить преимущество сверточной нейронной сети для решения задачи обработки изображения.

3.5 Математическая модель сверточной нейронной сети

В предыдущем разделе было установлено, что сверточная нейронная сеть будет лучшим решением для работы с изображениями, поэтому необходимо рассмотреть архитектуру сверточной нейронной сети с точки зрения математических формул.

3.5.1 Математическая модель сверточного слоя

Предположим, что на вход слоя свертки поступает матрица $H_1 \times H_2$, матрица весов (фильтр) представляет собой матрицу $h_1 \times h_2$, тогда размер выходной карты признаков:

$$(H_1 - h_1 + 1) \times (H_2 - h_2 + 1) \quad (16)$$

Определим операцию свертки I для фрагмента изображения Y (размером $h_1 \times h_2$) и матрицей фильтра W (размером $h_1 \times h_2$):

$$I = f(Y, W), \quad (17)$$

$$f(Y, W) = f_{act} \left(\sum_{i=0}^{h_1} \sum_{j=0}^{h_2} y_{ij} * w_{i,j} + b \right), \quad (18)$$

где y_{ij} – элемент фрагмента изображения,

$w_{i,j}$ – элемент матрицы фильтра,

b – пороговый коэффициент,

$f_{act}()$ – функция активации.

Определим матрицу результата R_{out} сверточного слоя на основе операции свертки I и входной матрицы R_{in} :

$$R_{out} = \begin{bmatrix} r_{11} & \cdots & r_{1H_2} \\ \vdots & \ddots & \vdots \\ r_{H_11} & \cdots & r_{H_1H_2} \end{bmatrix}, r_{i,j} = I(Y_{kl}, W), Y_{kl} \in R_{in}, \quad (19)$$

где Y_{kl} – фрагмент изображения размером $h_1 \times h_2$, $k = \overline{1, (H_1 - h_1 + 1)}, l = \overline{1, (H_2 - h_2 + 1)}$ определяют левый верхний угол матрицы, W – матрица фильтра размером $h_1 \times h_2$.

3.5.2 Математическая модель слоя субдискретизации

На слое субдискретизации происходит уплотнение матрицы R_{in} размером $H_1 \times H_2$ за счет функции $\max \begin{pmatrix} x_{11} & x_{1m} \\ x_{n1} & x_{nm} \end{pmatrix}$, которая находит максимальное значение в матрице:

$$R_{out} = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nm} \end{bmatrix}, r_{i,j} = f_{act}(\max(Y_{kl}) + b), Y_{kl} \in R_{in}, \quad (19)$$

где Y_{kl} – фрагмент изображения размером $p_1 \times p_2$, $k = \overline{1, (H_1 - p_1 + 1)}, l = \overline{1, (H_2 - p_2 + 1)}$ определяют левый верхний угол матрицы,

W – матрица фильтра размером $h_1 \times h_2$,

b – пороговый коэффициент,

$f_{act}()$ – функция активации.

3.5.3 Математическая модель полносвязного слоя

Полносвязный слой представляет собой многослойную искусственную нейронную сеть. Вначале рассмотрим модель однослойной искусственной нейронной сети.

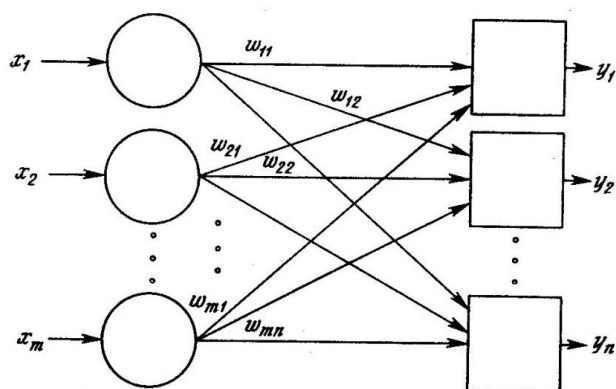


Рисунок 15 - Модель однослойного персептрона

Каждый вход x_m связан с каждым нейроном через связь, которая имеет определённый вес w_{mn} , где m – количество входов, n – количество нейронов. Выход y_n для каждого нейрона вычисляется по формуле:

$$y_n = f_{act} \left(\sum_{i=1}^m (w_{in} * x_i) - b_n \right), \quad (20)$$

где x_i – входной сигнал,

w_{in} – весовой коэффициент для i -ой связи n -ого нейрона,

b_n – пороговый коэффициент,

$f_{act}()$ – функция активации.

Таким образом, выходом нейронной сети является вектор:

$$Y = [y_1, y_2, \dots, y_n], \quad (21)$$

где n – количество нейронов.

В многослойной искусственной нейронной сети каждый последующий слой связан со всеми нейронами предыдущего слоя.

Обозначим входной вектор X как вектор Y^0 , все выходы на последующих слоях – Y^p и весовые коэффициенты на каждом слое как матрицу W^p , где p – количество слоев в сети. Выход последнего слоя будет считаться выходом всей многослойной нейронной сети, и рассчитываться по формуле:

$$Y^p = \langle Y^{(p-1)}, W^p, B^p \rangle, \quad (22)$$

где $Y^p = [y_1^p, \dots, y_n^p]$ – вектор входных или выходных сигналов слоя,

$$W^p = \begin{bmatrix} w_{11}^p & \dots & w_{1n}^p \\ \vdots & \ddots & \vdots \\ w_{m1}^p & \dots & w_{mn}^p \end{bmatrix} \text{ – матрица весов для } p\text{-ого слоя,}$$

$B^p = [b_1^p, \dots, b_n^p]$ – вектор пороговых коэффициентов.

Каждый сигнал y_n^p вычисляется по формуле, рассмотренной для однослойной искусственной нейронной сети:

$$y_n^p = f_{act} \left(\sum_{i=1}^m (w_{in}^p * x_i^p) - b_i^p \right), \quad (23)$$

3.5.4 Обучение сверточной нейронной сети

Основной целью обучения является минимизация функции ошибки.

Наиболее простым и популярным способом обучения является метод обучения с учителем – метод обратного распространения ошибки. Тогда, функция ошибки определяется как:

$$E^q = \frac{1}{2} \sum_j (d_j^q - y_j^q)^2, \quad (24)$$

где E^q – величина ошибки для входного вектора $Y^{(p-1)}$,

d_j^q – желаемый выход нейрона j для входного вектора $Y^{(p-1)}$,

y_j^q – полученный выход Y^p нейрона j для входного вектора $Y^{(p-1)}$.

Изменение весовых коэффициентов и пороговых коэффициентов будет происходить по следующим формулам:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \frac{\partial E}{\partial w_{ij}(t)}, \quad (25)$$

$$b_{ij}(t+1) = b_{ij}(t) - \alpha \frac{\partial E}{\partial b_{ij}(t)}, \quad (26)$$

где w_{ij} – весовой коэффициент j -ого нейрона i -ого слоя,

b_{ij} – пороговый коэффициент j -ого нейрона i -ого слоя,

t – номер итерации,

α – коэффициент инерциональности, отвечающий за скорость обучения сети,

$\frac{\partial E}{\partial w_{ij}(t)}, \frac{\partial E}{\partial b_{ij}(t)}$ – градиент функции ошибки для i -ого слоя.

Рассмотрим алгоритм обратного распространения ошибки:

- 1) зададим произвольные значения весовых коэффициентов;
- 2) передадим сигнал в сети в прямом направлении;
- 3) вычислим значение функции ошибки E^q ;
- 4) распространим значение ошибки E^q , полученной на шаге 3) в обратном направлении, то есть от последнего слоя к предыдущим. Выполним корректировку весовых коэффициентов w_{ij} и пороговых коэффициентов b_{ij} каждого нейрона с использованием градиентного спуска.

4 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ

Данная глава посвящена проектированию всей программы. В главе будет рассмотрена реализация моделей и инструментов, рассмотренных в предыдущих разделах, их взаимодействие, а также рассмотрена общая структура программы.

Также будут освещены следующие вопросы: с помощью каких средств будут реализовываться инструменты, модели и итоговая программа, процесс обучения нейронной сети, и ее применение.

4.1 Разработка общей структуры программы

Разработка всей программы строится на вокруг модуля, реализующего модель сверточной нейронной сети.

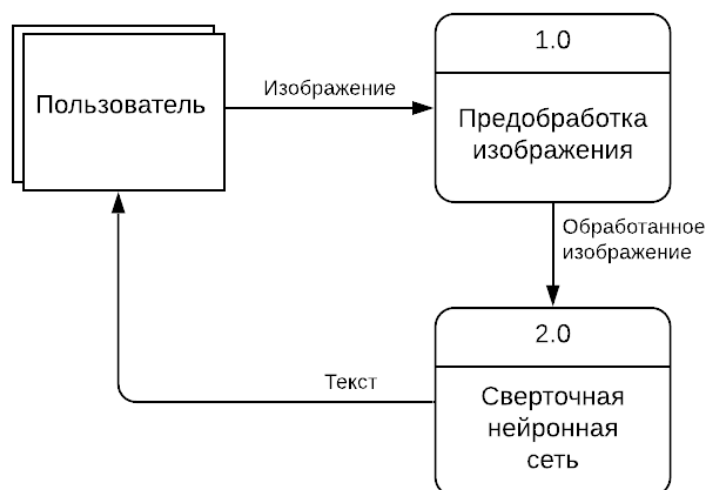


Рисунок 16 - Структура программы

Программа получает на вход пользовательское изображение, для которого необходимо сгенерировать заголовок. Изображение проходит предобработку: оно сжимается до размеров, пригодных к обработке внутри модели нейронной сети, преобразуется в массив чисел и проходит нормализацию – каждое число нормируется в интервале от 0 до 1. Только после этого изображение поступает на вход модели.

Модель нейронной сети обрабатывает полученный массив и в качестве результата выдает вектор слов. Этот вектор можно интерпретировать как набор слов, связанных с этим изображением.

На данном этапе можно интерпретировать полученный набор слов как некоторый текст, потому что без практических испытаний не понятно насколько точным окажется результат. Предполагается, что нейронная сеть должна выдавать результат, в котором текст как минимум будет состоять из одного слова.

4.2 Средства реализации модулей системы

Все модули системы будут разрабатываться на языке Python из-за наличия большой базы библиотек, упрощающих весь процесс разработки.

Для отправки HTML-запросов существует немало python-библиотек, такие как urllib/urllib2 и Requests. Кроме того, одной из главных задач при разработке алгоритмов является выбор библиотеки для разбора HTML-документа. Для решения этой задачи существуют следующие библиотеки:

- 1) библиотека регулярных выражений – re;

Регулярные выражения – мощный инструмент при разборе каких-либо строк или текстов, но существуют более удобные инструменты, которые внутри основываются на регулярных выражениях.

- 2) специализированные библиотеки для разбора HTML-документов – BeautifulSoup и lxml;

BeautifulSoup и lxml – это две популярные библиотеки для разбора HTML-документов. Обе библиотеки предоставляют возможность разложения полученного HTML-документа в DOM-дерево. Выбор между одной из них остается задачей разработчика.

- 3) фреймворк – scrapy.

Scrapy – это целый open-source framework для сбора данных с различных ресурсов в интернете. У него есть множество достоинств: асинхронные запросы,

интеграция с библиотекой XPath, CSS селекторы для обработки данных, удобная работа с кодировками и т.д. Подходит для серьезных крупных проектов.

Для задачи сбора данных остановимся на следующем стеке библиотек: Requests, BeautifulSoup и CSV.

Модель векторизации текста состоит из двух частей: предобработка данных и модель для векторизации текста на основе меры TF-IDF.

Инструмент по предобработке данных будет разработан с использованием библиотеки nltk, которая заточена под анализ и обработку естественного языка. Для морфологического анализа русского языка будет использоваться анализатор pymorphy2, потому что в отличии от nltk больше заточен под работу с русским языком и способен определять сложные словоформы. В набор его возможностей входит:

- 1) возможность приводить слово к нормальной форме (например, «люди – человек»). Под нормальной формой понимается именительный падеж, единственное число для существительных, инфинитив для глаголов, именительный падеж, единственное число, мужской род для прилагательных и так далее для других частей речи;
- 2) возможность поставить слово в определенную форму, например, во множественное число или изменить падеж и так далее;
- 3) возможность возвращать грамматическую информацию о слове (число, род, падеж, часть речи и так далее).

Готовую модель на основе меры TF-IDF можно взять из открытой библиотеки sklearn и обучить ее на полученных обработанных данных. После создания модели следует сохранить ее, чтобы в дальнейшем не тратить дополнительное время на ее построение. Для этого необходимо сериализовать модель. Python предоставляет такую библиотеку как pickle.

Для реализации второго инструмента будет использоваться популярная открытая библиотека для нейронных сетей TensorFlow от Google. Библиотека

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

42

позволяет решать задачи построения и обучения моделей нейронных сетей. TensorFlow может работать как на CPU, так и на GPU и TPU.

Для взаимодействия с TensorFlow можно использовать высокоуровневое API – библиотеку Keras. Keras позиционируется как библиотека, позволяющая быстро создать прототип нейронной сети и исследовать архитектуру для определения искомых гиперпараметров.

4.3 Сбор и подготовка данных для обучения нейронной сети

Данный раздел будет посвящен разработке основных алгоритмов для сбора коллекции данных и последующей реализации их на языке Python. Также будут рассмотрены результаты сбора, по ним будет проведен предварительный анализ полученных данных.

4.3.1 Разработка алгоритмов для сбора данных

Для решения поставленной задачи необходимо разработать эффективный алгоритм с точки зрения затрат времени.

Исходя из поставленной задачи алгоритм должен будет собирать большие объемы информации. Можно утверждать, что время работы линейно зависит от сложности алгоритма.

Для решения поставленной задачи был разработан следующий алгоритм, представленный ниже.

На вход программы (сценария) поступает ссылка на обрабатываемый новостной ресурс и задаются необходимые параметры для сбора данных. Инициализируется цикл, который повторяется пока существует адрес следующей страницы, и выбираемая статья находится в рамках временного промежутка. Во время цикла выполняется следующая последовательность действий:



Рисунок 17 - Общий алгоритм решения задачи сбора данных с новостного сайта «РИА Новости»

- 1) отправляется запрос по URL адресу;
- 2) загружается HTML-разметка;
- 3) выполняется разбор HTML-дерева с использованием одной из вышеописанных библиотек, результатом является список с данными;
- 4) выполняется попытка найти следующий адрес, при нахождении составляется новый адрес для запроса следующего HTML-документа.

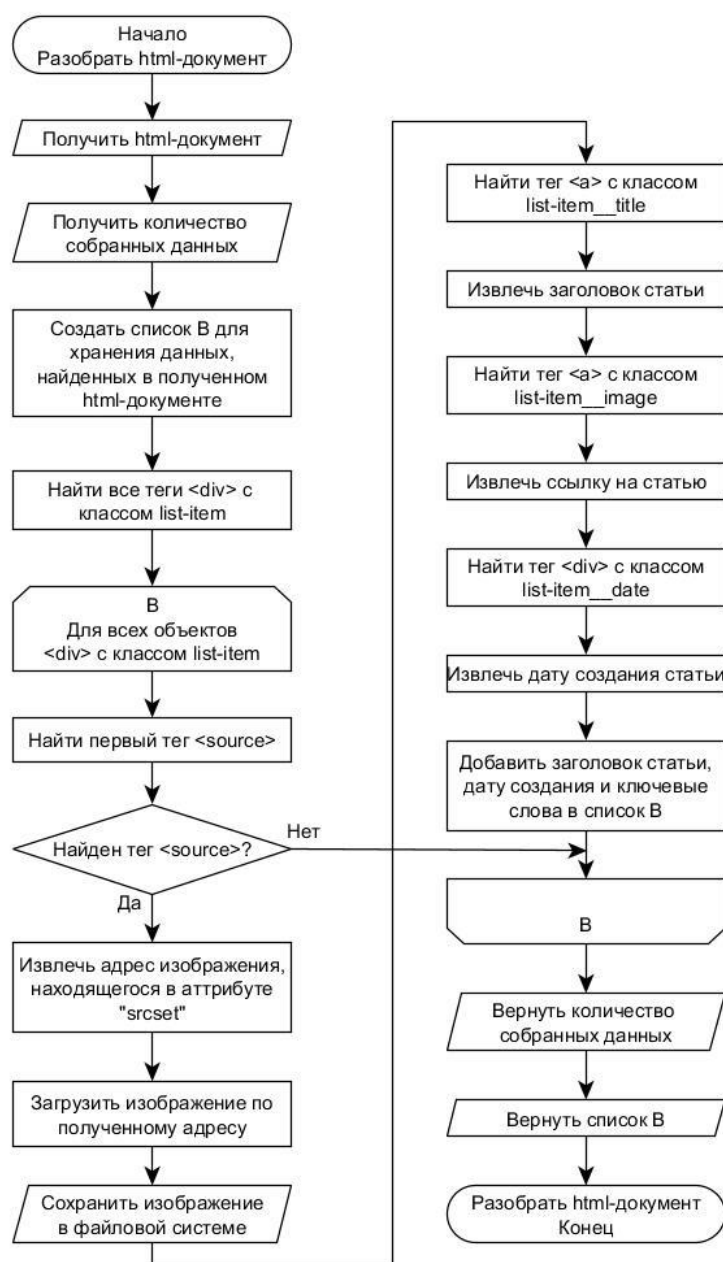


Рисунок 18 - Алгоритм разбора HTML-документа

Функция разбора HTML-документа выполняет синтаксический разбор поступившей информации. После нахождения начала разметки, соответствующей первой новостной статье, в цикле происходит последовательная обработка каждой потенциальной новостной статьи по заданным критериям. Если статья соответствует критериям, она помещается в результирующий список. По прошествии нескольких итерации список записывается в CSV-файл.

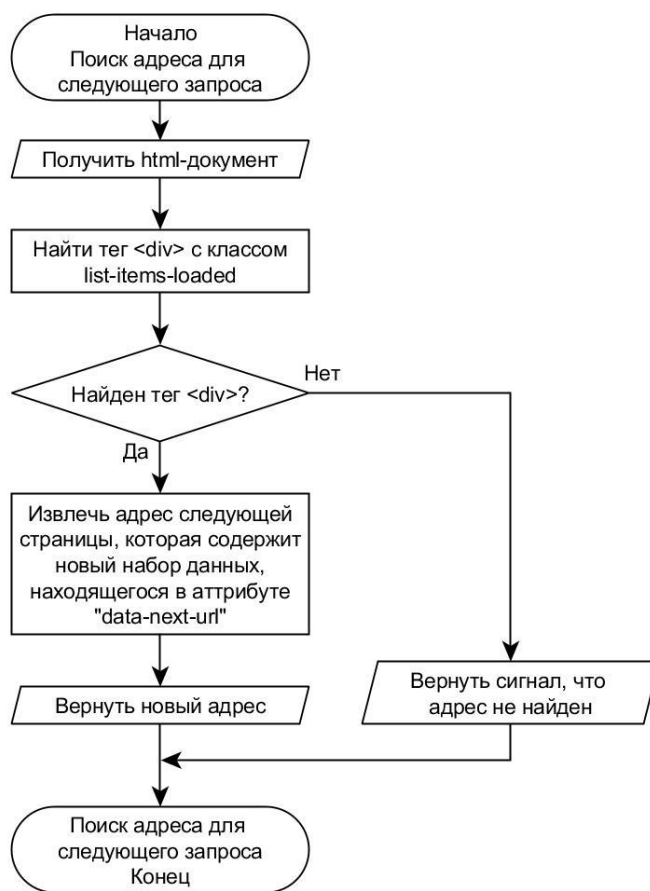


Рисунок 19 - Алгоритм поиска следующего адреса запроса

Функция нахождения адреса следующего запроса находит нужный тэг среди всего документа и просматривает его свойства. Если адрес следующей страницы с новостными статьями будет найден, то он будет записан, и внешний цикл продолжит работу по сбору данных, а иначе будет установлен сигнал отсутствия следующего адреса.

4.3.2 Результаты сбора данных

После того как программа была разработана и реализована, можно инициализировать её постоянную работу. Для первого полноценного сбора были взяты категории «Политика» и «Общество», так как содержащиеся там статьи и изображения имеют значительные различия по содержанию. Кроме того, в разделе «Общество» на каждый год приходится в среднем в 3 раза больше новостных статей, чем в разделе «Политика».

После двух дней работы программы были получены 10 таблиц в формате CSV. Каждая таблица имеет название, соответствующее новостному разделу и году, по которому были собраны данные.

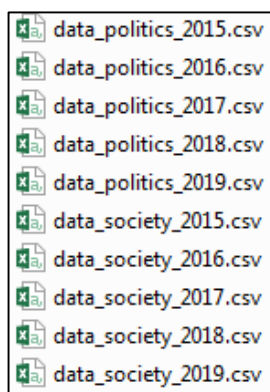


Рисунок 20 - Полученные таблицы данных

Каждой заполненной CSV-таблице соответствует единственная папка в локальном каталоге с соответствующим названием.

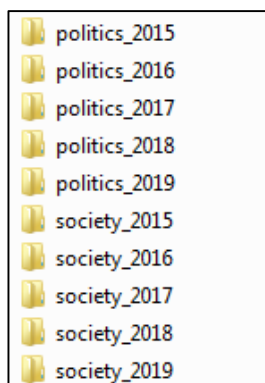


Рисунок 21 - Каталоги, содержащие изображения к таблицам

4.3.3 Подготовка данных для обучения нейронной сети

После того, как были собраны основные коллекции данных с новостного сайта «РИА Новости» по разделам «Политика» и «Общество», необходимо подготовить данные для обучения нейронной сети.

После рассмотрения полученной информации по разделу «Политика», было выявлено множество изображений, содержащих крупные планы известных политических личностей. Необходимо провести фильтрация всего набора изображений с целью уменьшения количества изображений, содержащих крупные планы.

Для решения подзадачи необходимо разработать алгоритм, который должен:

- 1) получить каталог с располагающимися в нем изображениями;
- 2) обработать каждое полученное изображение: определить есть ли на изображении лицо, снятое крупным планом;
- 3) создать новую CSV-таблицу и каталог с сохраненными изображениями.

При реализации ключевую роль играет функция по распознаванию лица на изображении. Было принято решение взять готовый классификатор, реализованный в библиотеке OpenCV.[3] Классификатор основан на методе Виолы-Джонса (Viola-Jones) и придерживается следующих принципов:

- 1) изображение используется в интегральном представлении (позволяет быстро рассчитать суммарную яркость произвольного прямоугольника на изображении; представляет собой матрицу, совпадающую по размерам с исходным изображением, где в каждом элементе хранится сумма интенсивностей всех пикселей, находящихся левее и выше данного элемента);
- 2) используются признаки Хаара (прямоугольные признаки, которые называются примитивами Хаара);

- 3) используется бустинг (процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов);
- 4) используется классификатор (функция, определяющая к какому классу, принадлежит объект);
- 5) используются каскадная модель (процедура последовательного применения различных признаков Хаара).

Результатом работы данного классификатора является набор прямоугольников, описывающих распознанные на изображении лица.

На вход программы (рисунок 22) поступают: относительный путь до конфигурации классификатора и относительный путь к CSV-таблице. Благодаря CSV-таблице определяется относительный путь до каждого изображения. Инициализируется цикл обхода всех изображений. В цикле выполняется следующая последовательность действий:

- 1) загружается некоторое изображение, согласно строке таблицы;
- 2) на загруженное изображение накладывается фильтр «оттенки серого»;
- 3) инициализируется классификатор по определению лица, снятого крупным планом, на изображении;
- 4) если лицо, снятое крупным планом, не найдено, то строка сохраняется в новую таблицу, а изображение записывается в новый каталог. иначе возвращаемся к пункту 1);
- 5) после выхода из цикла все данные записываются на диск.

После реализации, программа для фильтрации была применена на таблицах, относящихся к разделу «Политика», так как там было больше всего изображений с «крупным планом».

Стоит отметить тот факт, что после фильтрация вся коллекция по разделу «Политика» сократилась вдвое. Это говорит о том, что каждое второе изображение являлось ненужным ресурсом для дальнейшего анализа и могло бы

повлечь дополнительные затраты. Отсюда следует подтверждение необходимости первичной фильтрации полученной информации, так как это позволяет отсеять ненужную для исследования информацию.



Рисунок 22 - Алгоритм первичной фильтрации собранной коллекции данных

Стоит отметить тот факт, что после фильтрация вся коллекция по разделу «Политика» сократилась вдвое. Это говорит о том, что каждое второе изображение являлось ненужным ресурсом для дальнейшего анализа и могло бы повлечь дополнительные затраты. Отсюда следует подтверждение необходимости первичной фильтрации полученной информации, так как это позволяет отсеять ненужную для исследования информацию.

Результаты работы были сохранены в таблицу.

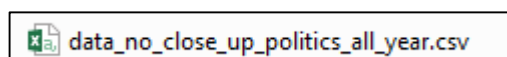


Рисунок 23 - Полученная таблица отфильтрованных данных

Заполненной CSV-таблице соответствует единственная папка в локальном каталоге с соответствующим названием.

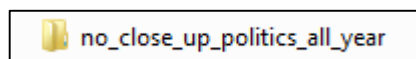


Рисунок 24 - Каталог, содержащий изображения к отфильтрованной таблице

4.4 Разработка инструмента для векторизации текста

Библиотека `rumorphy2` берет на себя часть работы по предобработке данных. Алгоритм предобработки представлен на рисунке 25.

После того как предложения были очищены и представлены в виде массива слов, а слова приведены в нормальную форму, можно перейти к реализации второй части программы – построению модели на основе меры TF-IDF.

Внутри библиотеки `sklearn` модель имеет название `TfidfVectorizer` и инициализируется следующим ключевым параметром: `max_features` – определяет размер итогового словаря, на основе самых часто встречающихся слов во всем входном корпусе. Этот параметр позволит менять размерность вектора в дальнейшем анализе. После инициализации можно загрузить в модель полученный массив строк, модель обучится на них и появится возможность

получить вектор для любого другого предложения, не состоящего во входных данных, при этом единственным условием его векторизации будет соответствующая предобработка.



Рисунок 25 - Алгоритм предобработки строк

После того как предложения были очищены и представлены в виде массива слов, а слова приведены в нормальную форму, можно перейти к реализации второй части программы – построению модели на основе меры TF-IDF.

Внутри библиотеки sklearn модель имеет название TfidfVectorizer и инициализируется следующим ключевым параметром: `max_features` – определяет размер итогового словаря, на основе самых часто встречающихся слов во всем входном корпусе. Этот параметр позволит менять размерность вектора в дальнейшем анализе. После инициализации можно загрузить в модель полученный массив строк, модель обучится на них и появится возможность получить вектор для любого другого предложения, не состоящего во входных данных, при этом единственным условием его векторизации будет соответствующая предобработка.

После создания модели следует сохранить ее, используя функции `pickle`.

Векторизатор текста будет представлен в виде модуля. Это позволит подменить конкретную реализацию алгоритма на другой во время практических испытаний. Поэтому обернем разработанный алгоритм в специальный класс с заданным интерфейсом. Интерфейс включает в себя функцию `get_vector()`, которая получает на вход строку и возвращает вектор, соответствующий этой строке. Диаграмма класса представлена ниже.

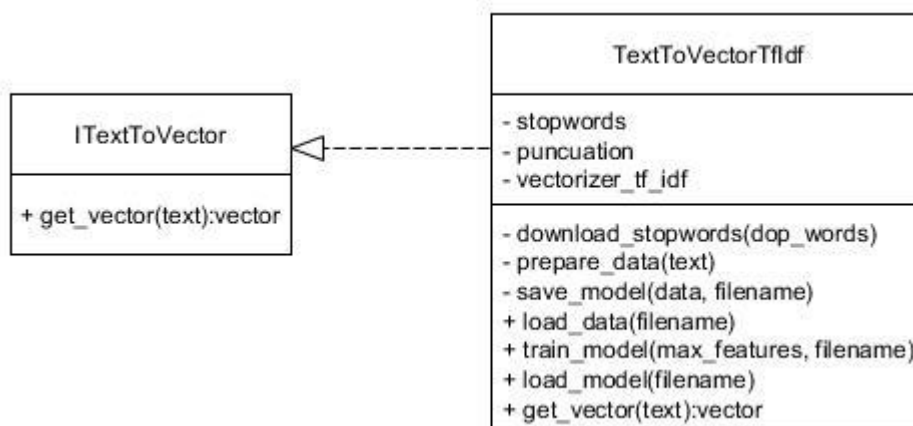


Рисунок 26 - Диаграмма классов

В реализуемом классе присутствуют следующие обязательные публичные функции:

- загрузить данные для построения модели – load_data();
- обучить модель на загруженных данных – train_data();
- загрузить уже готовую модель – load_model().

4.5 Построение модели сверточной нейронной сети. Обучение нейронной сети

Первоначально модель сверточной нейронной сети будет иметь следующую архитектуру.

Таблица 2 - Начальная архитектура сверточной нейронной сети

Слой	Количество нейронов	Ядро	Активационная функция
Сверточный №1	8	2x2	ReLU
Субдискретизирующий №1	–	2x2	–
Сверточный №2	8	2x2	ReLU
Субдискретизирующий №2	–	2x2	–
Сверточный №3	8	2x2	ReLU
Субдискретизирующий №3	–	2x2	–
Выравнивающий	–	–	–
Полносвязный №1	512	–	ReLU
Полносвязный №2	512	–	ReLU
Полносвязный №3	512	–	ReLU

Начальная архитектура сверточной нейронной сети предполагает наличие трех пар «Сверточный слой + Субдискретизирующий слой», одного выравнивающего слоя и трех полносвязных. Выравнивающий слой применяется для соединения слоев, работающих с матрицами, со слоями с векторной

структурой. Здесь данный слой применяется для превращения двухмерного массива в одномерный массив (вектор). Количество нейронов на каждом слое выбрано случайным образом, в дальнейшем их количество будет либо увеличиваться, либо уменьшаться.

Для сверточного слоя №1 предполагается установка размера входной матрицы исходя из размеров изображений, поступающих на вход нейронной сети. Для того, чтобы все изображения могли поместиться в оперативной памяти будет проведено их сжатие до размеров 50x100. В дальнейшем это также будет считаться гиперпараметром.

Для последнего полносвязного слоя всегда будет действовать правило: количество нейронов должно быть равно размерности вектора, используемого для представления текста. В текущем случае размерность вектора начинается с 512 элементов, поэтому количество нейронов на последнем полносвязном слое будет 512.

При обучении будет использоваться средняя квадратичная ошибка как функция потерь, а стохастический градиентный спуск как функция оптимизатора. Метрикой качества будет считаться показатель точности.

Коллекция данных разбивается на тренировочную и тестовую выборку в соотношении 90\10. Обе выборки будут рассматриваться в виде пар «изображение – текст в векторной форме». На вход модель получает изображение, на выходе – числовой вектор. С помощью средней квадратичной ошибки вычисляется погрешность результата от эталонного и производится обратное распространение ошибки с помощью стохастического градиентного спуска.

На этапе подготовки данных все текстовые предложения превращаются в числовые векторы, размерность вектора будет варьироваться в пределах множества {512, 1024, 2048}. Данные значения были выбраны эвристически, с учетом занимаемой памяти, и возможно будет корректироваться в дальнейшем. Также подвергаются обработке соответствующие тексту изображения, они

рассматриваются как трехмерные массивы (координаты X, координата Y, цвет пикселя в режиме RGB), которые необходимо нормализовать. Каждый пиксель переводится из диапазона [0;255] в диапазон [0;1]. В среднем размерность изображения варьируется от 210x410 до 230x430 пикселей.

Для построения моделей нейронных сетей требуются большие вычислительные ресурсы. В дальнейшем будет использоваться платформа Colab, принадлежащая Google.[10] Она поддерживает все необходимые библиотеки необходимые для построения модели и позволяет загружать свои данные для обучения с Google Drive. Это позволяет ускорить обучение моделей, при этом опробовав различные вариации гиперпараметров и внутренней архитектуры.

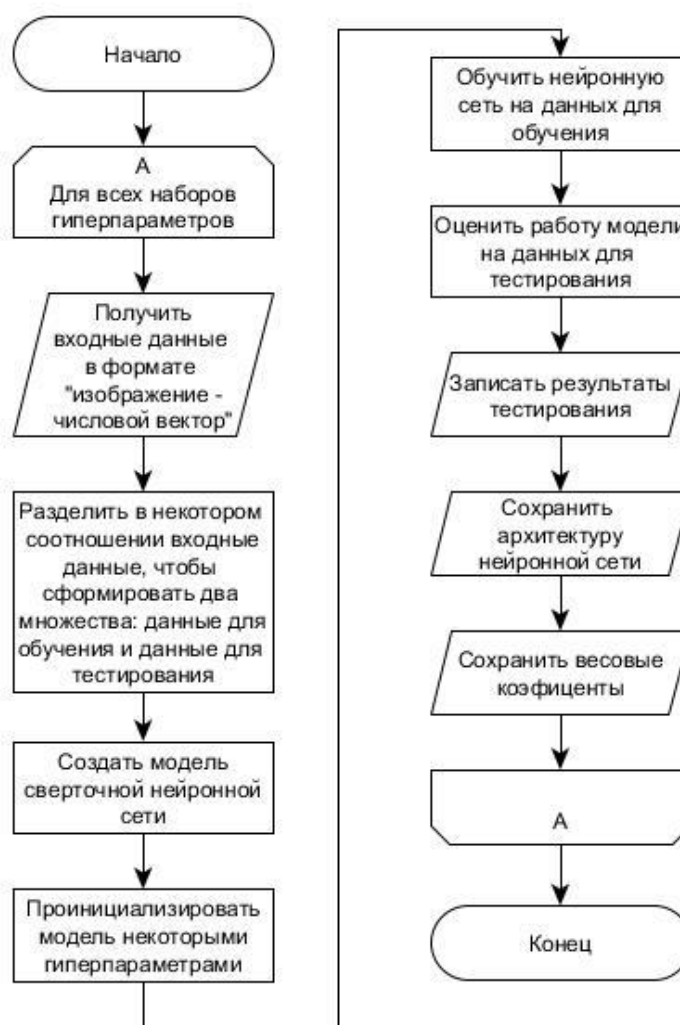


Рисунок 27 - Алгоритм исследования гиперпараметров сверточной нейронной сети

Google Colab предоставляет к использованию GPU Tesla K80 при наличии 25,5Гб оперативной памяти и 68Гб памяти на жестком диске.

В ходе подготовки данных была выявлена серьезная проблема, связанная с нехваткой памяти. Суммарно все обработанные изображения без сжатия занимали бы около 30-35 Гб оперативной памяти, когда в наличии имеется лишь около 25,5 Гб при использовании GPU или при использовании TPU. Исходя из этой ситуации было принято решение сжать каждое изображение. Сжатие изображения также будет варьироваться в ходе обучения, изображение планируется сжимать в 1,5–3 раза в зависимости от других гиперпараметров. Это, несомненно, приведет к потере данных и ухудшению результатов работы модели. Оценить на сколько модель потеряет в точности пока не представляется возможным, для этого необходимо более глубокое исследование с привлечением дополнительных вычислительных ресурсов.

Под управлением API Keras реализуется следующий алгоритм работы программы.

Определим список исследуемых гиперпараметров:

- размерность входного изображения;
- размерность выходного вектора;
- размер подвыборки;
- использовать перетасовку всей выборки;
- количество пар слоев «Сверточный слой + Субдискретизирующий слой»;
- количество полносвязных слоев;
- количество нейронов на каждом сверточном слое;
- количество нейронов на каждом полносвязном слое;
- ядро свертки для сверточного слоя;
- ядро свертки для субдискретизирующего слоя;
- функции активации на сверточных и полносвязных слоях;

- функция потерь;
- функция оптимизации;
- метрика качества обучения нейронной сети;
- изменение скорости обучения нейронной сети;
- изменение момента.

4.6 Результаты обучения сверточной нейронной сети

В результате сбора данных были собраны две крупные коллекции данных, различающихся тематикой изображений.

Первая коллекция основана на разделе «Политика» и имеет следующие особенности:

- содержит большое количество изображений с политическими деятелями;
- содержит большое количество изображений с политически значимыми объектами;
- общее количество объектов на изображении небольшое.

Вторая коллекция основана на разделе «Общество» и имеет следующие особенности:

- содержит большое количество изображения разноразнообразного характера;
- на изображении большое количество объектов.

Так как две собранные коллекции имеют достаточно серьезные отличия, то для каждой из них будут создаваться отдельные модели.

Рассмотрим обучение нейронной сети на данных из раздела «Политика». Данная выборка подверглась предварительной обработке: были удалены все изображения, которые имели крупный план политических деятелей.

В самом начале обучения имеется коллекция размером 60 тысяч уникальных записей, к каждой записи прилагается одно изображение.

Первые результаты показали, что обучение нейронной сети при выбранных параметрах невозможно. Результаты показывали, что подобранные слова мало соответствуют действительности, отображенной на изображениях. Поэтому, выполнив некоторые преобразования, был получен следующий вариант архитектуры.

Таблица 3 - Новый вариант архитектуры нейронной сети при обучении на данных из раздела «Политика»

Слой	Количество нейронов	Ядро	Активационная функция
Сверточный №1	32	2x2	ReLU
Субдискретизирующий №1	–	2x2	–
Сверточный №2	32	2x2	ReLU
Субдискретизирующий №2	–	2x2	–
Сверточный №3	64	2x2	ReLU
Субдискретизирующий №3	–	2x2	–
Сверточный №4	64	2x2	ReLU
Субдискретизирующий №4	–	2x2	–
Выравнивающий	–	–	–
Полносвязный №1	1024	–	ReLU
Полносвязный №2	1024	–	ReLU
Полносвязный №2	512	–	ReLU
Полносвязный №3	512	–	Softmax

При такой архитектуре нейронная сеть начала обучаться малыми шагами, была достигнута отметка по показателю точности в 2%, но результаты так и оставались неудовлетворительными.

После сделанных выводов было решено использовать другие функции потерь (таблица 4).

Таблица 4 - Функции потерь

Функция	Формула
Средняя квадратичная ошибка	$\frac{1}{n} \sum_{i=1}^n (y_i - x_i)^2$
Средняя абсолютная ошибка	$\frac{1}{n} \sum_{i=1}^n y_i - x_i $
Пуассон	$\frac{1}{n} \sum_{i=1}^n (y_i - x_i \cdot \log((y_i)))$
Косинусная близость	$\sum_{i=1}^n x_i \cdot y_i / \sqrt{\sum_{i=1}^n x_i^2} \cdot \sqrt{\sum_{i=1}^n y_i^2}$

Таблица 5 - Лучший вариант архитектуры нейронной сети при обучении на данных из раздела «Политика»

Слой	Количество нейронов	Ядро	Активационная функция
Сверточный №1	64	2x2	ReLU
Субдискретизирующий №1	—	2x2	—
Сверточный №2	64	2x2	ReLU
Субдискретизирующий №2	—	2x2	—
Сверточный №3	128	2x2	ReLU
Субдискретизирующий №3	—	2x2	—
Сверточный №4	128	2x2	ReLU
Субдискретизирующий №4	—	2x2	—
Выравнивающий	—	—	—
Полносвязный №1	4096	—	ReLU
Исключающий №1	—	—	—
Полносвязный №2	2048	—	ReLU
Исключающий №2	—	—	—
Полносвязный №3	1024	—	ReLU
Исключающий №3	—	—	—
Полносвязный №4	512	—	Softmax

Кроме применения новых функции потерь, было решено повысить скорость обучения нейронной сети и увеличить момент, что позволит выбраться из локальных минимумов при обучении.

По результатам многочисленных испытаний была получена следующая оптимальная архитектура, при которой получились достаточно интересные результаты.

Были добавлены так называемые «Исключающие слои» – их задача состоит в случайной установке доли единиц ввода в 0 при каждом обновлении во время обучения. Это помогает предотвратить переобучение нейронной сети.

Таблица 6 - Прочие гиперпараметры

Размер изображения	50x100
Размер вектор теста	512
Функция потерь	Косинусная близость
Функция оптимизации	Стохастический градиентный спуск
Скорость обучения	0,01
Момент	0,01
Размер подвыборки	32
Перетасовка обучающего множества	Использовать

Применение функции потерь на основе косинусной близости показало значительный прогресс при обучении. Полученные результаты оказались достаточно хорошими, но при этом оказалось, что нельзя опираться на какую-либо предлагаемую метрику, потому что они не могут правильно оценить смысловую близость двух предложений.

Все удовлетворительные результаты обучения нейронных сетей будут разобраны в следующей главе.

Рассмотрим обучение нейронной сети на данных из раздела «Общество». Размер коллекции составляет около 110 тысяч уникальных записей, к каждой записи прилагается одно изображение.

При обучении модели нейронной сети на данных из раздела «Общество» были сразу учтены доработки, полученные в результате обучения нейронной сети на данных из раздела «Политика».

После нескольких итерации различных вариантов обучения, были достигнуты удовлетворительные результаты, которые оказались даже более удачными, чем для данных по разделу «Политика». Далее рассмотрим оптимальную архитектуру, при которой удалось достичь этих результатов.

Таблица 7 - Лучший вариант архитектуры нейронной сети при обучении на данных из раздела «Общество»

Слой	Количество нейронов	Ядро	Активационная функция
Сверточный №1	128	3x3	ReLU
Субдискретизирующий №1	—	3x3	—
Исключающий №1	—	—	—
Сверточный №2	128	3x3	ReLU
Субдискретизирующий №2	—	3x3	—
Исключающий №2	—	—	—
Сверточный №3	256	3x3	ReLU
Субдискретизирующий №3	—	3x3	—
Исключающий №3	—	—	—
Сверточный №4	256	3x3	ReLU
Субдискретизирующий №4	—	3x3	—
Выравнивающий	—	—	—
Полносвязный №1	4096	—	ReLU
Исключающий №4	—	—	—
Полносвязный №2	4096	—	ReLU
Исключающий №5	—	—	—
Полносвязный №3	2048	—	ReLU
Исключающий №6	—	—	—
Полносвязный №4	2048	—	Softmax

Количество «Исключающих слоев» было увеличено, для того чтобы не допустить переобучения нейронной сети.

Таблица 8 - Прочие гиперпараметры

Размер изображения	100x200
Размер вектор теста	2048
Функция потерь	Косинусная близость
Функция оптимизации	Стохастический градиентный
Скорость обучения	0,2
Момент	0,2
Размер подвыборки	64
Перетасовка обучающего множества	Использовать

В виду того, что при обучении нейронной сети на коллекции данных из раздела «Политика» были получены удовлетворительные данные, то было принято решение увеличить размер входного и выходного вектора, для получение более качественного результата.



Полученные результаты оказались на порядок выше, чем при прошлых подходах. Это было ожидаемо, так как количество данных для обучения возросло почти в 2 раза, а также было увеличено количество полезной информации, которая оставалась после сжатия изображения, и увеличен размер вектора, отвечающего за представление текста, следовательно, нейронная сеть смогла сформировать более четкие соотношения между рассматриваемыми изображениями и текстом.

5 РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНОЙ АПРОБАЦИИ

После определения оптимальных гиперпараметров были получены две модели нейронных сетей, одна из них была обучена на данных, собранных из раздела «Политика» сайта РИА Новости, вторая – на данных, собранных из раздела «Общество» сайта РИА Новости.

Рассмотрим полученные результаты на основе случайных изображений, которые были предъявлены нейронной сети и не участвовали в обучении.

Таблица 9 - Результаты по разделу «Политика»

Изображение	Исходный текст	Полученный текст
1	2	3
	Главы МИД России и Финляндии обменялись поздравительными посланиями	'право'
	Депутат: контракт по электричеству в Крым как часть Украины невозможен	'пост', 'проект', 'эксперт'
	Путин с 1 января сократил штаты ряда министерств и ведомств	'президент'

1	2	3
	Челябинский губернатор уволил зама, нецензурно отозвавшегося о регионе	'нарушение'
	Комитет СФ: в докладе ФБК нет фактов противоправной деятельности Чайки	'глава', 'закон', 'совет'
	Путин подписал закон о приостановлении договора о ЗСТ с Украиной	'важность'
	Путин отметил конструктивное развитие отношений России и Франции	'доступ', 'президент', 'провести',
	ЦИК пообещал партиям техническую помощь в проведении праймериз	'голосование', 'дело'

1	2	3
	Расходы Думы на госзакупки нового созыва составят 2,7 миллиарда рублей	'госдума', 'комитет', 'проект',

При объективной оценке можно утверждать, что нейронной сети удалось научиться сопоставлять изображения с определёнными словами, которые в некоторой степени отражают смысл изображения. Кроме того, можно заметить, что нейронная сеть научилась распознавать некоторые объекты, например, на достаточно большом количестве изображений в коллекции присутствовал президент России и нейронная сеть смогла запомнить это.

Рассмотрим результаты полученные со второй модели, для которой обучающие данные были собраны с раздела «Общество».

Таблица 10 - Результаты по разделу «Общество»

Изображение	Исходный текст	Полученный текст
1	2	3
	Васильева готова обсудить с Путиным изучение родных языков в регионах	'кино', 'путешествие', 'флешмоб'

1	2	3
	Путин поддержал создание списка коренных народов по месту жительства	'гендиректор', 'интернет'
	Социолог: более 90% крымчан скажут "нет" энергоконтракту с Украиной	'опрос'
	Минюст назвал условия отказа в регистрации автомобиля для должников	'минюст'
	Сенатор о стрельбе в Дербенте: ИГ будет приписывать подобные ЧП себе	'завершить', 'миллиард', 'новое', 'обязать'
	В Севастополе обещают по максимуму подавать электроэнергию в праздники	'погода'

1	2	3
	Правительство одобрило проект о биологической безопасности страны	'законопроект', 'музей', 'открыть', 'правительство'
	В Москве в субботу ожидаются дождь с мокрым снегом и гололедица	'погода'

В сравнении с результатами по разделу «Политика», полученные результаты выглядят более сопоставимыми с изображениями. Это является следствием большего числа входных данных и их разнообразностью.

Полученные тексты можно считать не совсем осмысленными, так как они не стоят в нужном падеже или склонении, но они отражают действительность, отображенную на фотографии. Если интерпретировать полученный набор слов, как некоторый текст, то можно утверждать, что было достигнуто решение задачи сопоставление изображения и текста. С точки зрения человека, полученные результаты выглядят как не совсем связный текст, но как для результатов, которые были получены с помощью машины, они выглядят удовлетворительно.

Исходя из полученных результатов, можно сделать следующие выводы и подвести итоги:

- с точки зрения объективной оценки человеком, полученные результаты соответствуют тому, что изображено на фотографии, но выяснилось, что это никак нельзя связать с метрикой «точность», которая

используется при оценке результатов работы нейронной сети внутри библиотеки TensorFlow;

- в ходе обучения нейронных сетей стало понятно, что применяемые метрики не способны оценить смысловое сходство двух текстов, поэтому в будущем для развития данной задачи необходимо разрабатывать собственные метрики, сигнализирующие о качестве обучения;
- визуальная оценка результатов дала понять, что модель, обучающаяся на большем количестве данных, смогла давать более точные, с точки зрения человека, результаты.

					ТПЖА 090301.02.087 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана программа генерации подписи к цифровому изображению. Реализуемая программа должна решить задачу сопоставления изображения и текста.

Решение проблемы строилось на основе одного из методов машинного обучения – искусственных нейронных сетях.

В рамках проекта была собрана коллекция данных общим объемом около 150 тысяч уникальных записей. Коллекция представляет собой данные, которые в дальнейшем необходимы для обучения нейронной сети.

Была решена задача по представлению текста в виде числового вектора. Это представляло собой эталонный ответ нейронной сети на каждое входное изображение. Векторизатор текста был реализован на основе статистической меры TF-IDF.

Были спроектированы и обучены две модели сверточной нейронной сети на основе двух частей коллекции данных: первая – на основе данных из раздела «Политика», вторая – на основе данных из раздела «Общество». Обе модели способны выдавать достаточно качественные ответы по случайным входным изображениям, не участвовавшим в обучении.

В ходе обучения нейронных сетей была решена проблема попадания в локальный минимум, в такой ситуации нейронная сеть не обучалась и давала неудовлетворительные ответы. Решение заключалось в ведении дополнительных слоев и гиперпараметров, позволяющие исключать локальные минимумы и находить глобальные.

В процессе реализации была достигнута поставленная цель – это создание программы, способной генерировать подписи на основе цифрового изображения.

ПРИЛОЖЕНИЕ А

(обязательное)

Руководство по использованию

Руководство по использованию инструмента сбора данных

Для того, чтобы воспользоваться разработанной программой для сбора данных с сайта «РИА Новости» необходимо установить интерпретатор python

3.7.x. Необходимо открыть командную строку и ввести следующую команду:

C:\> python <Название программы>.

<Название программы> – это текущее название файла программы.

Для настройки необходимо открыть исходный код в блокноте и в первых строках исходного кода сделать правки: изменить год – переменная «year» и/или изменить название раздела – переменная «section». Название секции должно строго соответствовать названию с сайта

Для того, чтобы воспользоваться разработанной программой для фильтрации изображений с крупным планом необходимо установить интерпретатор python 3.7.x, и xml-файл с конфигурацией классификатора. Необходимо открыть командную строку и ввести следующую команду:

C:\> python <Название программы>.

<Название программы> – это текущее название файла программы.

Для настройки необходимо открыть исходный код в блокноте и в первых строках исходного кода сделать правки: изменить год – переменная «year» и/или изменить название CSV-файла – переменная «filename».

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг программы для сбора данных

create_dataset.py

```
import requests
from bs4 import BeautifulSoup
from CSVmodule import *

# Год
year = '2015'

# Раздел: society, politics
section = 'society'

# Адрес странички РИА Новости -
# С заданными временными промежутками от 1 января X года (20190101) до 31 декабря X года (20191231)
# С заданным разделом (politics)
# Полгода - 0630
url = 'https://ria.ru/services/tagsearch/?date_start=' + year + '0101&date=' + year + '1231&tags%5B%5D=' + section

# Список с результатами сбора данных
result = []

# Итератор для названий картинок в файловой системе
i = 0

# Делаем запрос на новый блок новостей, пока не дойдем до начала текущего года (начала следующего года)
while True:

    # Запрос к серверу по url
    request = requests.get(url)

    # Передаем полученный html в bs4-функцию
    soup = BeautifulSoup(request.text, features="lxml")

    # Необходимо найти все div, которые хранят ссылку на новость
    div_list = soup.find_all('div', {'class': 'list-item'})

    # Необходимо найти следующий url: если он ссылается на следующий год, то выйти из цикла
```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

72

```

next_url = soup.find('div', {'class': 'list-items-loaded'})['data-next-url']
if next_url.partition('date=')[-1][0:4] != year:
    break

# Перебираем все элементы div, в которых есть новости с картинками
for list_item in div_list:

    # Найти тег ресурса-фотографии
    source = list_item.find('source')

    # Если есть фотография - найти остальную информацию сохранить, иначе - пропустить
    if source is not None:
        # Строчка с данными для одной фотографии
        row = []
        # Источник новости
        row.append('РИА Новости')
        # Ссылка на статью
        link_to_news = list_item.find('a', {'class': 'list-item__image'})['href']
        row.append(link_to_news)
        # Рубрика
        row.append('Общество')
        # Название статьи
        title = list_item.find('a', {'class': 'list-item__title'}).contents[0]
        row.append(title)
        # Фотография
        pict = requests.get(source['srcset'])
        file_name = '.' + section + '_' + year + 'image_' + str(i) + '.jpg'
        with open(file_name, "wb") as file:
            file.write(pict.content)
        row.append(file_name)
        # Добавляем новую строчку в список
        result.append(row)

    i += 1

# Вытаскиваем ссылку на следующую страницу
url = 'https://ria.ru' + next_url

# Сколько загружено фотографий
print('Загружено - ', str(i))

# !) Необходимо дособирать статьи с последней страницы
# Перебираем все оставшиеся элементы div, при этом проверяем у каждой статьи год
for list_item in div_list:

```

```

# Найти тег ресурса-фотографии
source = list_item.find('source')

data = list_item.find('div', {'class': 'list-item__date'}).text
if data.split(',')[0].split(' ')[-1] == str(int(year)-1):
    break

# Если есть фотография - найти остальную информацию сохранить, иначе - пропустить
if source is not None:
    # Строчка с данными для одной фотографии
    row = []
    # Источник новости
    row.append('РИА Новости')
    # Ссылка на статью
    link_to_news = list_item.find('a', {'class': 'list-item__image'})['href']
    row.append(link_to_news)
    # Рублика
    row.append('Политика')
    # Название статьи
    title = list_item.find('a', {'class': 'list-item__title'}).contents[0]
    row.append(title)
    # Фотография
    pict = requests.get(source['srcset'])
    file_name = '\\ ' + section + ' _ ' + year + '\\image_ ' + str(i) + '.jpg'
    with open(file_name, "wb") as file:
        file.write(pict.content)
    row.append(file_name)
    # Добавляем новую строчку в список
    result.append(row)

    i += 1

# Сколько загружено фотографий
print('Загружено - ', str(i))

# Записываем результаты в файл с указанием атрибута записи
csv_writer(result, 'data_ ' + section + ' _ ' + year + '.csv', 'a')

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

74

face_detection.py

```
import cv2
import os.path
from CSVmodule import *

# Целевой год
year = 2019

# Название csv-файла со старшим годом
filename = './data_politics_%d.csv' % year

# Конфигурация для классификатора из openCV
face_cascade = cv2.CascadeClassifier('./haarcascade_face.xml')

# Список, содержащий новости с изображения без "крупного плана"
no_close_up = []

# Индекс изображения
count = 0

# Пока есть csv-файлы, ищем
while os.path.exists(filename):
    print('Данные по \'' + filename + '\' загружены.\n')

    # Чтение csv-файла
    data = csv_reader(filename)

    # Для каждой строчки в файле
    for row in data:
        # Для отбрасывания "битых изображений"
        try:
            # Загружаем картинку
            image = cv2.imread(row[-1])

            # Накладываем фильтр "оттенки серого"
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        except Exception:
            print('Warning - ', row)
            continue

    # Функция нахождения лица на изображения
    # grey - это входное изображение в оттенках серого.
    # scaleFactor - параметр, определяющий размер изображения при каждой шкале изображения.
    # minNeighbors - параметр, указывающий, сколько соседей должно иметь каждый прямоугольник кандидата,
```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

75

```

# чтобы сохранить его. Более высокое число дает более низкие ложные срабатывания.
# minSize - минимальный размер прямоугольника, который считается лицом.
faces = face_cascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=6,
    minSize=(40, 40)
)

# Если необходимо вывести лицо на экран - рисуется синий прямоугольник
# for [x, y, w, h] in faces:
#     cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)
#     roi_gray = gray[y:y + h, x:x + w]
#     roi_color = image[y:y + h, x:x + w]

# Если нет "крупного плана", то добавляем в список
if isinstance(faces, tuple):
    # Сохраняем изображение
    cv2.imwrite('./no_close_up_politics_all_year/image_%d.jpg' % count, image)

# Изменяем адрес изображения
row[-1] = './no_close_up_politics_all_year/image_%d.jpg' % count

# Добавляем строчку в новый список
no_close_up.append(row)
count += 1

# Отображение на экране изображения с распознанным лицом, подходящим под крупный план
# cv2.imshow('image', image)

# Запись в файл часть данных за один год
csv_writer(no_close_up, 'data_no_close_up_politics_all_year.csv', 'a')

print('Данные по \'' + filename + '\'' обработаны.\n')

# Переход к следующему году
year -= 1
filename = './data_politics_%d.csv' % year

# cv2.destroyAllWindows()

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг программы модели нейронной сети

CNN.py

```
# -*- coding: utf-8 -*-
"""CNN_politics.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1yYBvmiqkSTe6pBi4IMpQJno80G8YIHut
"""

from google.colab import drive
drive.mount('/content/drive')

!pip install pymorphy2

# Commented out IPython magic to ensure Python compatibility.
from typing import List, Any
from google.colab import files
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import AlphaDropout, Dropout
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras import losses
from keras import metrics
from keras import optimizers
import csv
import os
import numpy as np
import matplotlib.pyplot as plt
import PIL
from PIL import Image
import time
```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

77

```

# %matplotlib inline

from vectorize_TF_IDF import *
from CSVmodule import *
from read_write import *

nltk.download('stopwords')
nltk.download('punkt')

""""Создаем исходные данные для "Политики""""

# Загружаем CSV таблицу
data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_no_close_up_politics_all_year.csv')

# Ответы нейронной сети - предложения
Y_train_data, Y_test_data = [x[3] for x in data[0:round(len(data)*0.8)]], [x[3] for x in data[round(len(data)*0.8):len(data)]]
# Входные данные - изображения
X_train_data, X_test_data = [x[-1] for x in data[0:round(len(data)*0.8)]], [x[-1] for x in
data[round(len(data)*0.8):len(data)]]

""""Создаем исходные данные для "Общества""""

# Создаем единую CSV таблицу
data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_2015.csv')
for i in range(4):
    data += csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_'+str(2016+i)+'.csv')
    csv_writer(data,'data_all_society.csv','w')

# Ответы нейронной сети - предложения
Y_train_data, Y_test_data = [x[3] for x in data[0:round(len(data)*0.95)]], [x[3] for x in
data[round(len(data)*0.05):len(data)]]
# Входные данные - изображения
X_train_data, X_test_data = [x[-1] for x in data[0:round(len(data)*0.95)]], [x[-1] for x in
data[round(len(data)*0.05):len(data)]]

""""Подготовка тренировочных изображений для обучения нейронной сети""""

# Политика
start_time = time.clock()
all_part_X_train = []
for x in X_train_data
    all_part_X_train.append(np.asarray(Image.open('/content/drive/My Drive/Архив - РИА
Новости/'+x[2:]).resize((100,50)), dtype='float16'))

```

```

X_train = np.asarray(all_part_X_train)

print('Вычислил значение X_train и записываю его на диск в X_train.pickle')

with open('X_train.pickle', 'wb') as f:
    pickle.dump(X_train, f)
print("{:g} s".format(time.clock() - start_time))

# Тест

fail_string = []
for x in X_train_data:
    '/content/drive/My Drive/Архив - РИА Новости/'+x.replace("\\','")[2:]
    if not os.path.exists(image_path):
        fail_string.append(image_path)

write_mas('fail_string.txt',fail_string)
files.download("fail_string.txt")

load_img('/content/drive/My Drive/Архив - РИА Новости/society_2019/image_0.jpg', target_size=(100,200))

# Общество
start_time = time.clock()
for i in range(4,5):
    start_time = time.clock()

csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_'+str(2015+i)+'.csv')
list_data = [x[-1] for x in csv_data]

part = []
for x in list_data[:15000]:
    image_path = '/content/drive/My Drive/Архив - РИА Новости/'+x.replace("\\','")[2:]
    try:
        part.append(img_to_array(load_img(image_path, target_size=(100,200)), dtype='float16'))
    except PIL.UnidentifiedImageError:
        print(image_path)
        part.append(img_to_array(load_img('/content/drive/My Drive/Архив - РИА
Новости/society_'+str(2015+i)+'image_0.jpg', target_size=(100,200)), dtype='float16'))

if len(part)%1000==0:
    print("Загружено - " + str(len(part)))
# except Exception as err:
#     print(err + ' - ' + image_path)

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

79


```

        part.append(img_to_array(load_img('/content/drive/My Drive/Архив - РИА
Новости/society_'+str(2015+i)+'image_0.jpg', target_size=(100,200)), dtype='float16'))

X_train_society = np.array(part)

print('X_train_society годом '+str(2015+i)+' загружен и начинается запись на диск в X_train_society.pickle')

with open('X_train_society_'+str(2015+i)+'.pickle', 'wb') as f:
    pickle.dump(X_train_society, f)

print ("{:g} s".format(time.clock() - start_time))

"""Подготовка тестовых изображений для обучения нейронной сети"""

# Политика
start_time = time.clock()
all_part_X_test = []
for x in X_test_data:
    all_part_X_test.append(np.asarray(Image.open('/content/drive/My Drive/Архив - РИА Новости/'+x[2:]).resize((100,50)),
dtype='float16'))
X_test = np.asarray(all_part_X_test)

print('Вычислил значение X_test и записываю его на диск в X_test.pickle')

with open('X_test.pickle', 'wb') as f:
    pickle.dump(X_test, f)
print ("{:g} s".format(time.clock() - start_time))

# Общество
start_time = time.clock()

csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_2019.csv')
list_data = [x[-1] for x in csv_data]

part = []
for x in list_data[15000:]:
    image_path = '/content/drive/My Drive/Архив - РИА Новости/'+x.replace("\\','")[2:]
    try:
        part.append(img_to_array(load_img(image_path, target_size=(100,200)), dtype='float16'))
    except PIL.UnidentifiedImageError:
        print('UnidentifiedImageError - ' + image_path)
        part.append(img_to_array(load_img('/content/drive/My Drive/Архив - РИА Новости/society_2019/image_0.jpg',
target_size=(100,200)), dtype='float16'))
    except Exception as err:

```

```

        print(err + ' - ' + image_path)
        raise

X_test_society = np.array(part)

print('X_test_society загружен и начинается запись на диск в X_test_society.pickle')

with open('X_test_society.pickle', 'wb') as f:
    pickle.dump(X_test_society, f)

print("{:g} s".format(time.clock() - start_time))

"""Создание или загрузка модели для векторизации текста"""

text_model = TextToVectorTfidf()
# text_model.load_data(filename='/content/data_all_society.csv')
# text_model.train_model(max_features=2048, filename='society_2048_tfidf.pickle')

text_model.load_model('/content/politics_tfidf.pickle') # politics_tfidf society_tfidf

text_model.load_model('/content/society_2048_tfidf.pickle') # politics_tfidf society_tfidf

# Тестовый запуск модели для векторизации
csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_2015.csv')
list_data = csv_data[3000:3002]
vector = text_model.get_vector(list_data[0][3])
print(list_data[0][3], vector)
print('-----')
vector2 = text_model.get_vector(list_data[1][3])
print(list_data[1][3], vector2)
print('-----')
print(cosine_similarity(vector.toarray(), vector2.toarray()))

"""Подготовка ответов на тренировочный набор данных"""

# Политика
start_time = time.clock()
all_part_Y_train = []
for y in Y_train_data:
    all_part_Y_train.append(text_model.get_vector(y).toarray()[0])

print('Вычислил значение Y_train и записываю его на диск в Y_train.pickle')

Y_train = np.asarray(all_part_Y_train)

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

81

```

with open('Y_train.pickle', 'wb') as f:
    pickle.dump(Y_train, f)
print("{:g} s".format(time.clock() - start_time))

# Общество
start_time = time.clock()

part = []
for i in range(0,4):
    csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_'+str(2015+i)+'.csv')
    list_data = [x[3] for x in csv_data]

    for y in list_data:
        part.append(text_model.get_vector(y).toarray()[0])

    print('Данные за '+str(2015+i)+' загружены')

csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_2019.csv')
list_data = [x[3] for x in csv_data]

for y in list_data[:15000]:
    part.append(text_model.get_vector(y).toarray()[0])

Y_train_society = np.asarray(part)

print('Вычислили значение Y_train_society и записываю его на диск в Y_train_2048_society.pickle')

with open('Y_train_2048_society.pickle', 'wb') as f:
    pickle.dump(Y_train_society, f)

print("{:g} s".format(time.clock() - start_time))

""Подготовка ответов на тестовый набор данных""

start_time = time.clock()
all_part_Y_test = []
for y in Y_test_data:
    all_part_Y_test.append(text_model.get_vector(y).toarray()[0])

print('Вычислили значение Y_test и записываю его на диск в Y_test.pickle')

Y_test = np.asarray(all_part_Y_test)
with open('Y_test.pickle', 'wb') as f:
    pickle.dump(Y_test, f)

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

82

```

print("{:g} s".format(time.clock() - start_time))

# Общество
start_time = time.clock()

part = []
csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_2019.csv')
list_data = [x[3] for x in csv_data]

for y in list_data[15000:]:
    part.append(text_model.get_vector(y).toarray()[0])

Y_test_society = np.asarray(part)

print('Вычислил значение Y_test_society и записываю его на диск в Y_test_2048_society.pickle')

with open('Y_test_2048_society.pickle', 'wb') as f:
    pickle.dump(Y_test_society, f)

print("{:g} s".format(time.clock() - start_time))

""""Если имеются, то загружаем уже готовые засериализованные наборы""""

# Загрузка данных о Политике
with open('/content/drive/My Drive/X_train.pickle', 'rb') as f:
    X_train = pickle.load(f)

print('Загружено X_train')

with open('/content/drive/My Drive/Y_train.pickle', 'rb') as f:
    Y_train = pickle.load(f)

print('Загружено Y_train')

with open('/content/drive/My Drive/X_test.pickle', 'rb') as f:
    X_test = pickle.load(f)

print('Загружено X_test')

with open('/content/drive/My Drive/Y_test.pickle', 'rb') as f:
    Y_test = pickle.load(f)

print('Загружено Y_test')

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

83

```

# Тестик размерности
for i in range(0,5):
    with open('/content/drive/My Drive/X_train_society_'+str(2015+i)+'.pickle', 'rb') as f:
        load_X_train = pickle.load(f)

    csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_'+str(2015+i)+'.csv')
    print(load_X_train.shape)
    print(len(csv_data))
    print('-----')

# Загрузка данных об Обществе
# Соединение всех кусочков
with open('/content/drive/My Drive/X_train_society_2015.pickle', 'rb') as f:
    X_train = pickle.load(f)

for i in range(1,5):
    with open('/content/drive/My Drive/X_train_society_'+str(2015+i)+'.pickle', 'rb') as f:
        load_X_train = pickle.load(f)

    X_train = np.vstack((X_train, load_X_train))

print('Загружено X_train')

with open('/content/drive/My Drive/Y_train_2048_society.pickle', 'rb') as f:
    Y_train = pickle.load(f)

print('Загружено Y_train')

# with open('/content/drive/My Drive/X_test_society.pickle', 'rb') as f:
#     X_test = pickle.load(f)

# print('Загружено X_test')

# with open('/content/drive/My Drive/Y_test_2048_society.pickle', 'rb') as f:
#     Y_test = pickle.load(f)

# print('Загружено Y_test')

# Масштабирование матриц
X_train /= 255
X_test /= 255

from keras.models import model_from_json

```

```

from sklearn.feature_extraction.text import TfidfVectorizer
import pickle

# Дообучение готовой модели. Загрузка
print("Начинаю загрузку model.json")

# Загружаем данные об архитектуре сети
json_file = open("/content/model.json", "r")
model_json = json_file.read()
json_file.close()

print("Загрузка model.json\nНачинаю инициализацию модели и весов")

# Создаем модель
model = model_from_json(model_json)
# Загружаем сохраненные веса в модель
model.load_weights("/content/model.h5")
print("Загрузка сети завершена")

# Лучшая активационная функция - 'relu'
# Создаем последовательную модель
model = Sequential()
# Первый сверточный слой
model.add(Conv2D(128, (3, 3), input_shape=(100, 200, 3), activation='relu'))
# Первый слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Слой преобразования данных из 2D представления в плоское
model.add(Flatten())

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

85

```

# Полносвязный слой для классификации
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2048, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2048, activation='softmax'))

print(model.summary())

model.get_config()

# Лучший loss='cosine_proximity', optimizer=optimizers.SGD(learning_rate=0.05, momentum=0.01, nesterov=True),
metrics=['cosine_proximity', 'accuracy']
# mean_squared_error cosine_proximity
model.compile(loss='cosine_proximity',
              optimizer=optimizers.SGD(learning_rate=0.2, momentum=0.2, nesterov=True),
              metrics=['cosine_proximity', 'accuracy'])

"""Обучение с генератором"""

train_datagen = ImageDataGenerator(
    horizontal_flip=True,
    # rescale=1. / 255,
    # rotation_range=0.2,
    width_shift_range=0.05,
    height_shift_range=0.05,
    # shear_range=0.2,
    zoom_range=0.1)

history = model.fit_generator(
    train_datagen.flow(X_train,
                      Y_train,
                      batch_size=32),
    # target_size=(img_width, img_height)),
    steps_per_epoch=1000, #num_samples / batch_size
    epochs=200,
    validation_data=(X_test, Y_test),
    shuffle=True,
    # initial_epoch=100,
    verbose=2)

```

```

"""Обучение без генератора"""

history = model.fit(X_train, Y_train,
                    batch_size=64,
                    epochs=100,
                    validation_split=0.1,
                    shuffle=True,
                    # initial_epoch = 100,
                    verbose=2)

# Оцениваем качество обучения модели на тестовых данных
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Точность работы на тестовых данных: %.2f%%" % (scores[1]*100))

"""Сохранение"""

val_accuracy = history.history['val_accuracy']
xc = history.epoch

plt.figure(figsize=(12, 7))
plt.grid()
plt.plot(xc, val_accuracy, label='val_accuracy')
plt.legend()
plt.savefig('val_accuracy.jpg')
files.download('val_accuracy.jpg')

val_cosine_proximity = history.history['val_cosine_proximity']
xc = history.epoch

plt.figure(figsize=(12, 7))
plt.grid()
plt.plot(xc, val_cosine_proximity, label='val_cosine_proximity')
plt.legend()
plt.savefig('val_cosine_proximity.jpg')
files.download('val_cosine_proximity.jpg')

val_accuracy = history.history['val_accuracy']
val_cosine_proximity = history.history['val_cosine_proximity']
xc = history.epoch

plt.figure(figsize=(12, 7))
plt.grid()
plt.plot(xc, val_accuracy, label='val_accuracy')
plt.plot(xc, val_cosine_proximity, label='val_cosine_proximity')

```



```

plt.legend()
plt.savefig('val_accuracy+val_accuracy.jpg')
files.download('val_accuracy+val_accuracy.jpg')

accuracy = history.history['accuracy']
xc = history.epoch

plt.figure(figsize=(12, 7))
plt.grid()
plt.plot(xc, accuracy, label='accuracy')
plt.legend()
plt.savefig('accuracy.jpg')
files.download('accuracy.jpg')

cosine_proximity = history.history['cosine_proximity']
xc = history.epoch

plt.figure(figsize=(12, 7))
plt.grid()
plt.plot(xc, cosine_proximity, label='cosine_proximity')
plt.legend()
plt.savefig('cosine_proximity.jpg')
files.download('cosine_proximity.jpg')

model_json = model.to_json()
json_file = open("model.json", "w")
json_file.write(model_json)
json_file.close()
model.save_weights("model.h5")

# files.download("model.json")
# files.download("model.h5")

"""# ***Апробация!***"""

from keras.models import model_from_json

from sklearn.feature_extraction.text import TfidfVectorizer
import pickle

#
loaded_model = model

print("Загружаю сеть из файлов")

```

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

88

```

# Загружаем данные об архитектуре сети
json_file = open("/content/model (2).json", "r")
loaded_model_json = json_file.read()
json_file.close()

# Создаем модель
loaded_model = model_from_json(loaded_model_json)

# Загружаем сохраненные веса в модель
loaded_model.load_weights("//content/model (2).h5")
print("Загрузка сети завершена")

with open('/content/society_2048_tfidf.pickle', 'rb') as f: # society_2048_tfidf politics_tfidf
    tf_idf = pickle.load(f)
names = tf_idf.get_feature_names()

# Тест для Политики
csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_politics_2015.csv')
list_data = csv_data[:61]

result = []

for x in list_data:
    one_X_test = np.asarray([np.asarray(Image.open('/content/drive/My Drive/Архив - РИА Новости/'+x[-
1].replace('\\','/')[2:]).resize((100,50)), dtype='float16')])
    pred = loaded_model.predict(one_X_test)[0]
    sort_pred = np.sort(pred)
    # print(sort_pred)
    top = sort_pred[-5]

    top_word = []
    for j in range(512):
        if pred[j]>=top and pred[j] !=0:
            top_word.append(j)

    words = []
    for word in top_word:
        words.append(names[word])
    words.append(x[3])

    result.append(words)
    print(list_data.index(x), words)

i = 26
load_img('/content/drive/My Drive/Архив - РИА Новости/politics_2015/image_'+str(i)+'.jpg', target_size=(100,200))

```

```

# Тест для Общества
csv_data = csv_reader('/content/drive/My Drive/Архив - РИА Новости/data_society_2019.csv')
list_data = csv_data[50:100]

result = []

for x in list_data:
    part = []
    image_path = '/content/drive/My Drive/Архив - РИА Новости/'+x[-1].replace("\\\\, '/'")[2:]
    part.append(img_to_array(load_img(image_path, target_size=(100,200)), dtype='float16'))
    one_X_test = np.array(part)

    one_Y_test = text_model.get_vector(x[3]).toarray()[0]
    # print(one_Y_test)

    pred = loaded_model.predict(one_X_test)[0]
    sort_pred = np.sort(pred)
    top = sort_pred[-5]

    top_word = []
    for j in range(2018):
        if pred[j]>=top and pred[j] !=0:
            top_word.append(j)

    words = []
    for word in top_word:
        words.append(names[word])
    words.append(x[3])

    result.append(words)
    print(list_data.index(x), words)

i = 50 + 11
load_img('/content/drive/My Drive/Архив - РИА Новости/society_2019/image_'+str(i)+''.jpg')

result

write_mas('result.txt',result)
files.download("result.txt")

```

ПРИЛОЖЕНИЕ Г

(обязательное)

Авторская справка

Я, Пивоваров Иван Алексеевич, автор выпускной квалификационной работы «Разработка программы генерации подписи к цифровому изображению» сообщаю, что мне известно о персональной ответственности автора за разглашение сведений, подлежащих защите законами РФ о защите объектов интеллектуальной собственности.

Одновременно сообщаю, что:

1. При подготовке к защите выпускной квалификационной работы не использованы источники (документы, отчеты, диссертации, литература и т.п.), имеющие гриф секретности или «Для служебного пользования» ФГБОУ ВО «Вятский государственный университет» или другой организации.

2. Данная работа не связана с незавершенными исследованиями или уже с завершенными, но еще официально не разрешенными к опубликованию ФГБОУ ВО «Вятский государственный университет» или другими организациями.

3. Данная работа не содержит коммерческую информацию, способную нанести ущерб интеллектуальной собственности ФГБОУ ВО «Вятский государственный университет» или другой организации.

4. Данная работа не является результатом НИР или ОКР, выполняемой по договору с организацией.

5. В предлагаемом к опубликованию тексте нет данных по незащищенным объектам интеллектуальной собственности других авторов.

6. Использование моей дипломной работы в научных исследованиях оформляется в соответствии с законодательством РФ о защите интеллектуальной собственности отдельным договором.

Автор: Пивоваров И.А. «___» _____ 2020 г. _____
подпись

Сведения по авторской справке подтверждаю: «___» _____ 2020 г.

Заведующий кафедрой ЭВМ: Д.А. Страбыкин _____
подпись

					ТПЖА 090301.02.087 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		91

ПРИЛОЖЕНИЕ Д

(обязательное)

Библиографический список

- 1) Тявкин, И. В. Математическая модель информационного поиска и оценка эффективности поисковой системы [Электронный ресурс] / И. В. Тявкин, В. М. Тютюнник. – Вестник ТГТУ. 2008. Том 14. № 3. – Режим доступа: http://vestnik.tstu.ru/rus/t_14/pdf/14_3_004.pdf, свободный.
- 2) Habr.com: русскоязычный веб-сайт в формате коллективного блога с элементами новостного сайта [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/>, свободный. – Загл. с экрана.
- 3) Robocraft.ru: сообщество любителей робототехники, электроники и программирования [Электронный ресурс]. – Режим доступа: <http://robocraft.ru/page/opencv/>, свободный. – Загл. с экрана.
- 4) Райан, Митчелл Скрапинг веб-сайтов с помощью Python. Сбор данных из современного интернета / Митчелл Райан. – перевод Груздеев А. В. – под ред. Мовчан Д.А. – издательство ДМК-Пресс, 2016г.
- 5) Федорова А. А. Распознавание английского текста сверточной нейронной сетью // Молодой ученый. – 2016. – №14. – С. 97-102. – URL <https://moluch.ru/archive/118/32732/> (дата обращения: 03.04.2020).
- 6) Голубинский А. Н. Распознавание объектов на телевизионных изображениях с использованием аппарата сверточных нейронных сетей [Электронный ресурс] / Голубинский А. Н., Толстых А. А. // Журнал Вестник Воронежского института МВД России – Режим доступа: <https://cyberleninka.ru/article/n/raspoznavanie-obektov-na-televizionnyh-izobrazheniyah-s-ispolzovaniem-apparata-svertochnyh-neyronnyh-setey/viewer>, свободный.

Изм.	Лист	№ докум.	Подпись	Дата

ТПЖА 090301.02.087 ПЗ

Лист

92

- 7) Лагунов Н.А. Распознавание объектов на телевизионных изображениях с использованием аппарата сверточных нейронных сетей [Электронный ресурс] / Лагунов Н.А. // Журнал Пространство и Время – Режим доступа: <https://cyberleninka.ru/article/n/primeneniye-svertochnyh-neyronnyh-setey-v-zadachah-raspoznavaniya-mnogoparametricheskih-obektov/viewer>, свободный.
- 8) Jefkine Kafunah Backpropagation In Convolutional Neural Networks [Электронный ресурс] / Jefkine Kafunah // Jefkine, 5 September 2016 – Режим доступа: <https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/>, свободный.
- 9) О.А. Черненко Разработка автоматизированной системы семантического анализа текстовой информации [Электронный ресурс] / О.А. Черненко , О.А. Гордеева // Самарский национальный исследовательский университет имени академика С.П. Королева, Самара, Россия – Режим доступа: http://repo.ssau.ru/bitstream/Informacionnye-tehnologii-i-nanotehnologii/Razrabotka-avtomatizirovannoi-sistemy-semanticheskogo-analiza-tekstovoi-informacii-64154/1/paper%20324_1800-1804.pdf, свободный.
- 10) Google Colab Free GPU Tutorial [Электронный ресурс] / fuat // Deep Learning Turkey – Режим доступа: <https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d>, свободный.