РЕФЕРАТ

Выпускная квалификационная работа содержит 94 страниц, 37 рисунков, 7 таблиц, 28 использованных источника.

СЕНТИМЕНТ-АНАЛИЗ, БИНАРНАЯ КЛАССИФИКАЦИЯ, ГЛУБОКОЕ ОБУЧЕНИЕ, ВЕКТОРИЗАЦИЯ СЛОВ, АНСАМБЛИРОВАНИЕ, ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, КЛАСТЕРИЗАЦИЯ

Выпускная квалификационная работа посвящена исследованию сентимент-анализа и тематического моделирования текстов с помощью методов нейросетевого моделирования, кластеризации и вероятностных моделей. В ходе исследования были разработаны ансамблевая модель бинарной классификации тональности текстов, метод латентного размещение Дирихле (LDA), исследован метод, лежащий в основе модели тематического моделирования BertTopic, а также разработан веб-сервис визуализации имплементации этих инструментов.

Теоретическая часть работы содержит обзор современных методов решения задачи сентимент-анализа текстов, векторизации слов и тематического моделирования с математической формализацием и анализом использования выбранных моделей.

В практической части содержится анализ построенных моделей в зависимости от вариации гиперпараметров, ансамблирование моделей для сентимент-анализа и аргументация выбора конечного инструмента решения проблемы.

Результат разработки нейросетевого комплекса для решения задачи сентимент-анализа был представлен на XVI международной отраслевой научно-технической конференции «Технологии информационного будущего». Аргументация применения ансамблирования к смежной задаче бинарной классификации сложных структур была представлена на XLVIII Международной молодёжной научной конференции «Гагаринские чтения» [1]

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **4**

ОСНОВНАЯ ЧАСТЬ **6**

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7

1.1 Постановка задачи 7

1.2 Предварительная обработка текстов 9

1.3 Нейронные сети 19

1.4 Сверточные нейронные сети (CNN) 22

1.5 Рекуррентные нейронные сети (RNN) 28

1.6 Тематическое моделирование 36

2 ПРАКТИЧЕСКАЯ ЧАСТЬ 41

2.1 Используемые инструменты 41

2.2 Экспериментальные результаты 41

ЗАКЛЮЧЕНИЕ **56**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **57**

ПРИЛОЖЕНИЯ **60**

ПРИЛОЖЕНИЕ 1 61

ПРИЛОЖЕНИЕ 2 79

ВВЕДЕНИЕ

Актуальность автоматизации семантического анализа не способна подвергаться сомнению. Очевидным следствием является плотная концентрация научных работ, статей и книг, направленных на решение поставленной проблемы. Для этого предлагались различные варианты: от построения моделей, основанных на правилах, требующих много потраченного времени и постоянной модерации в силу нестабильности постоянно мутирующего языка, до методов машинного обучения и нейронных сетей, последним из которых и посвящена данная работа. С целью проведения экскурса в выбранной сфере рассмотрим некоторые из существующих работ выбранной тематики.

В статье [8] сконцентрированно разбираются подходы машинного обучения: наивный байесовский классификатор (Naive Bayes), случайные леса (random forest) и метод опорных векторов (SVM). Лучшая модель (наивный байесовский классификатор) показала результат метрики качества F1-мера: 0.94. В работе [28] за исключением предыдущих рассмотрены методы Rochio и k-ближайших соседей (kNN, k-Nearest Neighbors). Авторами решалась задача многоклассовой классификации (на 5 классов), поэтому лучшая модель с метрикой качества accuracy: 0.812. Работа [4] посвящена конкатенации двух нейросетевых концепций: сверточных нейронных сетей (одномерных) и рекуррентных нейронных сетей: LSTM (долгая краткосрочная память, long short-term memory). Рассмотренная модель показала результат accuracy: 0.8902. Решению этой задачи нейросетевым моделированием посвящена и статья [26], в которой объектом для исследований стала архитектура рекуррентной нейронной сети: GRU (управляемый рекуррентный блок, gated recurrent unit). Метрика качества модели accuracy: 0.717.

Основной вклад конкретно этой работы заключается в конкатенации современных подходов к решению поставленной задачи, мета моделировании с целью повышения качества и уверенности моделей и создании виджета, не имеющего аналогов и олицетворяющего законченность выпускной квали-

фикационной работы. Целями написания служили структуризация имеющихся знаний в исследуемой области и последующий эмпирически обусловленный выбор модели для применения. Конечная мета-модель, обученная на объединенном датасете отзывов с платформ IMDB и Amazon, общим объемом в 3.625 миллиона строк, оценивается метрикой качества accuracy в 0.965, что позволяет применять данную модель для анализа общественных мнений путем скраппинга комментариев с сайтов YouTube, Twitter и/или, например, Instagram. Кроме того, в работе рассмотрены приемы тематического моделирования LDA и BertTopic с целью дополнения ответа на вопрос: «Каково общественное мнение?» конкретизацией сферы его участия. Язык исследования: английский. Для адаптации результатов под другие языки необходима иного рода предобработка данных.

ОСНОВНАЯ ЧАСТЬ

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Постановка задачи

1.1.1 Постановка задачи классификации

Введем обозначения для формализации поставленной задачи. Пусть D – множество текстов (документов), W – конечное множество слов (терминов, токенов), изи которых состоят тексты, Y – конечное множество меток класса. Необходимо построить алгоритм, аппроксимирующий функцию $a: \ D \to Y$. Необходимо сопоставить каждому документу метку класса. Для обучения введены пары документ-метка $D^m = \{(d_1, y_1), \ldots, (d_m, y_m)\}$. Решив задачу классификации, можно будет сделать вывод о тональном отношении субъекта к исследуемой области.

1.1.2 Постановка задачи тематического моделирования

Помочь расширить понимание сущности общественного мнения способна задача тематического моделирования, заключающаяся в мягкой кластеризации текстов по латентно содержащимся в них темам. Формализуем вероятностную постановку задачи. Введем дополнительно T – конечное множество тем. Под темой будем понимать условное распределение на множестве терминов. Тогда $P(w|t)$, где $w \in W$ – слово из множества токенов, $t \in T$ – тема. Эту условную вероятность следует рассматривать как вероятность термина w в теме t. Наряду с этой условной вероятностью рассматрим необходимую для решения поставленной задачи апостериорную вероятность $P(t|d)$, означающую вероятность темы $t \in T$ в тексте $d \in D$. Таким образом можно получить дискретное вероятностное пространство $D \times W \times T$. В результате необходимо описать документ распределением вероятностей тем $(P(t|d))$, а каждую тему – дискретным распределением вероятности слов в теме $(P(w|t))$. Введем две гипотезы:

- Гипотеза «мешка слов». Порядок слов в тексте существенно не влияет на опреление темы. Это позволяет перейти к представлению, где $\forall d \in D$ ставится в соответствие словарь его уникальных токинов, причем $\forall w \in W$ подсчитывается значение частоты встречаемости w в документе d n_{dw};
- Гипотеза условной независимости. Токены в документы зависят от темы, но не зависят от документа. $P(w|t,d) = P(w|t)$.

Введем обозначения $P(w|t) = \phi_{wt}$, $P(t|d) = \theta_{td}$. Распишем вероятностное распределение термов в документе смесью введенных ранее распределений $P(w|t)$ и $P(t|d)$:

$$P(w|d) = \sum_{t \in T} P(w|t)P(t|d) = \sum_{t \in T} \phi_{wt}\theta_{td}.$$

В нахождении матриц $\Phi = (\phi_{wt})_{\dim W \times \dim T}$ и $\Theta = (\theta_{td})_{\dim T \times \dim D}$ и заключается задача тематического моделирования. Эта задача так же может быть описана через эвристику стохастического матричного разложение (рис. 1.1).

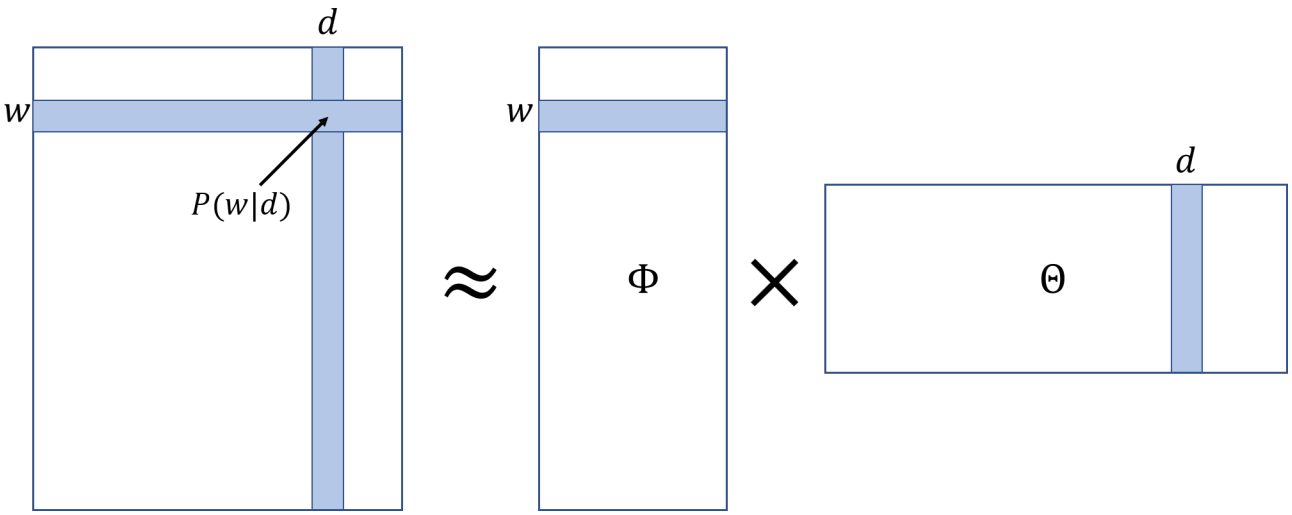


Рис. 1.1. Стохастическое разложение матриц

1.2 Предварительная обработка текстов

1.2.1 Общие сведения

Часто решение задачи, каким-либо образом связанной с моделированием естественного языка, требует предварительной обработки текста, содержащей в себе:

- приведение текста к единому регистру (обычно, нижнему);
- токенизацию – разбиение текстов на более мелкие единицы (токены), которые могут быть различными по размеру, в зависимости от задачи;
- морфологическую разметку: с целью разлилчать омонимы, несущие отличные семантические роли;
- удаление стоп-слов (слов, носящих излишне частотный или разреженный характер);
- лемматизацию (приведение слова к лемме – нормальной «словарной» форме) или стемминг (сокращения слова до его основы);
- векторизацию слова: представление токенов в качестве численных векторов для последующей обработки.

Так как выбранный язык исследования английский, то морфологическая разметка особого результативного прироста не даст и ее можно опустить. После приведения текста к единому регистру и токенизации необходимо составить словарь слов текстов. Для оптимального построения словаря выполняется редуцирование слов (лемматизация) и удаление излишне частых и разреженных слов. Рассмотрим эти процессы подробнее.

1.2.2 Лемматизация и удаление стоп-слов

Минимальную информацию о текстовых данных можно получить с помощью понятий частоты. Закон Ципфа гласит, что наиболее часто употребимых слов лишь небольшое количество, тогда как подавляющее большинство слов используются существенно реже: частота любого слова в корпусе текстов обратна пропорциональна его рангу в таблице частот слов этого текста. Рассмотрим один из датасетов, на которых будет происходить последующее обучение моделей: отзывы на фильмы с платформы IMDB. Построим закон Ципфа в логарифмических координатах рис. 1.2:

$$\ln r + \frac{1}{\alpha} \ln \omega_r = f(r),$$

где r – ранг слова в таблице частот, $\alpha \approx 1$ – константа, ω_r – абсолютная частота слова.

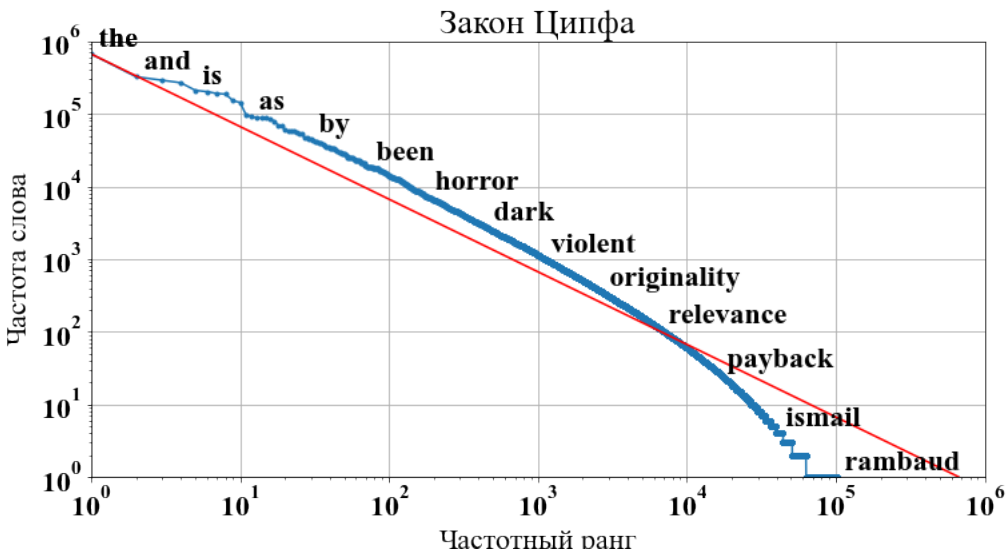
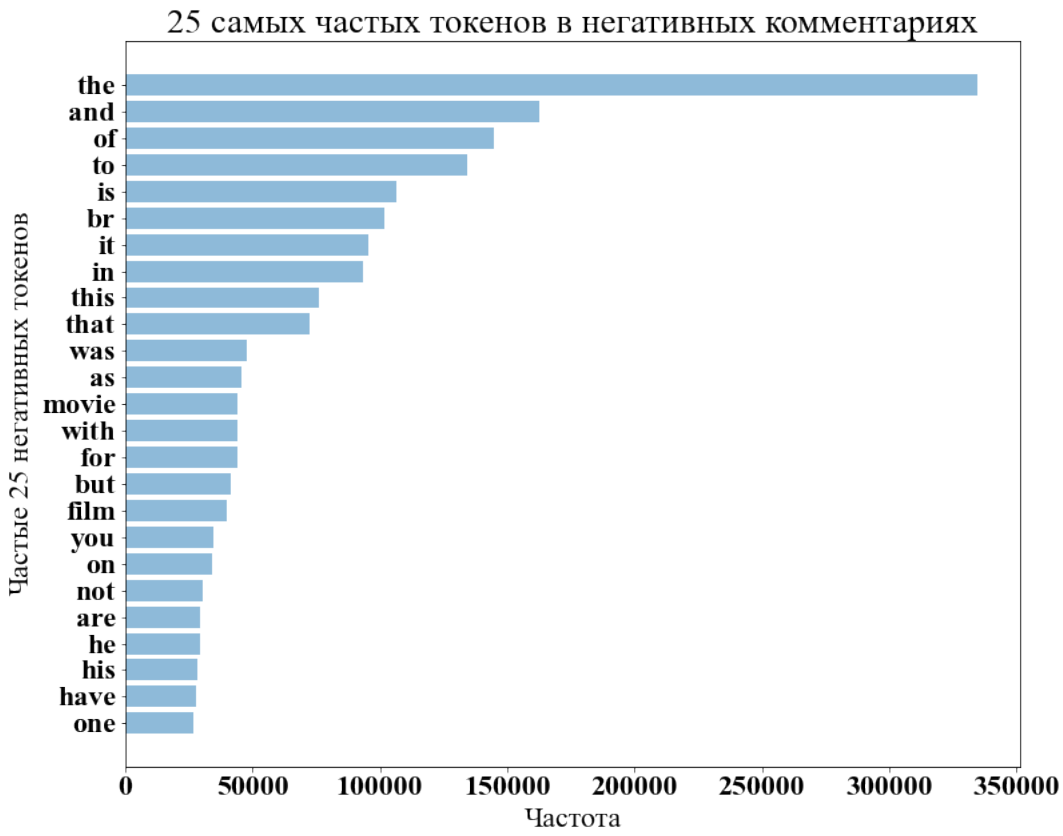


Рис. 1.2. Закон Ципфа на датасете IMDB

Явно заметен тренд к линейности, однако даже есть явное отклонение в районе слов с высоким ранжированием. Рассмотрим на распределения частот для позитивных и отрицательных комментариев (рис. 1.3):



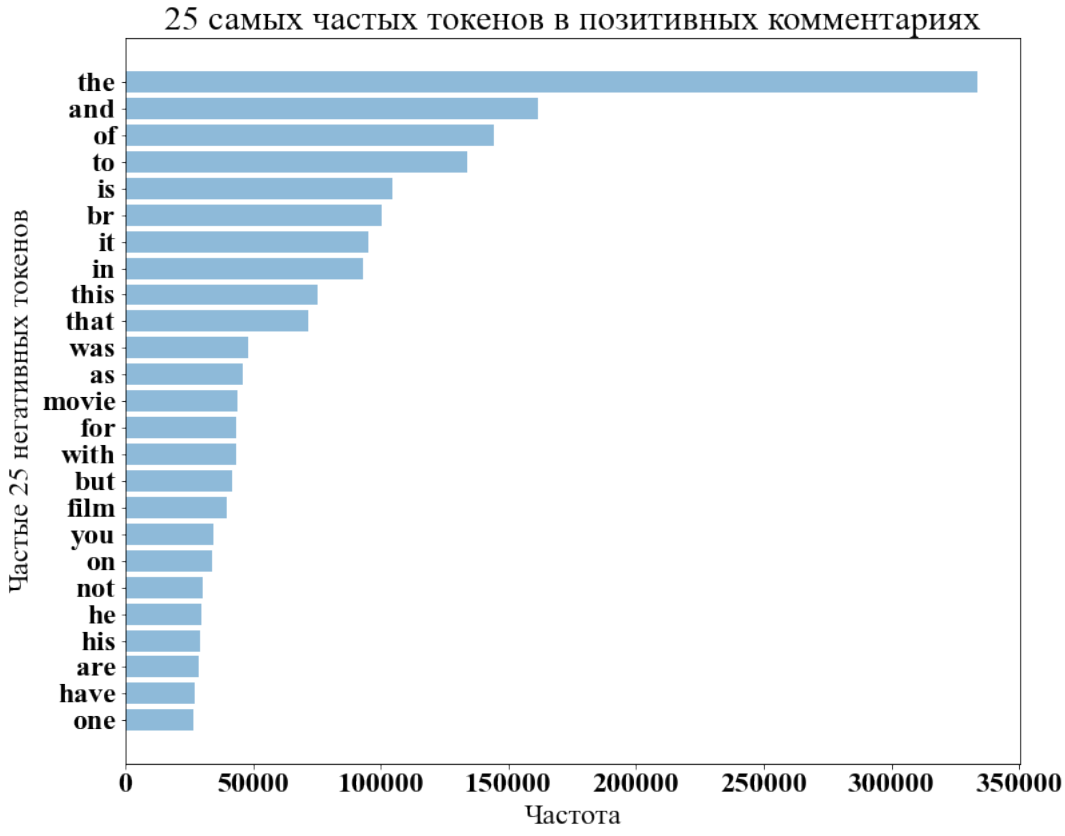


Рис. 1.3. Самые частые слова в позитивных и негативных комментариях

Слова настолько часто встречающиеся усложняют процесс поиска закономерностей в текстовых данных. Похожее влияние оказывают и излишне редко встречающиеся слова: зачастую это либо профессионализмы или диалектизмы, либо слова с ошибками. Поэтому, помимо оптимальности построения словаря, в рамках задачи решение о применении этих методов предобработки текстов приведет к упрощению поиска заномерностей в итак сложно структурированных данных. Рассмотрим частотные характеристи после лемматизации и удаления стоп-слов (рис. 1.4)

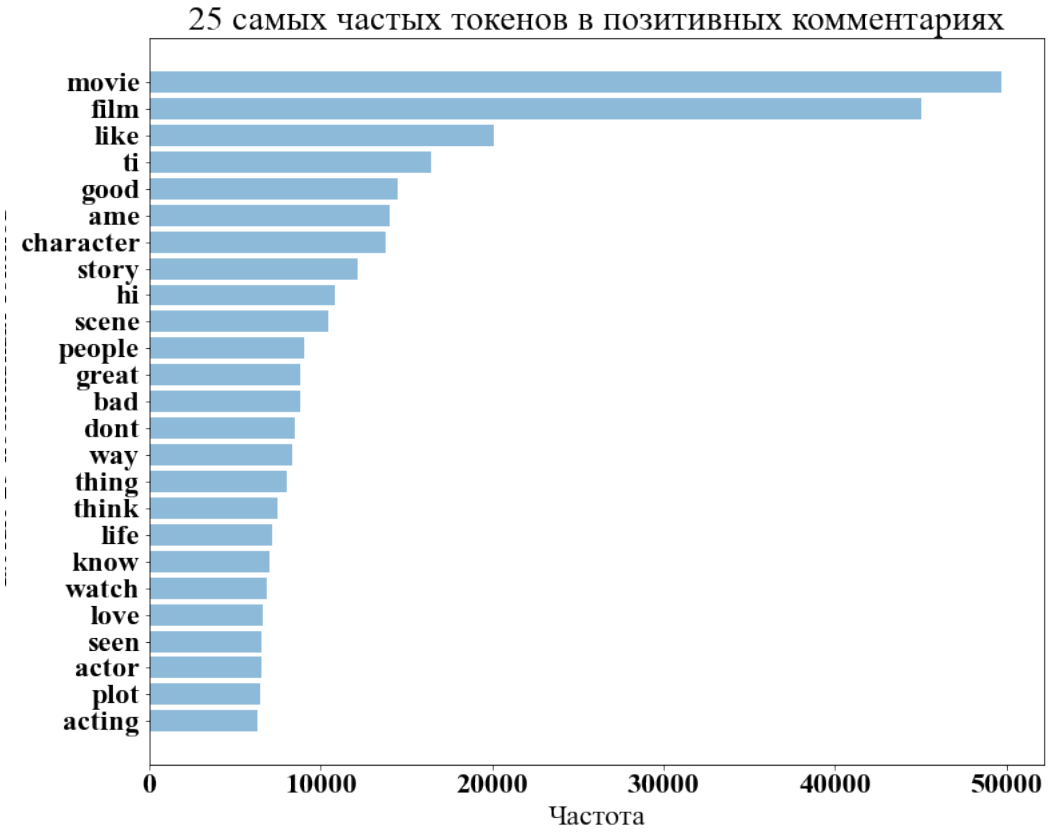


Рис. 1.4. Самые частые слова в позитивных комментариях после лемматизации и удаления стоп-слов

Позитивное влияние в виде упрощения структуры данных очевидно. Теперь необходимо представить предобработанные комментарии в векторном виде, чтобы в дальнейшем подавать их в модели глубокого обучения.

1.2.3 Векторизация слов

1.2.3.1 Bag of Words

Наиболее простым способом представления токенов в векторной форме является One Hot Encoding (OHE). Этот метод заключается в сопоставлении каждому токену вектора с размерностью, равной мощности словаря, и имеющего на всех позициях значения 0, кроме той, которая соответствует позиции самого токена в словаре, элемент с этим индексом равен 1 (рис. 1.5).

$$\text{Dictionary} = [\overset{0}{\text{start}} \quad \ldots \quad \overset{i\text{-ое слово}}{\text{network}} \quad \overset{i+1\text{-ое слово}}{\text{convolution}} \quad \ldots \quad \overset{m}{\text{end}}]$$

$$\text{network} = [\overset{0}{0} \quad \ldots \quad \overset{i}{1} \quad \overset{i+1}{0} \quad \ldots \quad \overset{m}{0}]$$

$$\text{convolution} = [\overset{0}{0} \quad \ldots \quad \overset{i}{0} \quad \overset{i+1}{1} \quad \ldots \quad \overset{m}{0}]$$

Рис. 1.5. One hot encoding (OHE) представление слов

Такой подход обладает достаточно закономерными недостатками:

- вектора неинформативны: все векторы полученного пространства ортогональны друг друга, что не является полезным, но большей информации о таком пространстве получить невозможно;
- большая размерность полученного пространства: с полученными векторами сложно оперировать функционально валидно из-за «проклятия размерности» [24];
- отсутствие возможности задать метрику в пространстве: слабая репрезентативность и отсутствие желаемых свойств пространству (арифметические операции над словами)

С целью получить количественную характеритику (частотную меру слов) применяется метод «мешок слов» [14]. В работе метода выдвигается

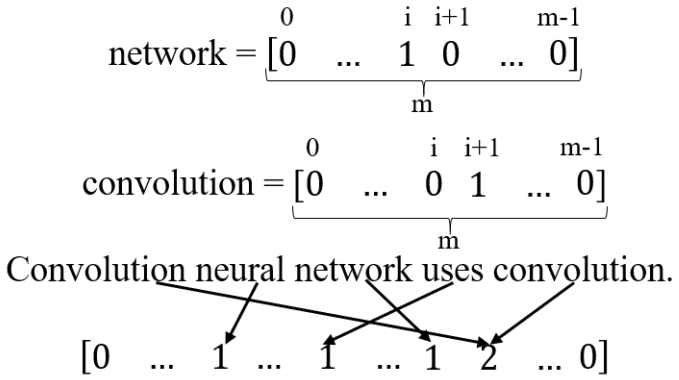


Рис. 1.6. Bag of Words текстовое представление

гипотеза о том, что порядок слов не несет в себе никакой информации. Получим векторы текстов путём сложения one-hot векторов токенов, в него непосредственно входящих рис. 1.6.

1.2.3.2 TF-IDF

Наиболее популярный способ в машинном обучении. Все тексты представляются в виде матрицы $T = (t)_{d,w} = TF - IDF(w, d, D)$, где $d \in D$ – текст, $w \in W$ – слово в тексте. TF-IDF – статистическая мера для оценки важности слова в контексте текста.

$$TF - IDF(w, d, D) = TF(w, d) \times IDF(w, D),$$

где TF – частота слова, которая оценивает слова в пределах документа (текста) по формуле:

$$TF(w, d) = \frac{n_i}{\sum\limits_k n_k},$$

где n_i – количество вхождений i-го слова в тексте. $\sum\limits_k n_k$ – общее число слов в тексте. IDF вычисляется по формуле:

$$IDF(w, D) = \log \frac{|D|}{\{d_i \in D | w \in d_i\}},$$

где $|D|$ длина корпуса текстов, $\{d_i \in D | w \in d_i\}$ – число текстов, в которых есть слово w_i.

1.2.3.3 Word2Vec

Word2Vec представляет собой полносвязную нейронную сеть с двумя различными архитектурами. В обоих случаях по окну проходят скользящим окном размера K. Среди K слов выделяют $K-1$ слов контекста и центральное слово. Первый тип Word2Vec Skip-gram (рис. 1.7) заключается в предсказывании контекста для слова:

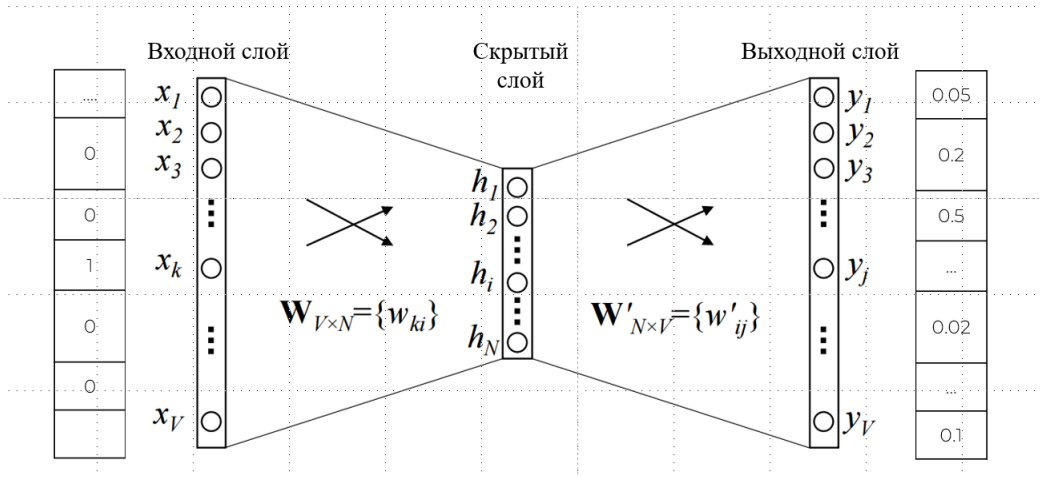


Рис. 1.7. Модель Word2Vec SkipGram [5]

На выходе имеем $K-1$ мультиномиальных распределений вида:

$$P(c_k|i) = \frac{e^{u_{kc_k}}}{\sum\limits_{j'=1}^{V} e^{u_{j'}}}$$

Функция ошибки (loss Function):

$$L = -\ln P\left(c_1, \ldots, c_n|i\right) = -\sum_{k=1}^{K} u_{kc_k} + n \ln \sum_{j'=1}^{V} e^{u'_j}.$$

Для обучения будем максимизировать функцию правдоподобия:

$$L(\theta) = \prod_{d\in D}\left(\prod_{c\in C(i)} P(c|d;\theta)\right) = \prod_{(d,c)\in D} P(c|d;\theta),$$

где $C(i)$ – множество возможных контекстных слов вокруг слова i при обходе скользящим окном всего текста. Вектор вероятности определяется

с помощью softmax вида:

$$P(c|d,\theta) = \frac{e^{\tilde{w}_c^T w_d}}{\sum c' e^{\tilde{w}_{c'} w_d}},$$

где $\tilde{w}_c$ – вектор слова из контекста c, который отличается от w_d. Так как слова редко встречаются в контексте самих себя, то была придумана идеология использования двух разных векторов одного и того же слова (вместо одного), в первом случае слово будет выступать в качестве центрального в контексте, в другом контекстуальным. Этот метод описан в [10]. Выразим максимум логарифма функции правдоподобия:

$$\arg\max_\theta \prod_{(d,c)\in D} P(c|d;\theta) = \arg\max_\theta \sum_{(d,c)\in D} \ln P(c|d;\theta) =$$

$$= \arg\max_\theta \sum_{(d,c)\in D} \sum_{(d,c)\in D} \left(e^{\tilde{w}_c^T w_d} - \ln \sum_{c'} e^{\tilde{w}_{c'}^T w_d} \right)$$

В статье [7] приведен стохастический алгоритм оптимизации процедуры матричного умножения: $\sum_c \tilde{w}_c^T w_d$ negative sampling, заключающийся в выборе только нескольких элементов скалярного произведения весов в качестве «отрицательных примеров». Выведем измененную функцию правдоподобия:

$$\arg\max_\theta \prod_{(d,c)\in D} P((d,c)\in D;\theta) = \arg\max_\theta$$

Для простоты выразим вероятности через сигмоидную функцию активации (так как задача бинарной классификации, то softmax может быть заменен на sigmoid). Имеем:

$$P((d,c)\in D;\theta) = \frac{1}{1+e^{-\tilde{w}_c^T w_d}}.$$

Добавим набор отрицательных данных:

$$\arg\max_\theta \prod_{(d,c)\in D} P((d,c)\in D;\theta) \prod_{(d,c)\in D'} P((d',c')\notin D;\theta)$$

Подставим с выраением для сигмоиды:

$$\arg\max_{\theta} \prod_{(d,c)\in D} P((d,c)\in D;\theta) \prod_{(d,c)\in D'} 1 - P((d',c')\notin D;\theta) =$$

$$= \arg\max_{\theta} \left[\sum_{(d,c)\in D} \ln P((d,c)\in D;\theta) + \sum_{(d',c')\in D'} \ln(1 - P((d',c')\notin D;\theta)) \right] =$$

$$= \arg\max_{\theta} \sum_{(d,c)\in D} \left[\ln\sigma(\tilde{w}_c^T w_i) + \sum_{(d,c')\in D'} \ln\sigma(-\tilde{w}_c^T w_d) \right]$$

Итоговая функция ошибки, для которой выполняется метод оптимизации:

$$L = \ln\sigma\left(\tilde{w}_c^T w_i\right) \sum_{(d,c')\in D'} \ln\sigma(-\tilde{w}_{c'}^T w_i.$$

Модель CBOW предсказывает слово для контекста (рис. 1.8):

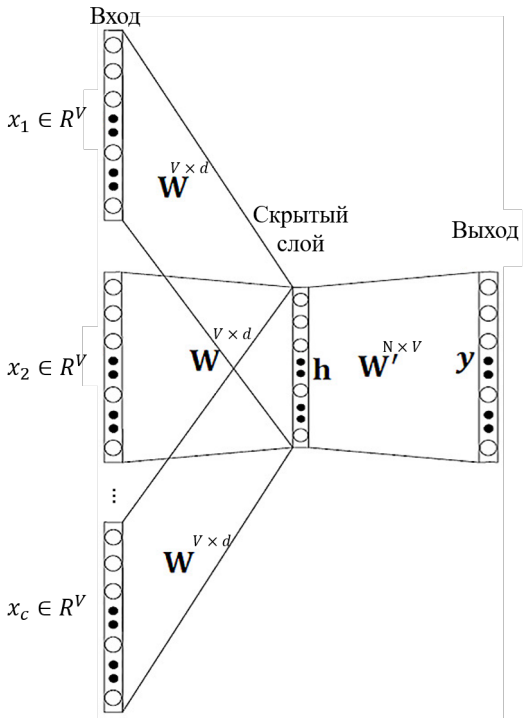


Рис. 1.8. Модель Word2Vec CBOW [7]

Найдем апостериорное распределение модели с помощью softmax:

$$P(i|c_1,\ldots,c_n) = \frac{e^{u_j}}{\sum\limits_{j=1}^{V} e^{u_j}}$$

Аппроксимируем с помощью функции потерь вида:

$$L = -\ln P(i|c_1, \ldots, c_n) = -u_j + \ln \sum_{j=1}^{V} e^{u_j}$$

1.2.3.4 Embedding

С другой стороны возможно внедрение слоя Embedding непосредственно в архитектуру нейронной сети рис. 1.9. Очевидным является вывод о лучших результатов конечной модели с таким подходом, поскольку в результате обучения будет происходить подбор гиперпараметров слоя под конкретную задачу. Однако объектом исследований является значимость релевантной разницы метрики качества модели с использованием предобученных эмбеддингов Word2Vec и обучаемых слоев. Если эта разница будет незначительна, то валидным является использование предобученных моделей векторизации, поскольку они латентно содержат информацию о большем объеме слов в силу обучении на огромном корпусе текстов.

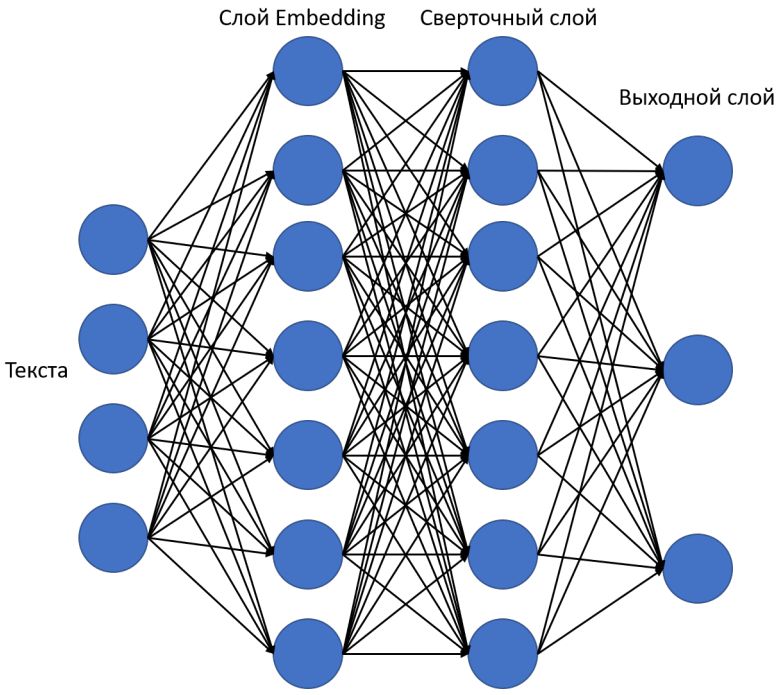


Рис. 1.9. Внедрение эмбеддингов непосредственно в архитектуру сети

1.3 Нейронные сети

1.3.1 Сети прямого распространения

Целью сети прямого распространения является аппроксимация некой неизвестной функции f. Сеть состоит из однотипных элементов, называемых нейронами. Искусственный нейрон состоит из двух элементов: взвешенного сумматора и нелинейной функции над ним рис. 1.10.

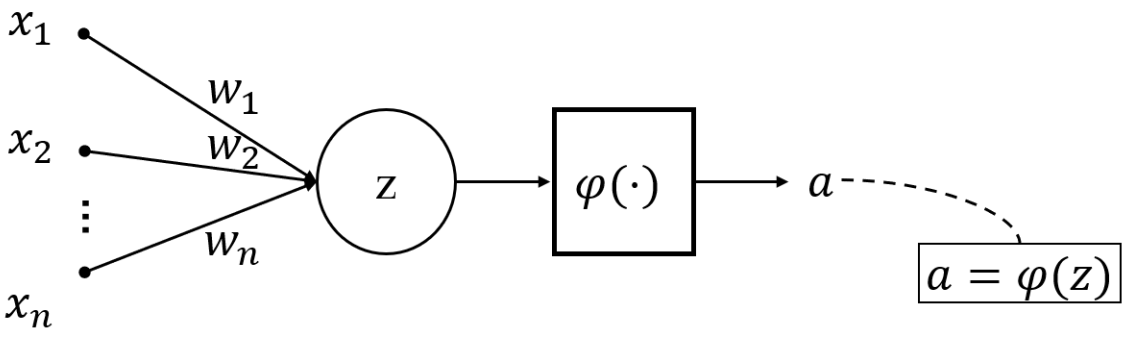


Рис. 1.10. Модель искусственного нейрона

Сумматор представляет собой выражение вида (скалярное произведение при введении $x_0 = 1$):

$$z = b + \sum_{i=1}^{n} w_i x_i \overset{x_0=1}{=} w^T x,$$

где $\vec{x} \in \mathbb{R}^n$ – вектор признаков, $\vec{w} =\in \mathbb{R}^n$ – вектор обучаемых параметров, $b \in \mathbb{R}$. Скопления нейронов называются слоями. Слой называется скрытым, если не определяет вывод модели. Линейная композиция линейных преобразований способна представлять только линейные функции. Чтобы обобщить модель на представление нелинейных функций, вводится так называемая функция активации над результатом взвешенного сумматора. Вопрос сводится к выбору отображения. В книге [11] приведены три совета по выбору функции активации.

1.3.2 Функции активации

- Бесконечномерная, используемая в ядерных методах (радиально-базисные) функция активации. Мощности такой функции хватит на аппроксимацию тренировочного набора, однако обобщаемость на тестовый датасет от этого страдает;
- Спроектировать функцию активацию φ самостоятельно;
- Обучение функции активации φ. Модель представляется в несколько ином виде:
$$y = f(x, \theta, w) = \varphi(x, \theta)^T w.$$
Преимуществом является выбор целого класса функций, однако придется подбирать новый гиперпараметр.

Композиция линейных и нелинейных преобразований предоставляют универсальную аппроксимацию. Универсальная теорема аппроксимации [17] утверждает, что сеть даже прямого распространения с линейным выходным слоем и нелинейным хотя одним слоем может аппроксимировать любую измеримую по Борелю функцию при условии, что в стеи достаточно блоков. Однако эта теорема не говорит о том, насколько должна быть большой сеть. В работе [2] предпринята попытка дать верхние границы размера сети с одним уровнем – экспоненциальное число блоков. Сети с одним слоем достаточно для универсального применения, однако этот слой может оказаться слишком большим: для решения этой проблемы под конкретные задачи берут конкретные архитектуры.

Для классификации n классов используется softmax – функция вида:
$$\varphi(x) = \frac{e^{x_j}}{\sum\limits_{j=1}^{V} e^{x_j}}.$$

Для бинарной классификации может быть использована сигмоидная функция активации:
$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

1.3.3 Обучение

Для обучения нейронной сети прямого распространения необходимо задать функцию ошибок между выходом сети $\tilde{y}$ и правильным результатом y. Алгоритм обратного распространения ошибки [15] позволяет передавать информацию о функции ошибки обратно по сети для вычисления градиентов. Опишем граф вычисления сети прямого распространения. Пример – граф для логистической регрессии рис. 1.11:

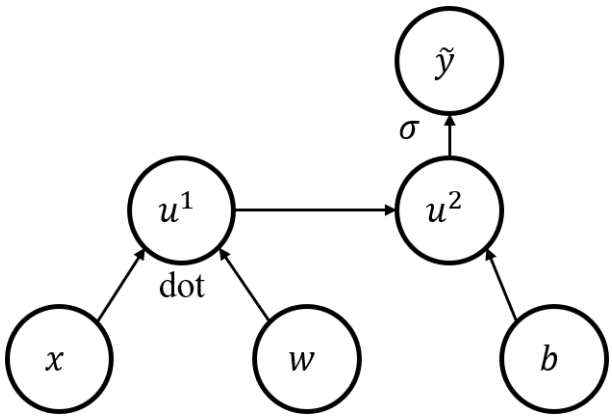


Рис. 1.11. Граф вычислений для логистической регрессии

Опишем алгоритм прохода по сети прямого распространения и вычисления функции ошибки (в общем случае с регуляризатором $\Omega(\theta)$):

Алгоритм 1 Прямое распространение с регуляризатором $\Omega(\theta)$

$W^{(i)},\ i \in \{1, \ldots, l\}$
$b^{(i)}, i \in \{1, \ldots, l\}$
$x \leftarrow$ input
$y \leftarrow$ label $s^{(0)} = x$
for $k = 1, \ldots, l$ **do**
  $a^{(k)} = b^{(k)} + W^{(k)} s^{(k-1)}$
  $s^k = f(a^{(k)})$
end for
$\tilde{y} = s^i$
$J = E(\tilde{y}, y) + \lambda\Omega(\theta)$

После вычисления прямого распостранения:

Алгоритм 2 Обратное распространение ошибки

$g \leftarrow \nabla_{\tilde{y}} J = \nabla_{\tilde{y}} E(\tilde{y}, y)$
for $k = l, \ldots, 1$ **do**
  $g \leftarrow \nabla_{a^{(k)}} J = g \odot f'(a^{(k)})$
  Вычислим градиенты для смещений и весов (с регуляризацией)
  $\nabla_{b^{(k)}} J = g + \lambda \nabla_{b^{(k)}} \Omega(\theta)$
  $\nabla_{W^{(k)}} J = g s^{(k-1)} + \lambda \nabla_{W^{(k)}} \Omega(\theta)$
  $g \leftarrow \nabla_{s^{(k-1)}} J = W^{(k)} g$
end for

Получив градиенты, можно интерпретировать эту информацию как указания о том, как следует изменить выходы слоев, чтобы уменьшить функцию ошибку.

1.4 Сверточные нейронные сети (CNN)

Созданные для задач компьютерного зрения сверточные нейронные сети [20, 21, 13] были достаточно продуктивно применены и в других сферах глубокого обучения, в том числе и NLP (natural language processing, обработка естественного языка). Использование их достигло максимума в ответ на ограниченность применения полносвязных сетей для обработки данных с сеточной топологией [11].

1.4.1 Операция свертки

Определение: Пусть $u, v : \mathbb{R}^n \to \mathbb{R}$ – две функции, интегрируемые относительно меры Лебега на пространстве $\mathbb{R}^n$. Тогда функция $u * v : \mathbb{R}^n \to \mathbb{R}$ называется сверткой и определяется соотношением:

$$(u * v)(x) := \int\limits_{\mathbb{R}^n} u(y) v(x - y) dy,$$

где y иногда называют давностью, u – входом, v – ядром свертки (kernel). Результат свертки именуется картой признаков (feature map). В задаче классификации текстов в силу плоского векторного представления слов будем рассматривать одномерную свертку. Для $n = 1$ формула примет вид:

$$(u * v)(x) := \int_{-\infty}^{\infty} u(y)v(x-y)dy$$

Кроме того, задача представляет собой конечный, дискретный случай, значит, окончательный вид формулы:

$$(u * v)(x) := \sum_{y=0}^{K} u(y)v(x-y),$$

где K – размер ядра.

Для иллюстрации операции свертки рассмотрим функции u, v как сигналы рис. 1.12. Сдвигая ядро v относительно входного сигнала u и по-

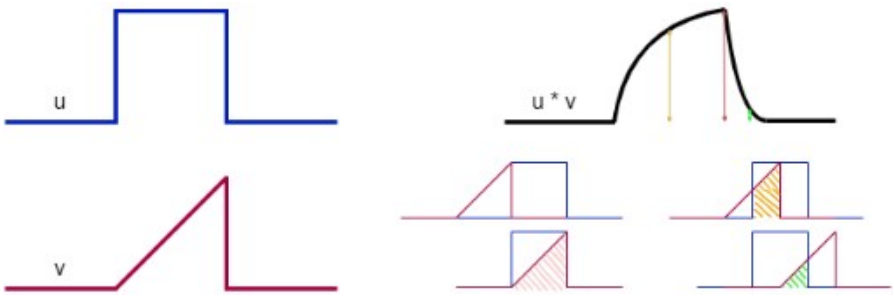


Рис. 1.12. Операция свертки между двумя сигналами u, v

считав скалярное произведение векторов, получим сходство фрагмента сигнала с ядром, что и является результатом свертки.

1.4.2 CNN для работы с изображениями

Для изображений характерно матричное представление, каждому элементу которого соответствует значение пикселя. На рис. 1.13 приведен случай с изображением размерности 5×5. Ядро свертки (размерности 3×3×1) прикладывается к каждому фрагменту изображения с задаваемым шагом (stride). На Рис. 1.13 приведен пример со значением шага, равным

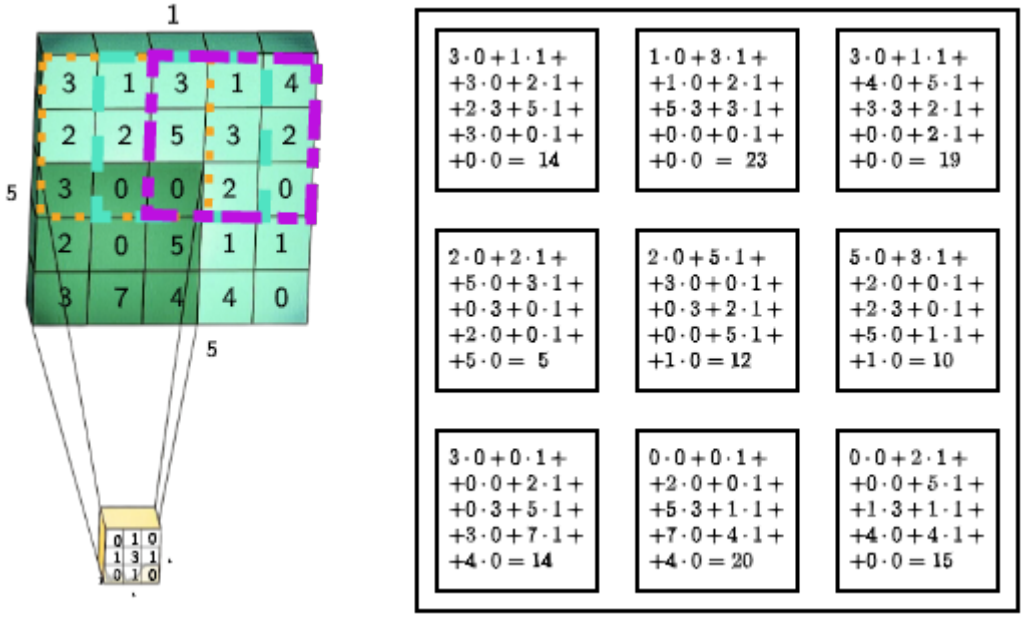


Рис. 1.13. Пример операции свертки для изображения $5 \times 5 \times 1$ с ядром свертки $3 \times 3 \times 1$

единице. Видно, что каждый нейрон связан лишь с частью пикселей, если речь идет о первом сверточном слое или нейронов, для последующих слоев (это свойство называют разреженной связностью). Необходимым становится обучение лишь значений ядер свертки, что значительно меньше, чем в случае традиционных нейронных сетей (полносвязных).

1.4.3 Одномерные CNN для работы с текстами

Как уже было сказано ранее, слова (после векторизации) представляют собой плоские вектора. Поэтому для задач обработки естественного языка применяются одномерные сверточные нейронные сети. Токены (в рассматриваемом случае, слова), представленные в векторной форме, конкатенируются в матрицу текста. Затем выбирается k строк матрицы, которые вытагиваются в плоский вектор. К нему прикладывается (берется скалярное произведение) одномерное ядро размерности $k \cdot s$, где s – размерность вектора токена. Этот процесс иллюстрирован на рис. 1.14 и рис. 1.15.

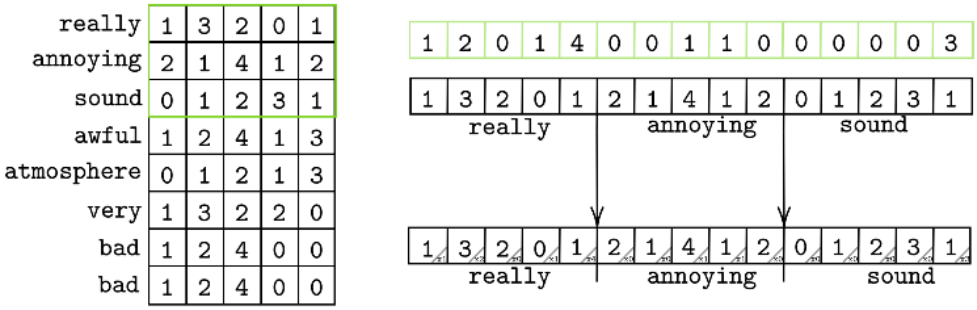


Рис. 1.14. Применение свертки для извлечения признаков из текста

Двигаясь таким образом по матрице текста, можно получить столбец значений – результатов операции одномерной свертки. Для получения большего количества столбцов, которые, в свою очередь, образуют матрицу признаков, используется несколько ядер.

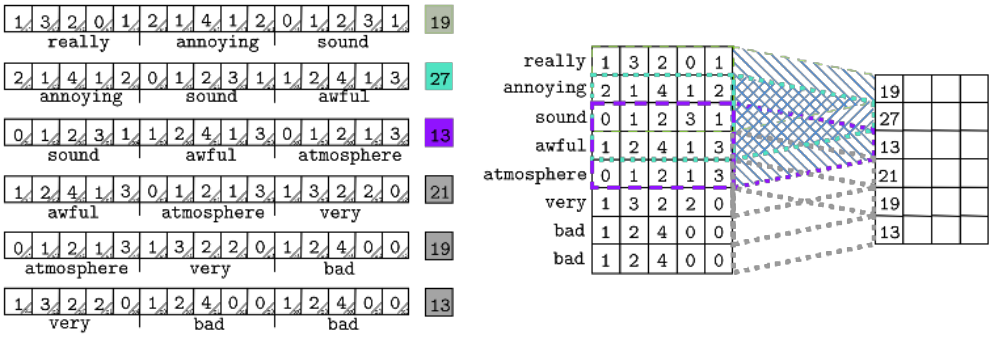


Рис. 1.15. Применение свертки для извлечения признаков из текста

После применения операции свертки как для изображений, так и для других структур, в том числе и текстов, возможно применение слоя субдискритизации (Pooling, подвыборки). Пулинг работает подобно сверточному слою, однако вместо скалярного произведения применяет другие функции на фрагменты, например, возвращает максимальный или средний выходы в прямоугольной окрестности (max-пулинг [27] и average-пулинг соответственно).

Max-Pooling:

$$f_{MP} = \max_i x_i.$$

Average-Pooling:

$$f_{AP} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Логика употребления слоя пулинга заключается в том, что выявленные сверточным слоем закономерности избыточны в своей подробности, значит, возможно уплотнение до менее подробного результата. Однако, если в задаче необходимо сохранять точную пространственную информацию, применение пулинга по всем признакам существенно увеличивает ошибку модели, что исследовано экспериментально [9]. Сверточный слой и слой пулинга, следующий за ним, образуют сверточный блок. Чем больше количество сверточных блоков, тем более абстрактной или высокоуровневой становится карта признаков. Заключим, что сверточная нейронная сеть достаточно успешно решает задачу извлечения признаков из данных. Экстраполируем этот вывод на одномерную плоскость для работы с текстами: сверточные нейронные сети, извлекая закономерности в текстовых данных, способны различать контекст. Причем ширину контекста можно приблизительно идеализировать шириной рецептивного поля (или пятна восприятия). На рис. 1.16 ширина рецептивного поля на первом слое равна трем, а ширина ре-цептивного поля на втором слое равна четырем, таким образом происходит расширение контекста. Однако всё же речь идет о, с точки зрения языка, маленьких паттернах, словосочетаниях и коротких фразах. Для масштабного и более полного моделирования языка необходимо делать очень глубокие сети, что неминуемо приводит к увеличению числа обучаемых параметров.

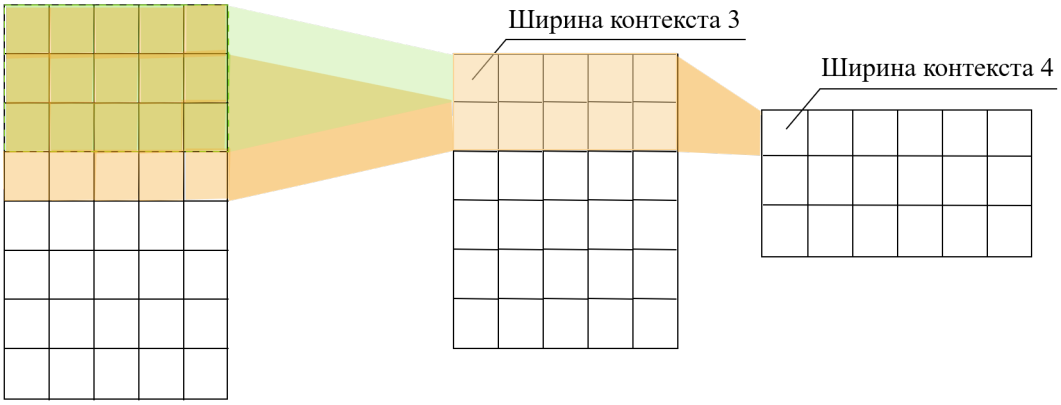


Рис. 1.16. Ширина рецептивных полей

Есть другой способ увеличения ширины контекста, который, так же как и пулинг, является спорным в употреблении в некоторых условиях задач: прореживание (dilation) [6]. Заключается он в применении свертки к

фрагменту, из которого удалена часть элементов. Например, возьмем k строк не подряд, а через одну, но значит это то, что мы пропускаем каждое второе слово, что не может однозначно сказаться на решаемой проблеме. Для решения задачи классификации после череды сверточных блоков (сверточный слой + слой пулинг) конечную карту признаков любой размерности, в зависимости от предыдущих слоев, необходимо растянуть в вектор и подать в полносвязный слой с функцией активации SoftMax (для классификации n-классов, $n \in \mathbb{N}$) или Sigmoid (для бинарной классификации), чтобы получить вероятности классов. Формула размерности карты признаков, учитывая всевозможные преобразования, после применения сверточного слоя или пулинга:

$$L_{\text{вых}} = \frac{L_{\text{вх}} + 2 \cdot \text{padding} - \text{dilation} \cdot (\text{kernel_size} - 1) - 1}{\text{stride}} + 1,$$

где padding(замощение) – параметр, используемый в случае, когда есть необходимость в сохранении размерности при выполнении свертки или пулинга: недостающие элементы для этого заполняются нулями.

1.4.4 Преимущества, недостатки и способы борьбы с ними

Сверточные нейронные сети хорошо находят закономерности в пространственных данных. Для текста эти зависимости именуются контекстом, идеализацией которого является ширина рецептивных полей сети. Так как вычислительные ресурсы ограничены, то построение очень глубоких сетей такой архитектуры затруднительно. Ширина рецептивных полей сверточных нейронных сетей позволяет определять частичный контекст текстовой информации: прореживание и пулинг дают возможность увеличить контекст, однако их использование требует достаточной осторожности и может привести к росту ошибки. Это не мешает отлично справляться с задачей классификации текстов одномерным сверточным сетям, что доказывают результаты. Но всё становится значительно хуже при попытках моделирования языка. К тому же, сверточные нейронные сети, как и традиционные, требуют входные данные фиксированной длины, что накладывает ограничение на плоское пространство текстов. В среднем, такие

ограничения не оказывают критичного влияния, но объемные текста могут нести важные для классификации детали в той своей части, которую не покрывает выбранный масштаб анализа.

1.5 Рекуррентные нейронные сети (RNN)

В случае сверточных нейронных сетей, текст позиционировался как пространственная структура. В случае рекуррентных нейронных сетей [23, 16] следует рассматривать его как последовательность, информация об элементе которой находится в зависимости от предшествующих ему элементов, другими словами, текст имеет направленное повествование. Подобная структура присуща не только текстам, но также и аудиосигналам или временным рядам.

1.5.1 Традиционные RNN

Основная идея рекуррентных нейронных сетей зиждется на введении скрытых состояний, и определении новых состояний через предыдущие. Таким образом, происходит латентное запоминание информации. То есть, в отличие от других нейронных сетей (сверточных, полносвязных), рекуррентная определяет своё состояние не только за счет входных данных, но и за счет предыдущих состояний.

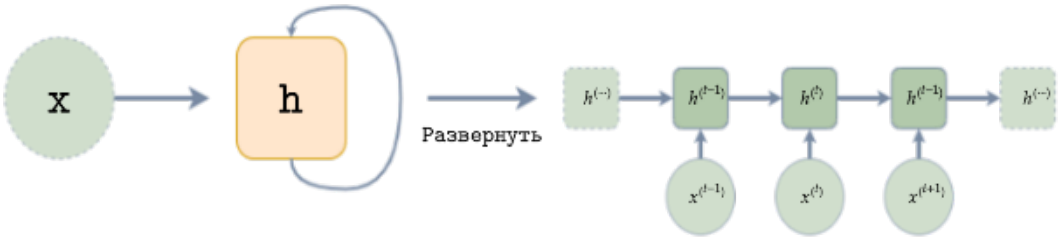


Рис. 1.17. Развернутая рекуррентная нейронная сеть без выходов

На каждом шаге рекуррентной сети:

а) Прочитать очередной элемент последовательности x^t, применить к

нему линейное преобразование:

$$z^t = W_{\text{input}} \cdot x^t;$$

б) Вычислить новое значение состояния, исходя из старого:

$$h^t = \text{act}\left(W_{\text{hidden}} \cdot h^{t-1} + z^t\right);$$

в) Выйти из рекуррентного шага:

$$y^t = W_{\text{output}} \cdot h^t.$$

Обучаемыми параметрами являются веса $W_{\text{input}}, W_{\text{hidden}}, W_{\text{output}}$, участвующие в линейных преобразованиях. Нулевое состояние сети может задаваться различными способами: исходя из экспериментальных показателей, лучше всего использовать малодисперсный шум [11].

Основными проблемами рекуррентных нейронных сетей во время обучения являются сходимости: возникают взрыв и затухание градиента. Рассмотрим эти ситуации подробнее.

1.5.2 Затухание и взрыв градиента

Рассмотрим прямой проход (forward pass) по классической рекуррентной нейронной сети (Vanilla RNN) на рис. 1.18

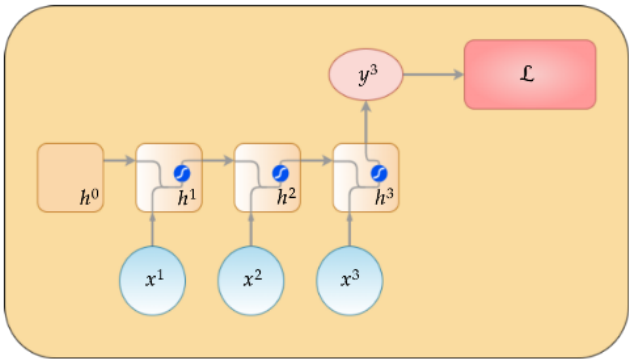


Рис. 1.18. Vanilla RNN

Введем обозначения. Входная последовательность $x^t \in \mathbb{R}$, скрытое состояние $h^t = f(w \cdot h^{t-1} + x^t)$, где f – нелинейная, дифференцируемая

функция активации, $w \in \mathbb{R}$ – параметр функции перехода, $y^t = g(h^t)$ – выход, функция потерь (функция ошибки, функционал качества) $\mathcal{L}(y^t)$:

а) $h^0 \in \mathbb{R}$;

б) $h^1 = f(w \cdot h^0 + x^1)$;

в) $h^2 = f(w \cdot h^1 + x^2) = f\left(w \cdot f\left(w \cdot h^1 + x^2\right) + x^3\right) =$
$= f\left(w \cdot f\left(w \cdot f\left(w \cdot h^0 + x^1\right) + x^2\right) + x^3\right).$

Выполним обратный проход (back propogation through time [25]):

$$\mathcal{L}(y^3) = \mathcal{L}\left(g\left(h^3\right)\right) = \mathcal{L}\left(g\left(f\left(w \cdot h^2 + x^3\right)\right)\right) = \mathcal{L}\left(g\left(f\left(w \cdot f\left(w \cdot h^1 + x^2\right) + x^3\right)\right)\right)$$

Применяя градиентный спуск, нам понадобится производная по параметрам:

$$\frac{\partial \mathcal{L}\left(y^3\right)}{\partial w} = \frac{\partial \mathcal{L}\left(y^3\right)}{\partial g} \cdot \frac{\partial g}{\partial w} = \frac{\partial L\left(y^3\right)}{\partial g} \cdot \frac{\partial g}{\partial f} \cdot \frac{\partial f\left(w \cdot h^2 + x^3\right)}{\partial w}$$

$$\begin{aligned}
\frac{\partial f\left(w \cdot h^2 + x^3\right)}{\partial w} &= f'\left(w \cdot h^2 + x^3\right) \cdot \left(w \cdot h^2 + x^3\right)' = f_3' \cdot \left(w \cdot h^2\right)' = \\
&= f_3'\left(w' \cdot h^2 + w \cdot h^{2'}\right) - f_3' \cdot \left(h^2 + w \cdot f\left(w \cdot h^1 + x^2\right)'\right) = \\
&= f_3' \cdot \left(h^2 + w \cdot f_2' \cdot \left(h^1 + w \cdot f\left(w \cdot h^0 + x^1\right)'\right)\right) = \\
&= \sum_{i=1}^{3} \left(h^{i-1} \cdot w^{3-i} \prod_{j=i}^{3} f_j'\right)
\end{aligned}$$

Именно последнее умножение является причиной затухания или взрыва градиента. Одной из часто употребляемых функций активаций является гиперболический тангенс, производная которого лежит в диапазоне: $0 < \tanh' < 1$. Умножение большого числа таких значений приведет к затуханию градиента (vanishing gradient) [25]. Борьбой с этим является ограничение количества скрытых состояний, что, по сути, ограничивает обрабатываемые последовательности. Основная идея рекуррентных нейронных сетей теряет смысл при подобном решении проблемы.

С другой стороны, если модуль произведения больше единицы, то произойдет взрыв градиента, приводящий к переполнению и падению точности. Борьба с такой проблемой является более очевидной – ограничение на сам градиент (ввод фиксированной величины, превышение которой ограничивает превысивший ее градиент этой величиной).

Затухающий градент так же связан с долгосрочными зависимостями [12]. Любые флуктуации в краткосрочных зависимостях буду подавлять долгосрочные. В случае, если модель способна представить долгосрочные зависимости, градиент долгосрочного взаимодействия будет экспоненциально ниже краткосрочного [11].

В связи с таким поведением градиента при обучении возникает явная проблема борьбы между мощностью и обучаемостью рекуррентных сетей. Ради ее решения были разработаны специализированные архитектуры рекуррентных нейронных сетей: LSTM [16] и GRU [19].

1.5.3 Долгая краткосрочная память (LSTM)

Сеть LSTM [16] использует концепцию вентильных нейронных сетей (gated Recurrent Neural Networks), основанную на идее контролированного выбора путей между состояниями. Путей, на которых не обнуляются и не устремляются в бесконечность производные. Веса связей скрытых состояний в такой структуре могут изменяться на каждом временном (или пространственном, зависит от структуры анализируемых данных) шаге. На рис. 1.19 и 1.20 представлены срез LSTM слоя и LSTM блок, соответственно.

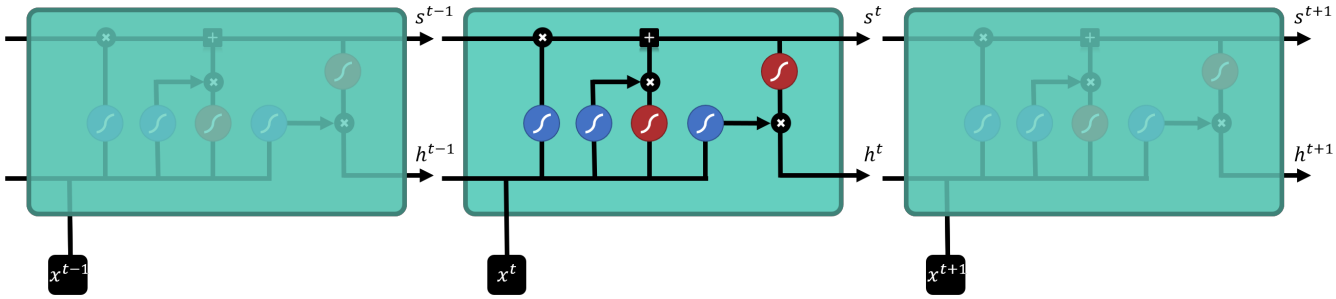


Рис. 1.19. Срез развернутого LSTM слоя.

Идея вентилей состоит в том, чтобы контролировать вектор значений за счет его умножения на вектор шлюза (вентиля), который управляет потоком ошибки. Целью является сохранение постоянного объема потока ошибки.

Запишем формальный прямой ход LSTM блока в момент времени t:

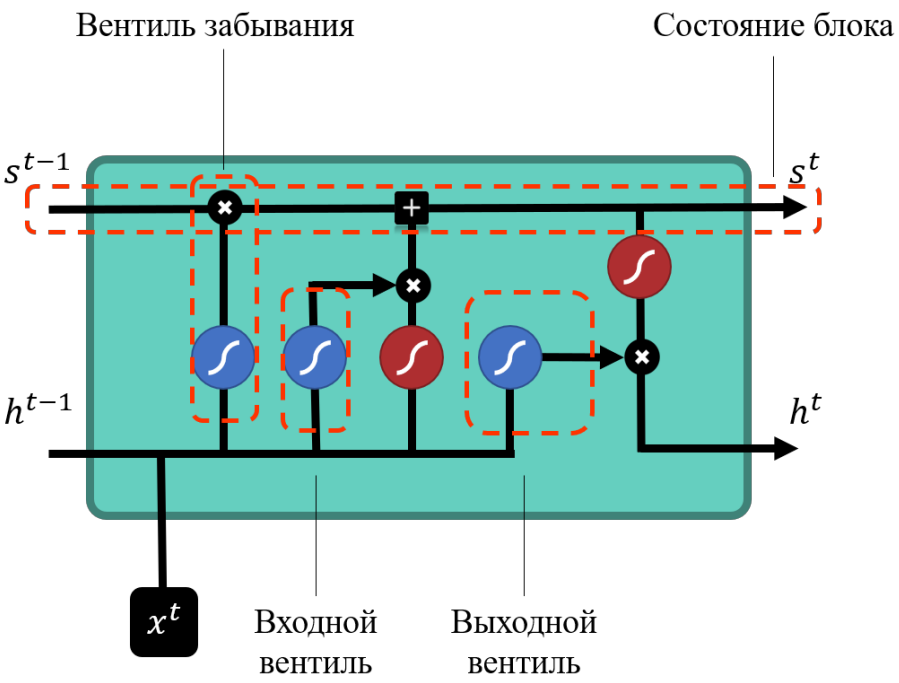


Рис. 1.20. LSTM блок.

- Конкатенируем прошлое скрытое состояние h^{t-1} и входной вектор x^t. Применяем линейное преобразование и нелинейную функцию активации:

$$f_t = \sigma\left(W_f \cdot \left[h^{t-1}, x^t\right] + b_f\right);$$
$$i_t = \sigma\left(W_i \left[h^{t-1}, x^t\right] + b_i\right);$$
$$\tilde{s}^t = \mathrm{act}\left(W_{\tilde{s}} \cdot \left[h^{t-1}, x^t\right] + b_{\tilde{s}}\right);$$
$$o_t = \sigma\left(W_o \cdot \left[h^{t-1}, x^t\right] + b_o\right);$$

act – функция активации (например, tanh);

- Получим сигнал как линейную комбинацию:

$$s^t = f_t \cdot s^{t-1} + i_t \cdot \tilde{s}^t;$$

- Выходим из рекуррентного шага:

$$h^t = o_t \cdot \tanh(s^t).$$

f_t – вентиль забывания (forgetting gate), обучаемый за счет весов W_f и сдвига b_f, определяющий насколько нужно забыть состояние с прошлого блока.

i_t – входной вентиль (input gate), обучаемый за счет весов W_i и сдвига b_i, ответственный за чувствительность ко входу: определяет, насколько важно для запоминания полученное новое состояние $\tilde{s}^t$. Состояние $\tilde{s}^t$ в рекуррентных сетях без вентильной парадигмы было бы выходом блока.
o_t – выходной вентиль (output gate), применяемый для увеличения точности аппроксимации нейронной сети за счет нелинейности функции активации.
За счет того, что сигмоида принимает значения в диапазоне: $0 < \sigma < 1$, все значения вентилей находятся в этом же диапазоне, что позволяет управлять амплитудой значений состояний, не меняя знак и контролируя объем потока ошибки.
Внутренние работы с новыми и старыми состояниями позволяют классифицировать их важность, как следствие настроить запоминание и забывание того, что нужно для решения задачи и бесполезно, соответственно. LSTM достаточно успешно представляет долгосрочные зависимости, моделирует язык, позволяет решать широкий спектр задач, но не лишена проблем. За счет появления новых внутренних обучаемых весов и сдвигов увеличивается количество обучаемых параметров в 4 раза по сравнению с обычной рекуррентной нейронной сетью. Поэтому LSTM часто может переобучаться и показывать плохие результаты на новых данных.

1.5.4 Управляемый рекуррентный блок (GRU)

Уменьшить количество обучаемых параметров без сильной потери качества удалось сети GRU (gated recurrent unit, управляемый рекуррентный блок). Основная идея остается той же – сохранение постоянства объема потока ошибки.

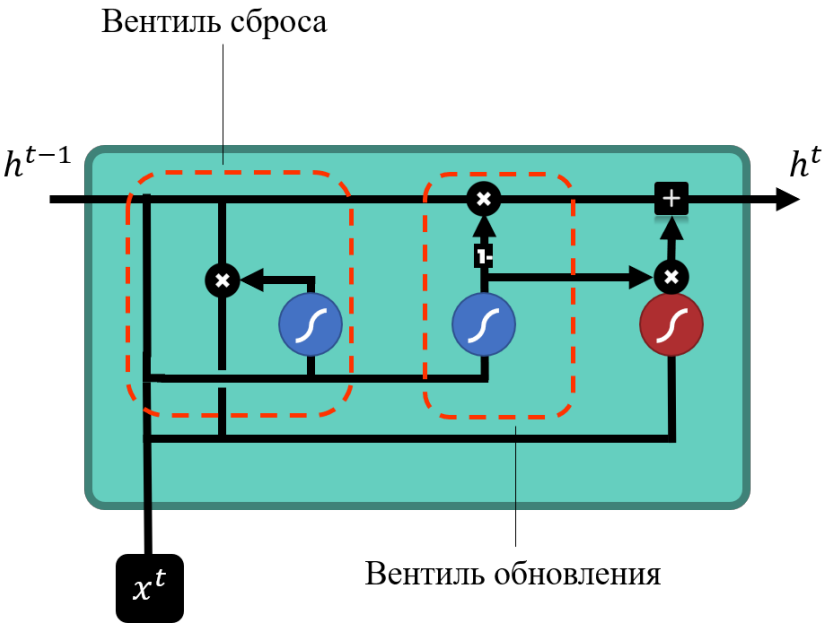


Рис. 1.21. GRU блок

Запишем формальный прямой ход GRU блока в момент времени t:

- Конкатенируем h^{t-1} и x^t. Применяем линейное преобразование и нелинейную функцию активации:

$$u_t = \sigma\left(W_u \cdot \left[x^t, h^{t-1}\right]\right);$$
$$r_t = \sigma\left(W_r \cdot \left[x^t, h^{t-1}\right]\right);$$
$$\tilde{h}^t = \mathrm{act}\left(W \cdot \left[x^t, r_t \cdot h^{t-1}\right]\right);$$

act – нелинейная функция активации (например, tanh);

- Получим сигнал как линейную комбинацию:

$$h^t = (1 - u_t) \cdot h^{t-1} + u_t \cdot \tilde{h}^t$$

- Выходим из рекуррентного шага.

u_t – вентиль обновления (update gate), решающий на каждом состоянии, оставить предыдущее значение или обновить.

r_t – вентиль сброса (reset gate), позволяющий контролировать поведение обновленного состояния. В результате, количество обучаемых параметров сократилось на треть (до порядка $6d^2$, где d – глубина реккурентного слоя). Сеть стала значительно проще в обучении, не потеряв мощности.

1.5.5 Недостатки и способы борьбы с ними

Традиционные рекуррентные нейронные сети не подвергаются распараллеливанию вычислений, так как происходит последовательная обработка данных. Из-за этого обучение происходит ощутимо дольше, чем для других нейронных сетей. Кроме того, из-за умножения большого числа параметров и производных функций активаций при обратном распространении ошибки возникают проблемы взрыва и затухания градиента, подходы для решения которых существуют, однако ухудшающие возможное качество моделей. Целенаправленно для решения этих проблем была разработана архитектура LSTM, которая, следуя парадигме вентильных нейронных сетей способна контролировать процессы запоминания и забывания последовательной информации, однако, это приводит к значительному увеличению обучаемых параметров. Решению этой проблемы LSTM посвящены архитектуры GRU и Simple-RNN, первая из которых использует концепцию одного вентиля, управляющего обоими процессами, и, на практике показывает схожее с LSTM качество, с ощутимым приростом производительности. Simple-RNN показывает качество хуже, хоть и имеет на треть меньшее количество параметров, чем GRU. Для задачи ВКР Simple-RNN был исключен, однако, не сказать о возможности уменьшения количества обучаемых параметров в рамках повествования о рекуррентных нейронных сетях считаю неправильным.

1.6 Тематическое моделирование

1.6.1 Вероятностный латентный семантический анализ (pLSA)

Запишем плотность распределения выборки $(d_i, w_i)_{i=1}^n$:

$$\prod_{i=1}^{n} p(d_i, w_i) = \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}},$$

что в сущности является функцией правдоподобия. Тогда максимизация логарифма правдоподобия запишется в виде:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w|d) p(d) \to \max_{\Phi, \Theta}.$$

Имеем задачу математического программирования:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \to \max_{\Phi, \Theta}$$

при ограничения неотрицательности и нормировки, накладываемыми за счет аксиоматики Колмогорова:

$$\phi_{wt} \geq 0, \quad \sum_{w \in W} \phi_{wt} = 1; \quad \theta_{td} \geq 0, \quad \sum_{t \in T} \theta_{td} = 1.$$

Из-за отсутствия корректной по Адамару постановки задачи необходимо применить регуляризацию Максимизация логарифма правдоподобия с регуляризацией ARTM:

$$\sum_{d,w} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + R(\Phi, \Theta) \to \max_{\Phi, \Theta}, \quad R(\Phi, \Theta) = \sum_{i} \tau_i R_i(\Phi, \Theta).$$

EM-алгоритм.

E-шаг:

$$p_{tdw} = p(t|d, w) = \operatorname*{norm}_{t \in T} (\phi_{wt} \theta_{td})$$

М-шаг:

$$\phi_{wt} = \underset{w\in W}{\operatorname{norm}}\left(n_{wt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}\right), \quad n_{wt} = \sum_{d\in D} n_{dw} p_{tdw}$$

$$\theta_{td} = \underset{t\in T}{\operatorname{norm}}\left(n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right), \quad n_{td} = \sum_{w\in d} n_{dw} p_{tdw},$$

где $\underset{x\in X}{\operatorname{norm}}(y_x) = \dfrac{\max\{y_x, 0\}}{\sum\limits_{k\in X}\max\{x_k, 0\}}$ Е-шаг: условные вероятности тем $p(t|d,w)$ вычисляются через апостериорные вероятности ϕ_{wt}, θ_{td} по формуле Байеса:

$$p(t|d,w) = \frac{p(w,t|d)}{p(w|d)} = \frac{p(w|t)p(t|d)}{p(w|d)} = \frac{\phi_{wt}\theta_{td}}{\sum\limits_{k}\phi_{wk}\theta_{kd}}.$$

М-шаг: при $R = 0$ частотные характеристики вычисляются суммированием $n_{tdw} = n_{dw}p(t|d,w)$:

$$\phi_{wt} = \frac{n_{wt}}{n_t}, \quad n_{wt} = \sum_{d\in D} n_{tdw}, \quad n_t = \sum_{w\in W} n_{wt}$$

$$\theta_{td} = \frac{n_{td}}{n_d}, \quad n_{td} = \sum_{w\in d} n_{tdw}, \quad n_d = \sum_{t\in T} n_{td}.$$

Для pLSA:

$$R(\Phi,\Theta) = 0.$$

М-шаг – частотные оценки условных вероятностей:

$$\phi_{wt} = \underset{w\in W}{\operatorname{norm}}(n_{wt}), \quad \theta_{td} = \underset{t\in T}{\operatorname{norm}}(n_{td}).$$

1.6.2 Латентное размещение Дирихле (LDA)

Регуляризация:

$$R(\Phi,\Theta) = \sum_{t,w}(\beta_w - 1)\ln\phi_{wt} + \sum_{d,t}(\alpha_t - 1)\ln\theta_{td}.$$

М-шаг – сглаженные частотные характеристики с параметрами β_w, α_t:

$$\phi_{wt} = \underset{w\in W}{\operatorname{norm}}(n_{wt} + \beta_w - 1), \quad \theta_{td} = \underset{t\in T}{\operatorname{norm}}(n_{td} + \alpha_t - 1).$$

Гипотеза: Векторы $\phi_t = (\phi_{wt})$ и $\theta_d = (\theta_{td})$ порождаются распределениями Дирихле, причем $\alpha \in \mathbb{R}^{|T|},\ \beta \in \mathbb{R}^{|W|}$:

$$\mathrm{Dir}(\phi_\mathrm{t}|\beta) = \frac{\Gamma(\beta_0)}{\prod_{w\in W}\Gamma(\beta_w)} \prod_{w\in W} \phi_{wt}^{\beta_w - 1},\ \ \phi_{wt} > 0;\ \ \beta_0 = \sum_{w\in W} \beta_w,\ \beta_t > 0;$$

$$\mathrm{Dir}(\theta_\mathrm{d}|\alpha) = \frac{\Gamma(\alpha_0)}{\prod_{t\in T}\Gamma(\alpha_t)} \prod_{t\in T} \theta_{td}^{\alpha_t - 1},\ \ \theta_{td} > 0;\ \ \alpha_0 = \sum_{t\in T} \alpha_t,\ \alpha_t > 0;$$

Запишем апостериорную вероятность для LDA:

$$\ln \prod_{d\in D} \prod_{w\in d} p(w, d|\Phi, \Theta)^{n_{dw}} = \prod_{t\in T} \mathrm{Dir}(\phi_t|\beta) \prod_{d\in D} \mathrm{Dir}(\theta_d|\alpha) \to \max_{\Phi,\Theta}$$

- при $\beta_w > 1,\ \alpha_t > 1$ – сглаживание;
- при $\beta \in (0,1),\ \alpha \in (0,1)$ – слабое разреживание,
- при $\beta = \alpha = 1$ – pLSA.

1.6.3 BERT Topic

Другим подходом к решению задачи тематического моделирования является кластеризация в векторном пространстве текстов с нейросетевой векторизацией текстов. Одной из самых актуальных моделей в решении поставленной задачи является BERTopic. Этапы работы алгоритма:

а) Векторизация текстов (предобученные модели Bert Embeddings [3]);

б) Понижение размерности (методом UMAP [22]); Во время работы алгоритм производит построение взвешенного графа, соединяя только те вершины, которые являются ближайшими соседями. Пусть $X \in \mathbb{R}^n$ – вершины графа (вектора слов в n-мерном пространстве). $T \in \mathbb{R}^k$ – множество k-ближайших к x_i $\forall i$. Для каждого объекта рассчитывается расстояние до ближайшего соседа и величина σ_i по формулам, соответственно:

$$\rho_i = \min_{t\in T} d(x_i, t),$$

$$\sum_i e^{-\left(\frac{d(x_i, y) - \rho_i}{\sigma_i}\right)} = \log_2 k$$

Вес ребра, соединяющего x_i и его соседа t_k определяется по формуле:

$$w_{ij} = e^{-\left(\frac{d(x_i, t_j) - \rho_i}{\sigma_i}\right)}.$$

Множество ребер построенного графа является нечетким с вероятностью существования ребра между двумя вершинами как функцией принадлежности. Вес ребра между двумя вершинами определяется по формуле:

$$w_{x_i,x_j} = w(x_j \to x_i) + w(x_i \to x_j) - w(x_i \to x_j) \cdot w(x_j \to x_i).$$

Затем происходит построение графа в низкоразмерном пространстве и приближение его с исходным путем минимизации суммы дивергенций Кульбака-Лейблера для каждого из ребер e из исходного и нового нечетких множеств:

$$\sum_{e \in E} w_h(e) \log \frac{w_h(e)}{w_l(e)} + (1 - w_h(e)) \log \left(\frac{1 - w_h(e)}{1 - w_l(e)}\right) \to \min_{w_l},$$

$w_h(e)$ – функция принажлежности нечеткого множества из ребер в высокоразмерном пространстве, $w_l(e)$ – функция принадлежности нечеткого множества из ребер в низкоразмерном пространстве. Задача оптимизации решается с помощью стохастического градиентного спуска.

в) Кластеризация (метод HDBSCAN) [18]:

Рис. 1.22. Результат работы HDBSCAN

Формализуем HDBSCAN.

а) Пусть $U_\epsilon(x)$ – ϵ-окрестность точки x, которая до этого не просматривалась.

б) Если точка x содержит $minPts$ соседей (где $minPts$ и метрика, по которой находится расстояние между точками – задаваемые гиперпараметры) или более, то образуется кластер

в) Иначе точка помечается шумом. Возвращаемся к п.1.

г) Повторяем, п.2-3 до момента, пока не получится связный кластер (между любыми соседями есть путь, как минимум, один);

д) Повторяем п.1-4 для новой не просмотренной точки.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Используемые инструменты

- Python 3
- NumPy
- Pandas
- Matplotlib
- NLTK
- Spacy
- Scikit-learn
- TensorFlow
- JavaScript
- HTML и CSS
- PyTorch
- Bootstrap5
- Flask
- torchtext

2.2 Экспериментальные результаты

2.2.1 Используемые данные

В задаче сентимент-анализа рассматривались датасеты для бинарной классификации (рецензии на кино с ресурса IMDB – 50 тысяч строк и отзывы YELP – 560 тысяч строк), для задачи классификации на три класса – датасет комментариев социальной сети Twitter (74681 комментарий). В задаче тематического моделирования рассматривался предобработанный для обучения моделей классификации датасет IMDB. Для ансамблирования нейросетевых моделей сентимент-анализа были сконкатенированы датасеты Amazon и IMDB – общее количество строк 3.625 миллиона.

2.2.2 Одномерные сверточные нейронные сети

В Таблице 2.1 приведены метрики качества accuracy исследуемых архитектур на этапах обучения, валидации и тестирования моделей.

Таблица 2.1 Метрика accuracy на разных этапах обучения

Данные	Модели								
	CNN (1 слой)			CNN (3 + 3 пулинг)			CNN (5+5)		
	Train	Val	Test	Train	Val	Test	Train	Val	Test
IMDB (2)	**0.9996**	0.8757	0.889	0.997	**0.8994**	0.8765	0.9963	0.8991	**0.8996**
Twitter (3)	0.913	0,8338	0.8447	**0.945**	0.8491	0.8312	0.93	**0.8542**	**0.8771**
Yelp (2)	0.5312	0.5671	0.5431	0.5538	0.534	**0.613**	**0.5652**	**0.5843**	0.548
Epochs	5			5			5		

Таблица 2.2 содержит сравнение современных фреймворков для обучения нейронных сетей PyTorch и Tensorflow. В ней приведены длительности обучения моделей.

Таблица 2.2 Затраченное время на разных фреймворках

	Модели						
Данные	CNN (1 слой)		CNN (3+3 пулинг)		CNN (5+5)		Объем
	TF	PyTorch	TF	PyTorch	TF	PyTorch	
IMDB (2)	221	81	307	100	383	132	50000
Twitter (3)	385	313	497	433	754	532	75000
Yelp (2)	100421	120212	220053	189429	315371	435341	560000

На рис. 2.1 и рис. 2.2 изображены поведения ошибки и метрики качества в зависимости от выбора размера пакета и оптимизатора, соответственно.

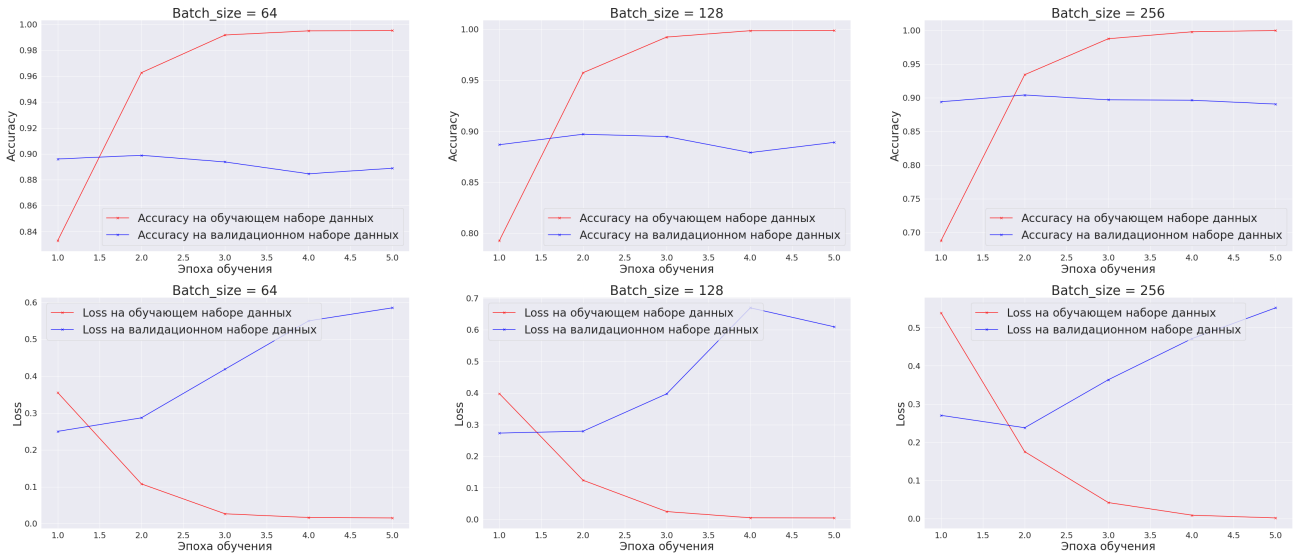


Рис. 2.1. Поведение accuracy и loss в зависимости от batch size

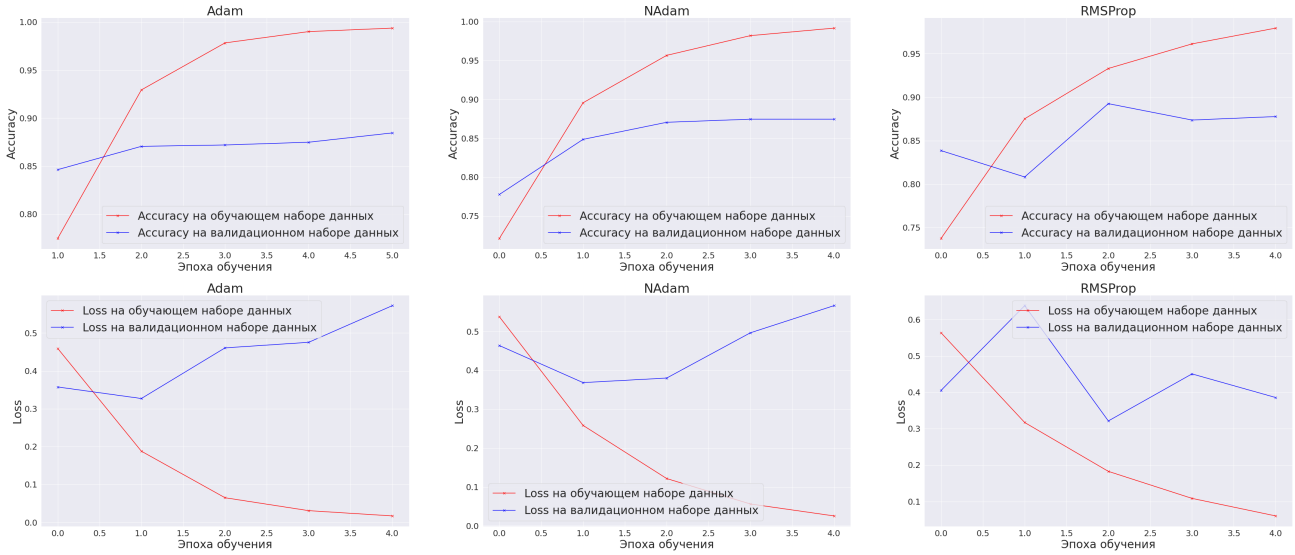


Рис. 2.2. Поведение accuracy и loss в зависимости от выбора оптимизатора

2.2.3 Рекуррентные нейронные сети

В Таблице 2.3 приведены метрики качества accuracy исследуемых архитектур на этапах обучения, валидации и тестирования моделей.

Таблица 2.3 Метрика accuracy на разных этапах обучения

Данные	Модели								
	LSTM (1 блок)			LSTM (2 блока)			CNN + LSTM (1 слой + 1 пулинг + 1 блок)		
	Train	Val	Test	Train	Val	Test	Train	Val	Test
IMDB (2)	0.9968	0.8809	0.8573	0.9533	**0.8834**	0.8673	**0.9982**	0.88	**0.8765**
Twitter (3)	**0.855**	0.84	0.778	0.8431	0.8393	**0.841**	0.815	**0.8542**	0.8175
Yelp (2)	**0.9172**	**0.9472**	**0.945**	0.91	0.864	0.8621	-	**-**	-
Epochs	13			10			10		

Таблица 2.4 содержит сравнение современных фреймворков для обучения нейронных сетей PyTorch и Tensorflow.

Таблица 2.4 Затраченное время на разных фреймворках

	Модели						
Данные	LSTM (1 блок)		LSTM (2 блока)		LSTM + CNN		Объем
	TF	PyTorch	TF	PyTorch	TF	PyTorch	
IMDB (2)	225	201	8400	8513	1800	1680	50000
Twitter (3)	541	650	28326	31652	3366	3254	75000
Yelp (2)	432002	542331	479032	482151	-	-	560000

На рис. 2.3 и рис. 2.2 изображены поведения ошибки и метрики качества в зависимости от выбора размера пакета и оптимизатора, соответственно.

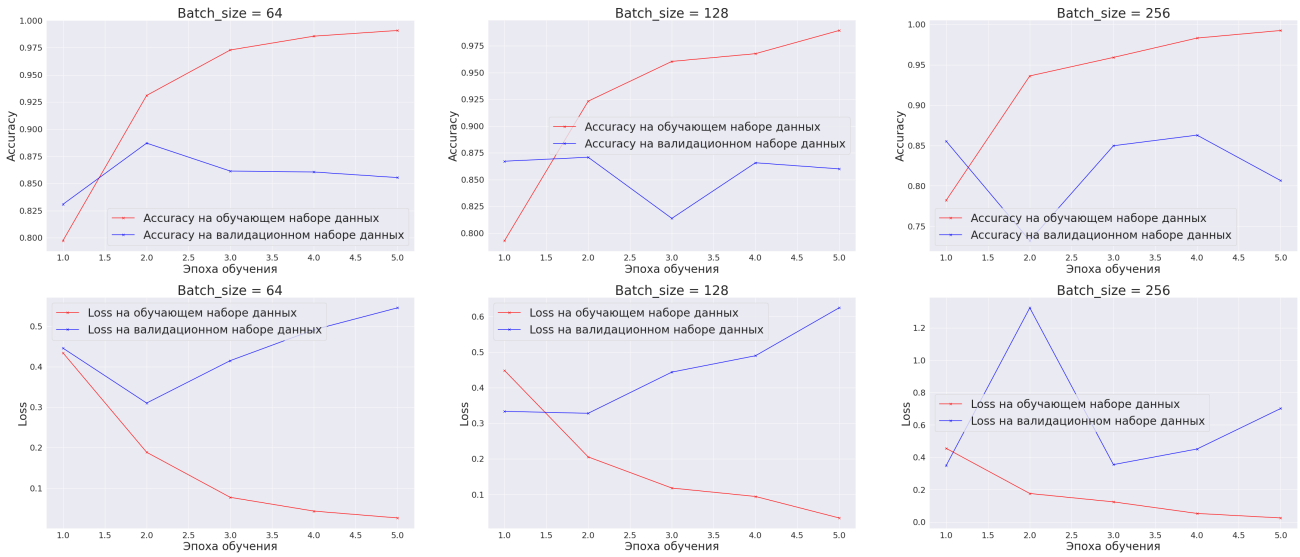


Рис. 2.3. Поведение accuracy и loss в зависимости от batch size

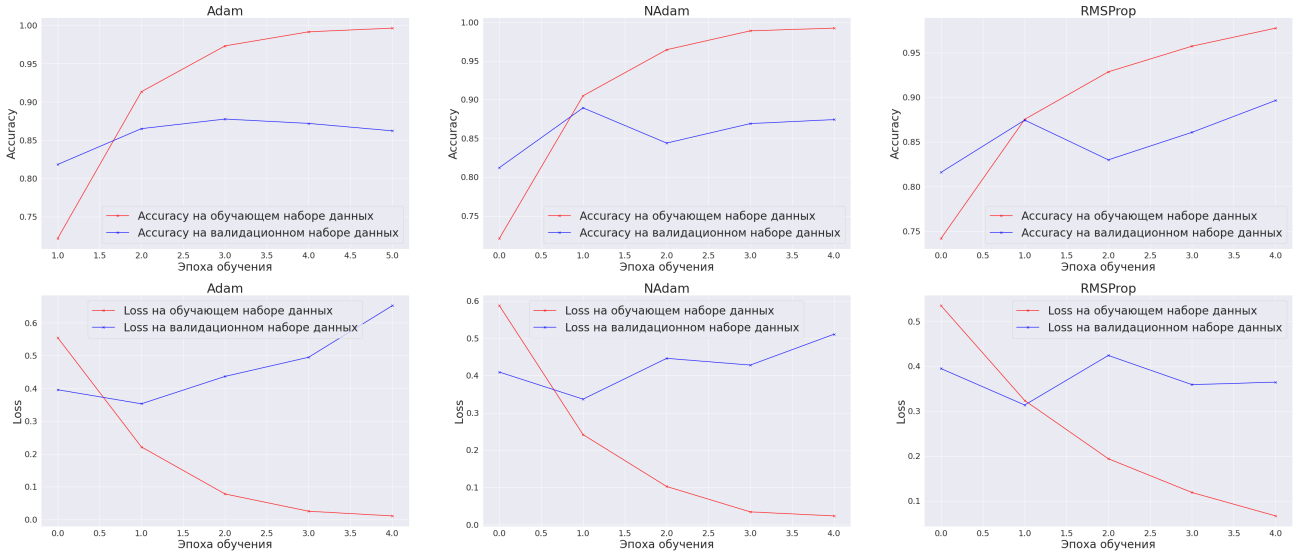


Рис. 2.4. Поведение accuracy и loss в зависимости от выбора оптимизатора

В Таблице 2.5 приведены метрики качества accuracy исследуемых архитектур на этапах обучения, валидации и тестирования моделей.

Таблица 2.5 Метрика accuracy на разных этапах обучения

Данные	Модели								
	GRU (1 блок)			GRU (2 блока)			CNN + GRU (5 слой + 5 пулинг + 1 блок)		
	Train	Val	Test	Train	Val	Test	Train	Val	Test
IMDB (2)	0.9751	**0.9612**	**0.9353**	0.9903	0.8764	0.8931	**0.9969**	0.8851	0.9185
Twitter (3)	0.8788	0.8338	0.8447	0.9937	0.9605	0.9574	0.93	**0.872**	**0.8771**
Yelp (2)	**0.9670**	**0.9562**	**0.9459**	0.954	0.943	0.9126	-	**-**	-
Epochs	5			5			5		

Таблица 2.6 содержит сравнение современных фреймворков для обучения нейронных сетей PyTorch и Tensorflow. В ней приведены длительности обучения моделей.

Таблица 2.6 Затраченное время на разных фреймворках

	Модели						
Данные	GRU (1 блок)		GRU (2 блока)		GRU + CNN		Объем
	TF	PyTorch	TF	PyTorch	TF	PyTorch	
IMDB (2)	189	204	569	642	244	183	50000
Twitter (3)	546	498	759	802	650	663	75000
Yelp (2)	433200	454233	679036	592431	-	-	560000

На рис. 2.5 и рис. 2.6 изображены поведения ошибки и метрики качества в зависимости от выбора размера пакета и оптимизатора, соответственно.

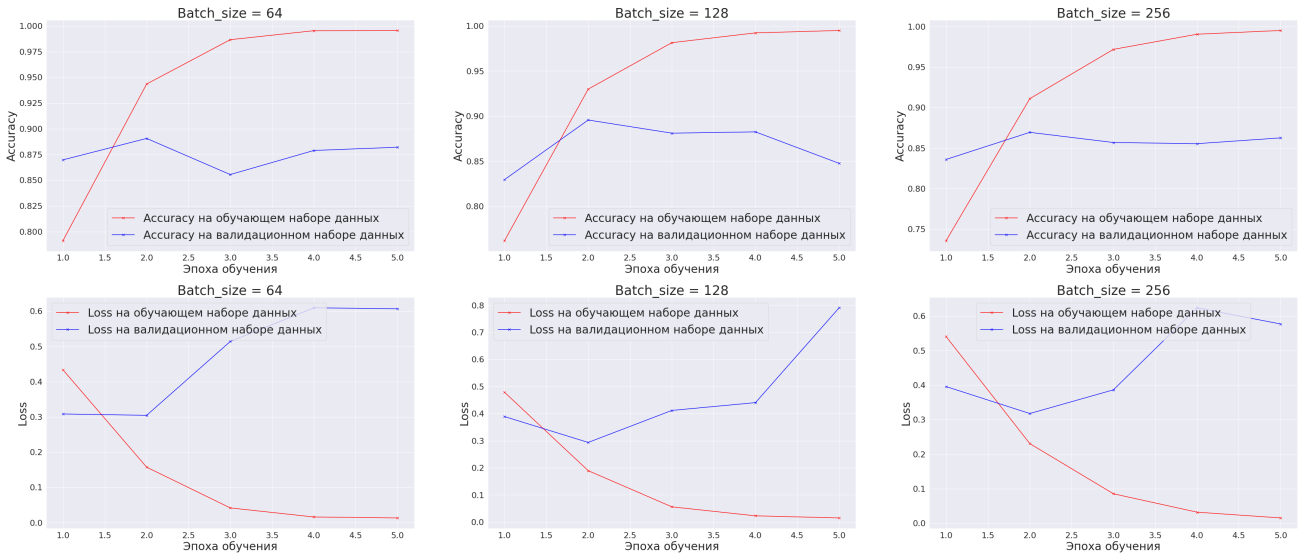


Рис. 2.5. Поведение accuracy и loss в зависимости от batch size

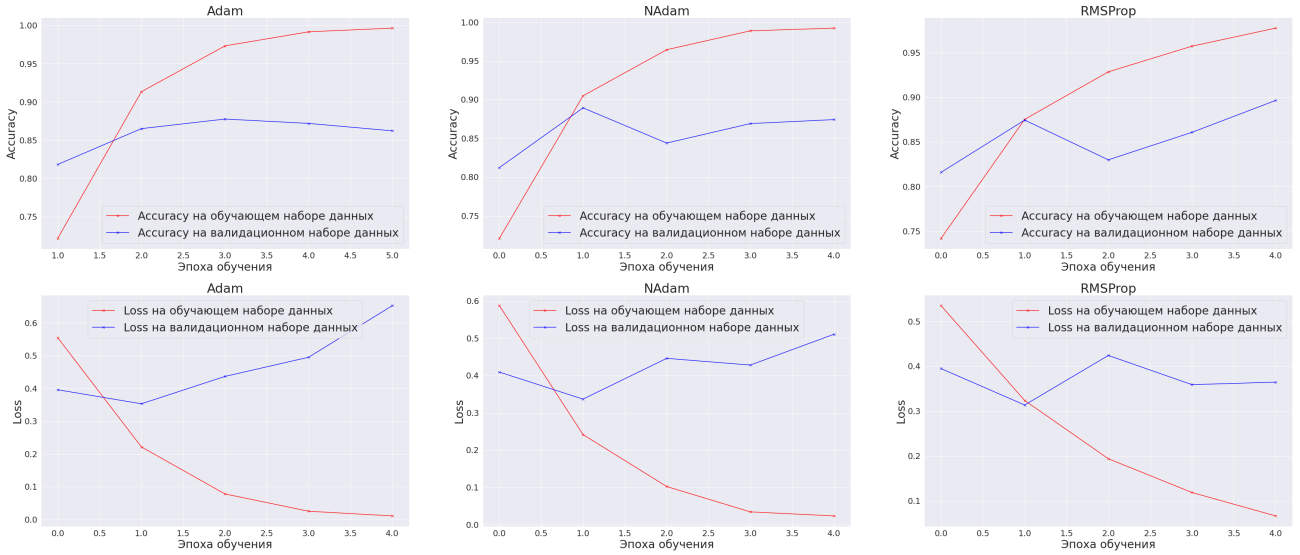


Рис. 2.6. Поведение accuracy и loss в зависимости от выбора оптимизатора

Embedding-слой, как и предполагалось, показал лучший результат по сравнению с предобученными моделями Word2Vec (Рис. 2.7), однако, разница невелика, поэтому можно сделать вывод, что в промышленной разработке следует использовать последний в силу большего по объемам латентного словаря (если нет ограничений по памяти).

Таблица 2.7 Сравнение различных векторных представлений слов

Эмбеддинги	Модели		
	CNN	LSTM	GRU
W2V CBOW	0.8991	0.88	0.9353
W2V Skip-gram	**0.8771**	**0.841**	**0.9574**
Trainable embedding	0.9012	0.9172	0.9562

2.2.4 Приемы мета-моделирования

Полученные обученные модели показывают хорошие результаты, однако их можно дополнительно улучшить с помощью технологий стэкинга. Идея заключается в создании метамодели, которая будет по результатам мета-признаков моделей давать конечное предсказание. Обучение моделей для создания мета-модели проходило по принципу Stratified K-Foldation с сохранением балансировки классов (рис. 2.7).

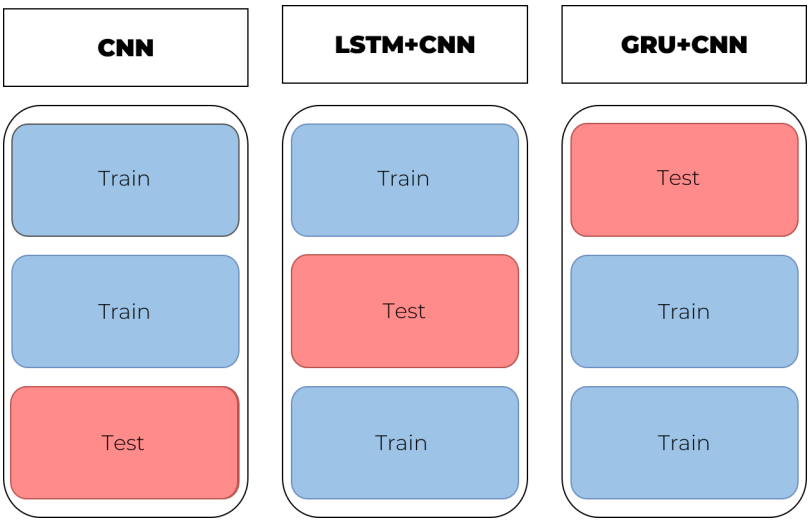


Рис. 2.7. Stratified K-Foldationg для мета моделирования

Тренировочный датасет объемом 3.625 миллионов отзывов (результат присоединения датасетов Amazon и IMDB), был разбит на три части, на двух из которых каждая модель обучалась, затем у нее убирался полносвязный слой с sigmoid функцией активации, чтобы получить признаковое

описание начальных данных сложной структуры в виде табличной структуры, а на третьей части каждая модель давала предсказания без пересечения между собой. Мета-предсказания, будучи табличными данными, были адресованы в классификаторы Random Forest, GBM, SVM, обученные по принципу кросс-валидации (рис. 2.8)

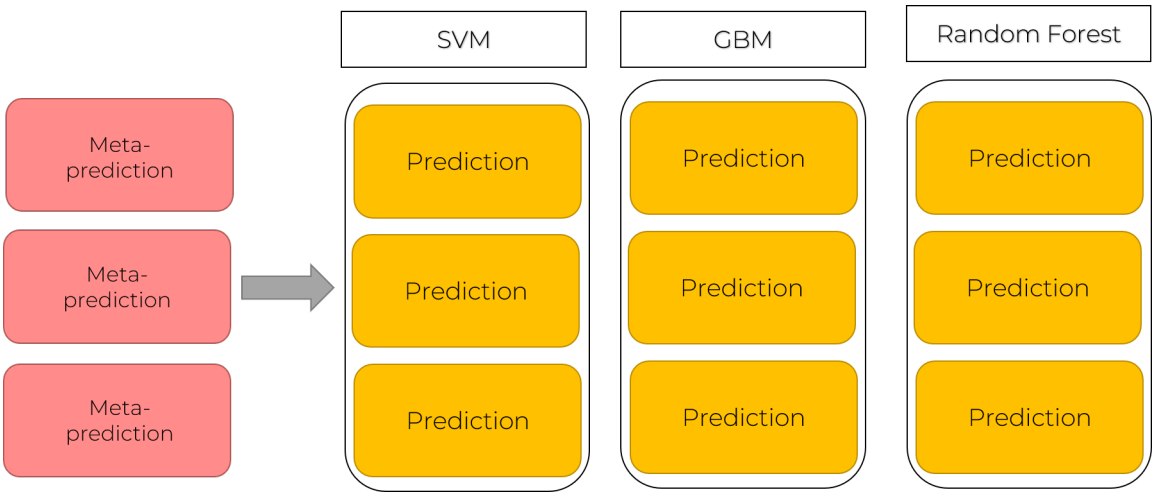


Рис. 2.8. Кроссвалидация ансамблированных моделей

Усредненный результат трех предсказательных моделей является конечным результатом (рис. 2.9):

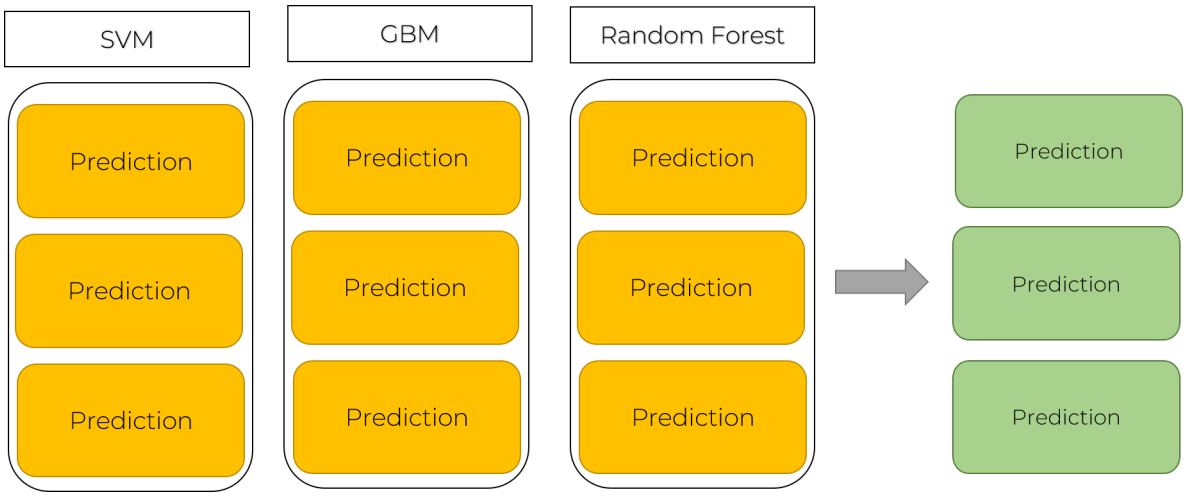


Рис. 2.9. Ансамблирование и голосование

2.2.5 Реализация веб-сервиса

2.2.5.1 Скраппинг

Для реализации веб-сервиса по анализу общественного мнения в видеохостинге YouTube были написаны скрапперы с помощью Selenium и Youtube API (так как время работы прототипа с использованием системы автоматизации Selenium было излишне долгим). В рамках Selenium собирается информация о видео (так как это не требует скроллинга) и асинхронно с этим происходит запрос в YouTube API с целью собрать комментарии, количество лайков и ответов на эти комментарии. Затем наскрапленные данные направляются в обученные модели, логиты с последнего слоя которых подаются в классификаторы машинного обучение, затем результат усредняется. Такой же подход был реализован в статье, представляемой в [1] с реализацией телеграм-бота.

2.2.5.2 Frontend разработка

В качестве шаблона был использован Bootstrap5 и Jinja2 в качестве шаблонизатора. Верстка шаблонов страниц проделана с помощью HTML и CSS. В рамках работы была произведено масштабирование и адаптация под разные экраны пользователей. Можно говорить о кросс-платформенном использовании web-приложения.

2.2.5.3 Backend разработка

В качестве серверной части web-приложения использован фреймвор Flask и база данных NoSQL.

2.2.5.4 Визуализация

Главная страница сервиса с навигацией рис. 2.10, 2.11:

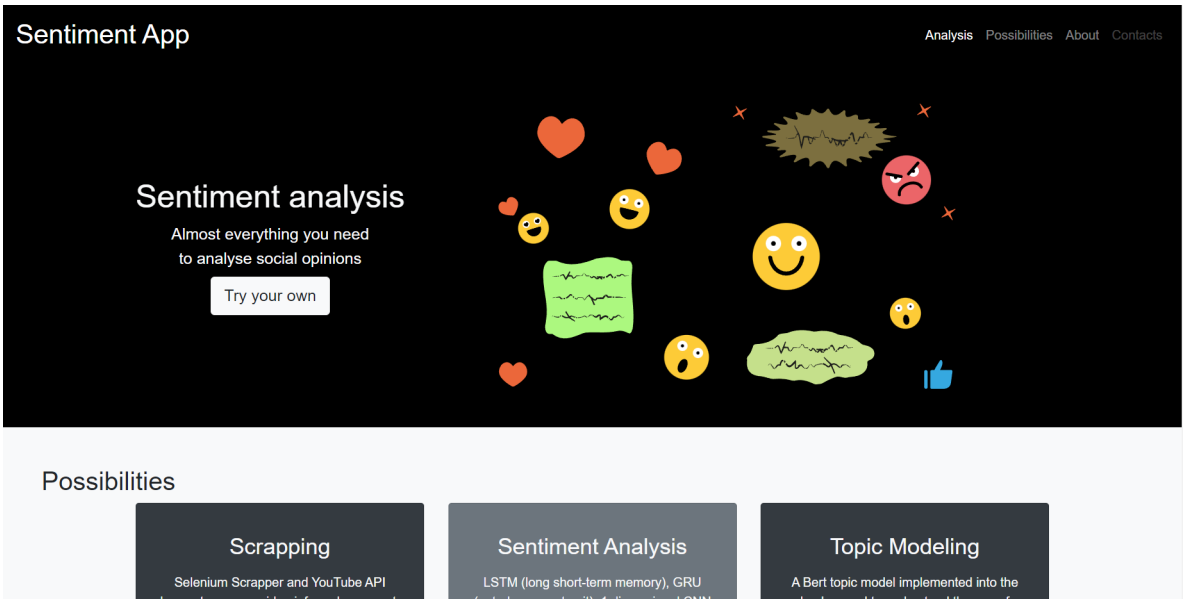


Рис. 2.10. Главная страница сервиса

В поле Link можно вставить ссылку на видео YouTube и запустить таким образом процесс анализа:

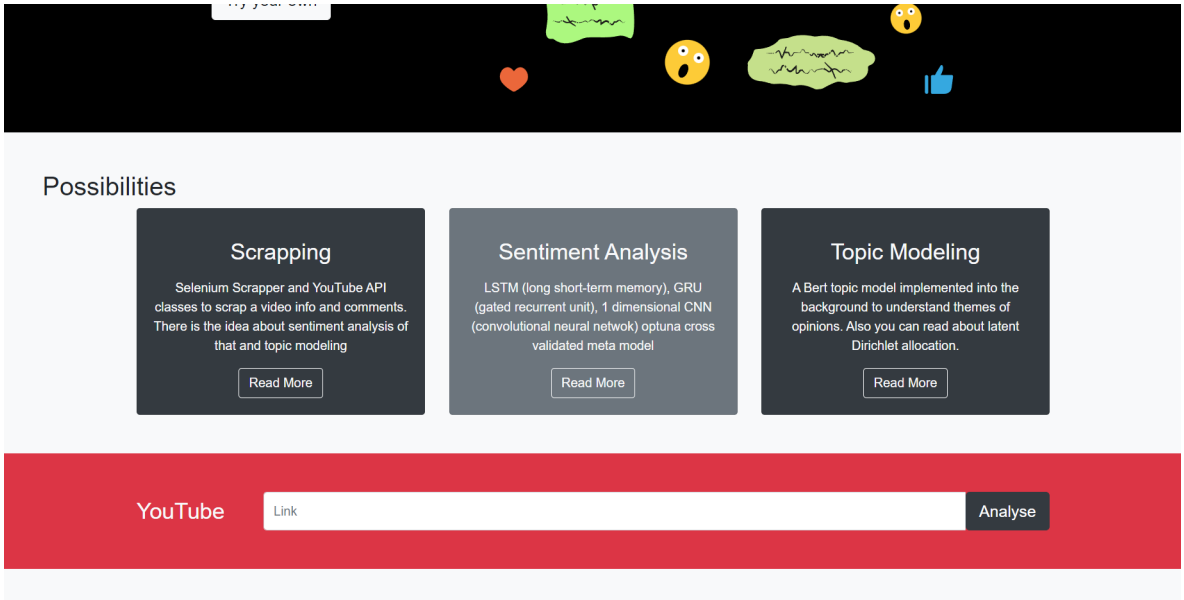


Рис. 2.11. Главная страница сервиса

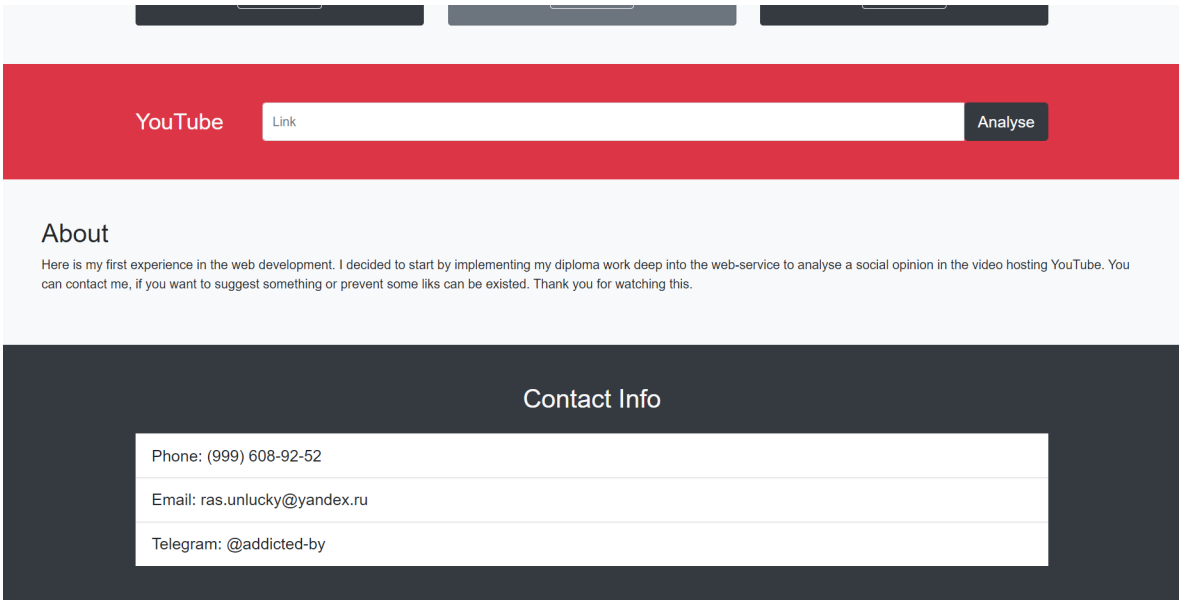


Рис. 2.12. Главная страница. Поле ввода

Как уже было сказано, реализованный сервис является кросплатформенным рис. 2.13:

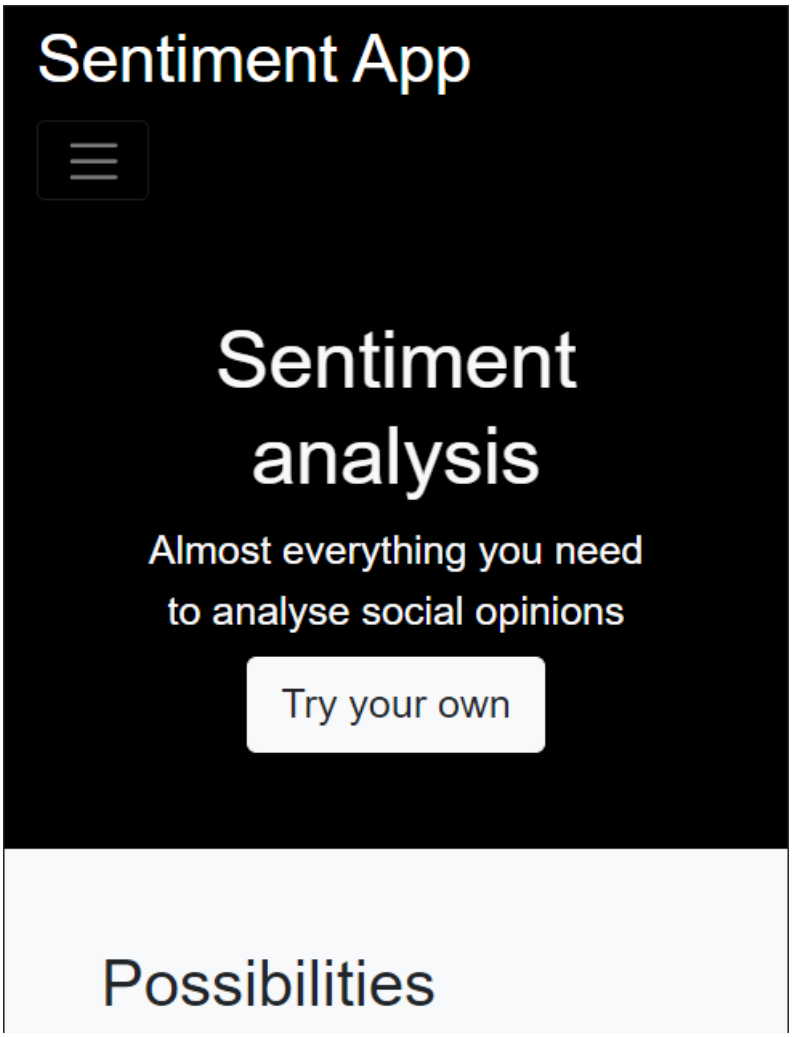


Рис. 2.13. Мобильная версия главной страницы

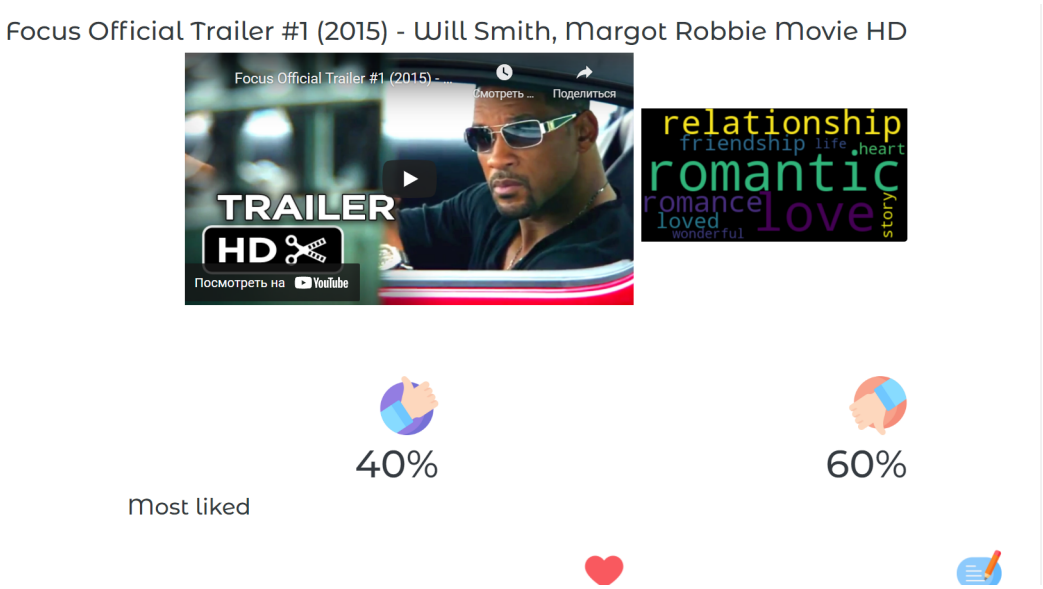


Рис. 2.14. Страница анализа

Кроме того в процессе анализа можно посмотреть на самые интересу-

ющие публику (с наибольшим количеством лайков и ответов) комментарии (рис. 2.15).

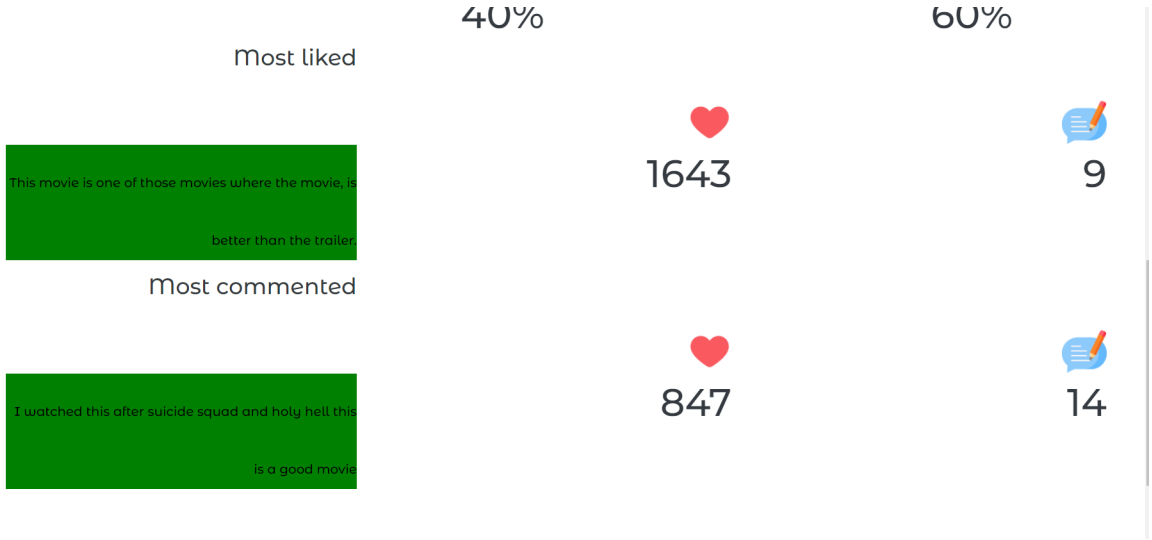


Рис. 2.15. Страница анализа – наиболее популярные комментарии

ЗАКЛЮЧЕНИЕ

ЗАКЛЮЧЕНИЕ

В данной работе были реализованы ансамбль моделей нейросетевого проектирования задачи сентимент-анализа, рассмотрены и использованы модели тематического моделирования (вероятностные и пространственные), а так же показана возможная реализация веб-сервиса, не имеющего аналогов и служащего в качестве завершения дипломной работы. Работа актуальна в силу постоянно растущего объема данных, генерируемых в виде общественных мнений в сети интернет. Задача может быть применена не только к анализу общественных мнений, такой подход применим для рубрикации научных статей, для проведения латентных социальных опросов и как Feature Engineering, например, для задачи предсказания поведения акций на бирже, поскольку общественное мнение и поведение рынка это сильно связанные вещи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рябыкин А. С., Сухов Е. А. Детекция оружия в реальном времени нейросетевыми методами для задач обеспечения безопасности и разведки // XLVIII Международная молодёжная научная конференция «Гагаринские чтения – 2022». — 2022.
2. Barron A. Barron, A.E.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans. on Information Theory 39, 930-945 // Information Theory, IEEE Transactions on. — 1993. — Июнь. — Т. 39. — С. 930—945. — DOI: 10.1109/18.256500.
3. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin [и др.]. — 2018. — DOI: 10.48550/ARXIV.1810.04805. — URL: https://arxiv.org/abs/1810.04805.
4. ConvLSTMConv network: a deep learning approach for sentiment analysis in cloud computing / M. Ghorbani [и др.] // Journal of Cloud Computing Advances Systems and Applications. — 2020. — Март. — Т. 9. — С. 16. — DOI: 10.1186/s13677-020-00162-1.
5. Distributed Representations of Words and Phrases and their Compositionality / T. Mikolov [и др.]. — 2013. — arXiv: 1310.4546 [cs.CL].
6. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / N. Srivastava [и др.] // Journal of Machine Learning Research. — 2014. — Т. 15, № 56. — С. 1929—1958. — URL: http://jmlr.org/papers/v15/srivastava14a.html.
7. Efficient Estimation of Word Representations in Vector Space / T. Mikolov [и др.]. — 2013. — DOI: 10.48550/ARXIV.1301.3781. — URL: https://arxiv.org/abs/1301.3781.
8. Fang X., Zhan J. Sentiment analysis using product review data // J Big Data. — 2015. — Дек. — Т. 2. — DOI: 10.1186/s40537-015-0015-2.
9. Going Deeper with Convolutions / C. Szegedy [и др.]. — 2014. — Сент.

10. Goldberg Y., Levy O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. — 2014. — DOI: 10.48550/ARXIV.1402.3722. — URL: https://arxiv.org/abs/1402.3722.

11. Goodfellow I., Bengio Y., Courville A. Deep learning. — 2017. — С. 283—308.

12. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies / F. Informatik [и др.] // A Field Guide to Dynamical Recurrent Neural Networks. — 2003. — Март.

13. Handwritten Digit Recognition with a Back-Propagation Network / Y. Le Cun [и др.] // Proceedings of the 2nd International Conference on Neural Information Processing Systems. — MIT Press, 1989. — С. 396—404. — (NIPS'89).

14. Harris Z. Distributional structure // Word. — 1954. — Т. 10, № 2/3. — С. 146—162. — DOI: 10.1007/978-94-009-8467-7_1. — URL: https://link.springer.com/chapter/10.1007/978-94-009-8467-7_1.

15. Hirose Y., Yamashita K., Hijiya S. Back-propagation algorithm which varies the number of hidden units // Neural Networks. — 1991. — Т. 4, № 1. — С. 61—66. — ISSN 0893-6080. — DOI: https://doi.org/10.1016/0893-6080(91)90032-Z. — URL: https://www.sciencedirect.com/science/article/pii/089360809190032Z.

16. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Comput. — Cambridge, MA, USA, 1997. — Нояб. — Т. 9, № 8. — С. 1735—1780. — ISSN 0899-7667. — DOI: 10.1162/neco.1997.9.8.1735. — URL: https://doi.org/10.1162/neco.1997.9.8.1735.

17. Hornik K., Stinchcombe M., White H. Multilayer feedforward networks are universal approximators // Neural Networks. — 1989. — Т. 2, № 5. — С. 359—366. — ISSN 0893-6080. — DOI: https://doi.org/10.1016/0893-6080(89)90020-8. — URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

18. Jain A. K., Dubes R. C. Algorithms for clustering data. — Prentice-Hall, Inc., 1988.

19. Jozefowicz R., Zaremba W., Sutskever I. An Empirical Exploration of Recurrent Network Architectures // Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. — Lille, France : JMLR.org, 2015. — С. 2342—2350. — (ICML'15).

20. Lecun Y., Bengio Y. Convolutional Networks for Images, Speech, and Time-Series //. — 01.1995.

21. Lecun Y., Kavukcuoglu K., Farabet C. Convolutional Networks and Applications in Vision //. — 05.2010. — С. 253—256. — DOI: 10.1109/ISCAS.2010.5537907.

22. McInnes L., Healy J., Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. — 2018. — DOI: 10.48550/ARXIV.1802.03426. — URL: https://arxiv.org/abs/1802.03426.

23. Schmidhuber J. Deep learning in neural networks: An overview // Neural Networks. — 2015. — Т. 61. — С. 85—117. — ISSN 0893-6080. — DOI: https://doi.org/10.1016/j.neunet.2014.09.003. — URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

24. Venkat N. The Curse of Dimensionality: Inside Out. — 2018. — Сент. — DOI: 10.13140/RG.2.2.29631.36006.

25. Williams R. J., Zipser D. Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. — 1995.

26. Xing Y., Xiao C. A GRU Model for Aspect Level Sentiment Analysis // Journal of Physics: Conference Series. — 2019. — Авг. — Т. 1302. — С. 032042. — DOI: 10.1088/1742-6596/1302/3/032042.

27. Zhou Y., Chellappa R. Computation of optical flow using a neural network //. — 08.1988. — 71—78 vol.2. — DOI: 10.1109/ICNN.1988.23914.

28. Котельников Е. В., Клековкина М. В. Автоматический анализ тональности текстов на основе методов машинного обучения //. — Компьютерная лингвистика и интеллектуальные технологии. РГГУ, 2012. — С. 27—36.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

```
import random
import numpy as np

import gensim.downloader as api

import torch
import torch.nn as nn
import math

class CNNModel(nn.Module):
    def __init__(self, dict_size, embed_size, hidden_size, num_classes=4):
        super().__init__()
        self.embeddings = nn.Embedding(len(dict_size), embedding_dim=embed_size)
        self.cnn = nn.Sequential(
            nn.Conv1d(embed_size, hidden_size, kernel_size=3, padding=1, stride=2),
            nn.BatchNorm1d(hidden_size),
            nn.ReLU(),
            nn.Conv1d(hidden_size, hidden_size, kernel_size=3, padding=1, stride=2),
            nn.BatchNorm1d(hidden_size),
            nn.ReLU(),
            nn.Conv1d(hidden_size, hidden_size, kernel_size=3, padding=1, stride=2),
            nn.BatchNorm1d(hidden_size),
            nn.ReLU(),
            nn.AdaptiveMaxPool1d(1),
            nn.Flatten(),
        )
```

```
27          self.cl = nn.Sequential(
28              nn.Linear(hidden_size, num_classes)
29          )
30
31      def forward(self, x):
32          x = self.embeddings(x)  # (batch_size, seq_len,
    embed_dim)
33          x = x.permute(0, 2, 1)
34          x = self.cnn(x)
35          prediction = self.cl(x)
36          return prediction
37
38
39  class GRU(nn.Module):
40      def __init__(self, embed_size, hidden_size):
41          super().__init__()
42
43          self.embed_size = embed_size
44          self.hidden_size = hidden_size
45
46          self.w_rh = nn.Parameter(torch.rand(hidden_size,
    hidden_size))
47          self.b_rh = nn.Parameter(torch.rand((1, hidden_size)))
48          self.w_rx = nn.Parameter(torch.rand(embed_size,
    hidden_size))
49          self.b_rx = nn.Parameter(torch.rand(1, hidden_size))
50
51          self.w_zh = nn.Parameter(torch.rand(hidden_size,
    hidden_size))
52          self.b_zh = nn.Parameter(torch.rand((1, hidden_size)))
53          self.w_zx = nn.Parameter(torch.rand(embed_size,
    hidden_size))
54          self.b_zx = nn.Parameter(torch.rand(1, hidden_size))
```

```
55
56          self.w_nh = nn.Parameter(torch.rand(hidden_size,
    hidden_size))
57          self.b_nh = nn.Parameter(torch.rand((1, hidden_size)))
58          self.w_nx = nn.Parameter(torch.rand(embed_size,
    hidden_size))
59          self.b_nx = nn.Parameter(torch.rand(1, hidden_size))
60
61      def forward(self, x, hidden = None):
62          if hidden is None:
63              hidden = torch.zeros((x.size(0), self.hidden_size)).
    to(x.device)
64
65          for cur_idx in range(x.size(1)):
66              r = torch.sigmoid(
67                  x[:, cur_idx] @ self.w_rx + self.b_rx + hidden @
     self.w_rh + self.b_rh
68              )
69              z = torch.sigmoid(
70                  x[:, cur_idx] @ self.w_zx + self.b_zx + hidden @
     self.w_zh + self.b_zh
71              )
72              n = torch.tanh(
73                  x[:, cur_idx] @ self.w_nx + self.b_nx + r * (
    hidden @ self.w_nh + self.b_nh)
74              )
75              hidden = (1 - z) * n + z * hidden
76
77          return hidden
78
79
80 class GRUModel(nn.Module):
```

```
81     def __init__(self, dict_size, embed_size, hidden_size,
   num_classes=2):
82         super().__init__()
83         self.embed = nn.Embedding(len(dict_size), embed_size)
84         self.gru = GRU(embed_size, hidden_size)
85         self.cls = nn.Linear(hidden_size, num_classes)
86
87     def forward(self, x):
88         x = self.embed(x)
89         hidden = self.gru(x)
90         output = self.cls(hidden)
91         return output
92
93
94 class LSTMModel(nn.Module):
95     def __init__(self, input_sz, hidden_sz):
96         super().__init__()
97         self.input_sz = input_sz
98         self.hidden_size = hidden_sz
99         self.W = nn.Parameter(torch.Tensor(input_sz, hidden_sz *
    4))
100        self.U = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz
   * 4))
101        self.bias = nn.Parameter(torch.Tensor(hidden_sz * 4))
102        self.init_weights()
103
104    def init_weights(self):
105        stdv = 1.0 / math.sqrt(self.hidden_size)
106        for weight in self.parameters():
107            weight.data.uniform_(-stdv, stdv)
108
109    def forward(self, x,
110                init_states=None):
```

```
        """Assumes x is of shape (batch, sequence, feature)"""
        bs, seq_sz, _ = x.size()
        hidden_seq = []
        if init_states is None:
            h_t, c_t = (torch.zeros(bs, self.hidden_size).to(x.device),
                        torch.zeros(bs, self.hidden_size).to(x.device))
        else:
            h_t, c_t = init_states

        HS = self.hidden_size
        for t in range(seq_sz):
            x_t = x[:, t, :]
            # batch the computations into a single matrix multiplication
            gates = x_t @ self.W + h_t @ self.U + self.bias
            i_t, f_t, g_t, o_t = (
                torch.sigmoid(gates[:, :HS]), # input
                torch.sigmoid(gates[:, HS:HS*2]), # forget
                torch.tanh(gates[:, HS*2:HS*3]),
                torch.sigmoid(gates[:, HS*3:]), # output
            )
            c_t = f_t * c_t + i_t * g_t
            h_t = o_t * torch.tanh(c_t)
            hidden_seq.append(h_t.unsqueeze(0))
        hidden_seq = torch.cat(hidden_seq, dim=0)
        # reshape from shape (sequence, batch, feature) to (batch, sequence, feature)
        hidden_seq = hidden_seq.transpose(0, 1).contiguous()
        return hidden_seq, (h_t, c_t)
```

```
import os
import warnings
```

```
3 warnings.simplefilter("ignore", DeprecationWarning)
4 warnings.simplefilter("ignore", UserWarning)
5 from selenium import webdriver
6 from selenium.webdriver.common.keys import Keys
7 import time
8 import urllib.parse as urlparse
9 from bs4 import BeautifulSoup
10 import re
11 import pandas as pd
12 from googleapiclient.discovery import build
13 import json
14 from csv import writer
15 from urllib.request import urlopen
16 from urllib.parse import urlencode
17 import os
18 import time
19
20
21 class YouTubeScrapper():
22     # Constructor
23     def __init__(self,
24                      webdriver_path:str='./',
25                      browser={'edge', 'chrome'},
26                      output_path:str='', headless:bool=True):
27         assert browser in {'edge', 'chrome'}, 'browser not
    comparable'
28         self.scrapper_class = 'YouTube scrapper'
29         options = webdriver.edge.options.Options() if browser ==
     'edge' \
30                                                 else webdriver.
    chrome.options.Options()
31         if headless:
32             options.add_argument("--headless")
```

```
33        try:
34            if webdriver_path == ’./’:
35                self.driver = webdriver.Edge(options=options) if
 browser == ’edge’ \
36                                        else webdriver.Chrome(
options=options)
37            else:
38                self.driver = webdriver.Edge(webdriver_path,
39                                     options=options) if browser
==’edge’ \
40                             else webdriver.Chrome(webdriver_path
, options=options)
41        except:
42            print("webdriver can’t be found with that input path
")
43        self.output_path = output_path
44
45    def get_video_id(self, link):
46        query = urlparse.urlparse(link)
47        if query.hostname == ’youtu.be’:
48            return query.path[1:]
49        if query.hostname in (’www.youtube.com’, ’youtube.com’):
50            if query.path == ’/watch’:
51                p = urlparse.parse_qs(query.query)
52                return p[’v’][0]
53            if query.path[:7] == ’/embed/’:
54                return query.path.split(’/’)[2]
55            if query.path[:3] == ’/v/’:
56                return query.path.split(’/’)[2]
57        # fail?
58        return None
59
60    def scroll_to_bottom(self, limit=None, time_sleep:int=1.5):
```

```
61          last_height = self.driver.execute_script('\
62          return window.scrollY')
63          current = 1
64          limit_check = lambda x: True if x > limit else False
65          while True:
66              from selenium.webdriver.common.keys import Keys
67              html = self.driver.find_element_by_tag_name('html')
68              html.send_keys(Keys.END)
69              time.sleep(time_sleep)
70              new_height = self.driver.execute_script('\
71                  return window.scrollY')
72              if last_height == new_height:
73                  break
74              if limit:
75                  if limit_check(current):
76                      break
77                  current += 1
78              last_height = new_height
79
80      def from_string_to_int(self, fol_string: str):
81          tens = {'K': 10e2, 'M': 10e5, 'B': 10e8, 'k': 10e2, 'm':
    10e5, 'b': 10e8}
82          if (fol_string[-1] != 'K' and fol_string[-1] != 'M'
83                          and fol_string[-1] != 'k' and fol_string
    [-1] != 'm'
84                          and fol_string[-1] != 'b' and fol_string
    [-1] != 'B'):
85              return int(fol_string)
86          f = lambda x: int(float(x[:-1])*tens[x[-1]])
87          return f(fol_string)
88      # get likes method
89      def get_likes(self) -> int:
90          fol_string = self.soup.find(id='menu-container').find(id
```

```
=’text’).text
91          return self.from_string_to_int(fol_string=fol_string)
92      #get views
93      def get_views(self) -> int:
94          fol_string = self.soup.find_all(
95              ’div’, id=’info’)[-1].find(
96                  ’div’, id=’count’).find(
97                      class_=re.compile(r’short*’)).text.split()
[0]
98          return self.from_string_to_int(fol_string=fol_string)
99      #scrape method
100     def scrape_comments(self, link: str, save=True,
101                                 scroll_limit=None, time_sleep:int=3)
:
102         self.driver.get(link)
103         time.sleep(time_sleep//2)
104         self.soup = BeautifulSoup(self.driver.page_source, ’html
.parser’)
105         del self.driver
106         video_info = {
107             ’video_id’: self.get_video_id(link=link),
108             ’link’: link,
109             ’likes’: self.get_likes(),
110             ’views’: self.get_views(),
111         }
112         df_video = pd.DataFrame(video_info, index=[’video’])
113         if save:
114             df_video.to_csv(os.path.join(self.output_path, ’
video_info.csv’), index=False)
115         return df_video
116
117
118
```

```
119 class YouTubeAPIscrapper():
120     def __init__(self,
121      key="AIzaSyAND6aT1SjcmWu6abbWVsP6ztfRShkERyM", api_version:
    str = 'v3'):
122         YOUTUBE_API_SERVICE_NAME = "youtube"
123         YOUTUBE_API_VERSION = api_version
124         self.service = build(YOUTUBE_API_SERVICE_NAME,
    YOUTUBE_API_VERSION,developerKey=key)
125
126     def get_comments(self, part='snippet',
127                      maxResults = 100,
128                      textFormat='plainText',
129                      order='time',
130                      videoId='UrqqpUQtFwc',
131                      output_path = "./",
132                      csv_filename="data.csv"):
133
134         #3 create empty lists to store desired information
135         comments, authors,sources, dates, likes, replies = [],
    [], [], [], [], []
136         # build our service from path/to/apikey
137         # service = build_service()
138
139         #4 make an API call using our service
140         response = self.service.commentThreads().list(part=part,
     maxResults=maxResults,
141                                 textFormat=textFormat,
142                                 order=order, videoId=videoId).
    execute()
143         pd.DataFrame(columns=['Sources', 'Date', 'Author name',
    'Comments',
144                                 'Likes', 'Replies']).to_csv(os.
    path.join(output_path, csv_filename), index=False)
```

```
145         while response: # this loop will continue to run until
    you max out your quota
146             for item in response['items']:
147                 #4 index item for desired data features
148                 comment1 = item['snippet']['topLevelComment']['
    snippet']
149                 comment = comment1['textDisplay'].replace('\n',
    '')
150                 author = comment1['authorDisplayName']
151                 date = comment1['publishedAt']
152                 source = comment1['videoId']
153                 likes_ = comment1['likeCount']
154                 replies_ = item['snippet']['totalReplyCount']
155
156
157                 #4 append to lists
158                 comments.append(comment)
159                 authors.append(author)
160                 sources.append(source)
161                 dates.append(date)
162                 likes.append(likes_)
163                 replies.append(replies_)
164
165
166                 #7 write line by line
167                 with open(os.path.join(output_path, csv_filename
    ),'a+',encoding='utf-8-sig') as f:
168                     # write the data in csv file with colums(
    source, date, author, text of comment)
169                     csv_writer = writer(f)
170                     csv_writer.writerow([source,date,author,
    comment, likes_, replies_])
171
```

```
172                 #8 check for nextPageToken, and if it exists,
    set response equal to the JSON response
173             if 'nextPageToken' in response:
174                 response = self.service.commentThreads().list(
175                     part=part,
176                     maxResults=maxResults,
177                     textFormat=textFormat,
178                     order=order,
179                     videoId=videoId,
180                     pageToken=response['nextPageToken']
181                 ).execute()
182             else:
183                 break
184
185
186         #9 return our data of interest
187         return {
188             'Sources' : sources,
189             'Date': dates,
190             'Author name': authors,
191             'Comments': comments,
192             'Likes': likes,
193             'Replies': replies
194         }
```

```
1 from importlib.resources import path
2 import json
3 import pickle
4 import math
5 import os
6
7 from sklearn import preprocessing
8 from models import *
9 from models.models import CNNModel, GRUModel, LSTMModel
```

```
10 from preprocessing import *
11 from pyexpat import model
12 from flask import request, abort, make_response, redirect,
    render_template
13 from flask import Flask
14 from flask import jsonify
15 from json import dumps
16 from scrapping import *
17 from catboost import CatBoostClassifier
18 import torch
19 import os
20 from numpy import np
21 from sklearn.datasets import fetch_20newsgroups
22 from bertopic import BERTopic
23 from wordcloud import WordCloud
24 import matplotlib.pyplot as plt
25
26 def create_wordcloud(topic_model, topic, save:bool = True):
27     plt.figure(figsize=(15,15))
28     text = {word: value for word, value in topic_model.get_topic
    (topic)}
29     wc = WordCloud(background_color="black", max_words=1000)
30     wc.generate_from_frequencies(text)
31     plt.imshow(wc, interpolation="bilinear")
32     plt.axis("off")
33     plt.savefig(os.path.join('./static/', 'wordscloud.png'))
34
35
36
37 app = Flask(__name__)
38
39 @app.before_first_request
40 def load_global_data():
```

```
41     global model
42
43     with open(os.path.join('models/', 'knn_classifier.pkl'), 'rb
    ') as fid:
44         model = pickle.load(fid)
45
46
47 def get_prediction(loader):
48     WEIGHTS = np.array([1, 1.2])
49
50     cnn_logits = CNNModel_(loader).detach().numpy()
51 #     resnet152_logits = resnet152(transformed_image[None,]).
    detach().numpy()
52     gru_logits = GRUModel_(loader).detach().numpy()
53
54     predictions = np.array([0, 0])
55
56     boosting_cnn_probas = boosting.predict_proba(gru_logits)
57     boosting_gru_probas = boosting.predict_proba(cnn_logits)
58
59     forest_cnn_probas = random_forest.predict_proba(gru_logits)
60     forest_gru_probas = random_forest.predict_proba(cnn_logits)
61
62     svm_cnn_probas = svm.predict_proba(gru_logits)
63     svm_gru_probas = svm.predict_proba(cnn_logits)
64
65     predictions = (boosting_cnn_probas + boosting_gru_probas +
66                    forest_cnn_probas + forest_gru_probas +
67                    svm_cnn_probas + svm_gru_probas) / 4
68     predictions = predictions * WEIGHTS
69     predictions /= predictions.sum()
70     return predictions
71
```

```
72
73
74 def bad_request(message, code=400):
75     abort(make_response(jsonify(message=message), code))
76
77 @app.route('/analyse', methods=['GET'])
78 def analyze():
79     MODEL_FOLDER = './models/stacking/'
80     url = request.args.get('link', type=str)
81     boosting = CatBoostClassifier().load_model(os.path.join(
    MODEL_FOLDER, 'boosting.model'))
82     random_forest = pickle.load(open(os.path.join(MODEL_FOLDER,
    'random_forest.pkl'), 'rb'))
83     svm = pickle.load(open(os.path.join(MODEL_FOLDER, 'svm.pkl')
    , 'rb'))
84     preprocessor = Preprocessor(nltk=True)
85     CNNModel_ = CNNModel()
86     CNNModel_.load_state_dict(torch.load(os.path.join(
    MODEL_FOLDER, 'cnn.pt')))
87     CNNModel_.eval();
88
89     LSTMModel_ = LSTMModel()
90     LSTMModel_.load_state_dict(torch.load(os.path.join(
    MODEL_FOLDER, 'lstm.pt')))
91     LSTMModel_.eval();
92
93     GRUModel_ = GRUModel()
94     GRUModel_.load_state_dict(torch.load(os.path.join(
    MODEL_FOLDER, 'gru.pt')))
95     GRUModel_.eval();
96
97     boosting = CatBoostClassifier().load_model(os.path.join(
    MODEL_FOLDER, 'boosting.model'))
```

```
98   random_forest = pickle.load(open(os.path.join(MODEL_FOLDER,
     'random_forest.pkl'), 'rb'))
99   svm = pickle.load(open(os.path.join(MODEL_FOLDER, 'svm.pkl')
     , 'rb'))
100  scrapper = YouTubeScrapper()
101  API_scrapper = YouTubeAPIscrapper()
102  video_info = scrapper.scrape_comments(url)
103  comments = API_scrapper.get_comments(url)
104  loader = preprocessor.preprocess(comments)
105  most_likeble = comments.sort_values(by='likes').iloc[0,0]
106  ml_comment_color = 'red' if get_prediction(most_likeble) ==
     0 else 1
107  ml_comment_sentences = most_likeble.split('.')
108  ml_comment_sentence_colors = [get_prediction(sentence) for
     sentence in ml_comment_sentences]
109  ml_comment_json = json.dumps({
110          'color': ml_comment_color,
111          'sentences': ml_comment_sentences,
112          'sentence_colors': ml_comment_sentence_colors})
113  most_comment = comments.sort_values(by='comments').iloc[0,0]
114
115  mc_comment_color = 'green' if get_prediction(most_commented)
      == 1 else 0
116  mc_comment_sentences = most_comment.split('.')
117  mc_comment_sentence_colors = [get_prediction(sentence) for
     sentence in ml_comment_sentences]
118  mc_comment_json = json.dumps({
119          'color': mc_comment_color,
120          'sentences': mc_comment_sentences,
121          'sentence_colors': mc_comment_sentence_colors})
122  topic_model = BERTopic(verbose=True)
123  topics, probs = topic_model.fit_transform(comments)
124
```

```
125     create_wordcloud(topic_model, topic=1)
126     youtube_title = video_info['label'].values()[0]
127     path_word_clouds = os.path.join('./static/', 'wordscloud.png
    ')
128     probas = get_prediction(loader)
129     predicted_class = [np.argmax(probas_) for probas_ in probas]
130     count_positive = sum(predicted_class)
131     count_negative = len(predicted_class) - count_positive
132     positive_percentage = count_positive // len(predicted_class)
133     negative_percentage = negative_percentage // len(
    predicted_class)
134     ml_value_of_likes = most_likeble.head(1)['likes'].values()
    [0]
135     ml_value_of_comments = most_likeble.head(1)['comments'].
    values()[0]
136     mc_value_of_likes = most_comment.head(1)['likes'].values()
    [0]
137     mc_value_of_comments = most_comment.head(1)['comments'].
    values()[0]
138     return render_template('analysis.html',
139                            youtube_title=youtube_title,
140                            path_word_clouds=path_word_clouds,
141                            positive_percentage=
    positive_percentage,
142                            negative_percentage=
    negative_percentage,
143                            context_str=url,
144
145                            ml_comment_json=ml_comment_json,
146                            mc_comment_json=mc_comment_json,
147
148                            ml_value_of_likes=ml_value_of_likes,
149                            ml_value_of_comments=
```

```
    ml_value_of_comments,
150                                 mc_value_of_likes=mc_value_of_likes,
151                                 mc_value_of_comments=
    mc_value_of_comments
152                                 )
153
154 @app.route('/')
155 def main_page():
156     return render_template('main_page.html')
157
158 if __name__ == '__main__':
159     app.run()
```

ПРИЛОЖЕНИЕ 2

```
1 <!-- MAIN_PAGE -->
2 <!DOCTYPE html>
3 <html>
4     <head>
5         <meta charset="utf8">
6         <link rel="preconnect" href="https://fonts.gstatic.com">
7         <link href="https://fonts.googleapis.com/css2?family=
    Montserrat:wght@300;400&display=swap" rel="stylesheet">
8         <meta name="viewport" content="width=device-width,
    initial_scale=1">
9         <link rel="stylesheet" href="https://cdn.jsdelivr.net/
    npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="
    sha384-Gn5384xqQ1aoWXA+058
    RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="
    anonymous">
10
11        <!-- Latest compiled and minified CSS -->
12        <!-- <link rel="stylesheet" href="css/style.css"> -->
13        <link rel="stylesheet" href="{{ url_for('static',
    filename='style.css')}}">
14
15        <script>
16          function analyse() {
17            let link = document.getElementById("link-holder").
    value;
18            window.location.replace(`/analyse?link="${link}"`);
19          }
20        </script>
21        <nav class="navbar navbar-expand-lg navbar-dark">
22            <div class="container">
23                <a class="navbar-brand" href="#">Sentiment App</
```

```
a>
24          </div>
25      <button class="navbar-toggler" type="button" data-toggle
="collapse" data-target="#navbarNavAltMarkup" aria-controls="
navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle
navigation">
26          <span class="navbar-toggler-icon"></span>
27      </button>
28      <div class="collapse navbar-collapse" id="
navbarNavAltMarkup">
29          <div class="navbar-nav">
30          <a class="nav-item nav-link active" href="#analysis
">Analysis <span class="sr-only">(current)</span></a>
31          <!-- ! подменю сделать -->
32          <!-- ? результаты анализа -->
33          <!-- ? топики -->
34          <!-- ? синтаксические деревья -->
35          <a class="nav-item nav-link" href="#possibilities">
Possibilities</a>
36          <a class="nav-item nav-link" href="#about">About</a>
37          <a class="nav-item nav-link disabled" href="#
contacts">Contacts</a>
38          </div>
39      </div>
40      </nav>
41      <title>Sentiment analysis</title>
42  </head>
43  <body>
44      <!-- Showcase -->
45      <section
46     class="bg text-light p-5 p-lg-0 pt-lg-5 pb-lg-5 text-
center text-sm-start"
47  >
```

```
48      <div class="container">
49        <div class="d-sm-flex align-items-center justify-content
    -between">
50          <div>
51            <h1>Sentiment analysis </span></h1>
52            <p class="lead my-2">
53              Almost everything you need <br>to analyse social
    opinions
54            </p>
55            <button
56              class="btn btn-light btn-lg"
57              data-bs-toggle="modal"
58              data-bs-target="#try"
59              style="position: center;"
60            >
61              Try your own
62            </button>
63          </div>
64          <img
65            class="img-fluid w-50 rounded mx-auto float-right d-
    flex d-sm-inline-block"
66            src="{{ url_for('static', filename='showcase1.svg')
    }}"
67            alt="photo tuta"
68          />
69        </div>
70      </div>
71    </section>
72
73    <!-- Possibilities -->
74
75    <section id="possibilities" class="bg-light p-5">
76        <h2>Possibilities</h2>
```

```
77        <div class="container">
78            <div class="row text-center g-4">
79              <div class="col-md">
80                <div class="card bg-dark text-light">
81                  <div class="card-body text-center">
82                    <div class="h1 mb-3">
83                      <i class="bi bi-laptop"></i>
84                    </div>
85                    <h3 class="card-title mb-3">Scrapping</h3>
86                    <p class="card-text">
87                      Selenium Scrapper and YouTube API classes
    to scrap a video info and comments. There is the idea about
    sentiment analysis of that and topic modeling
88                    </p>
89                    <a href="#" class="btn btn-outline-light">
    Read More</a>
90                  </div>
91                </div>
92              </div>
93              <div class="col-md">
94                <div class="card bg-secondary text-light">
95                  <div class="card-body text-center">
96                    <div class="h1 mb-3">
97                      <i class="bi bi-person-square"></i>
98                    </div>
99                    <h3 class="card-title mb-3">Sentiment
    Analysis</h3>
100                   <p class="card-text">
101                     LSTM (long short-term memory), GRU (gated
    recurrent unit), 1 dimensional CNN (convolutional neural
    netwok) optuna cross validated meta model
102                   </p>
103                   <a href="#https://github.com/addicted-by/NLP
```

```
/blob/main/articles/neural_networks_text_classification.pdf"
class="btn btn-outline-light">Read More</a>
104                 </div>
105               </div>
106             </div>
107             <div class="col-md">
108               <div class="card bg-dark text-light">
109                 <div class="card-body text-center">
110                   <div class="h1 mb-3">
111                     <i class="bi bi-people"></i>
112                   </div>
113                   <h3 class="card-title mb-3">Topic Modeling</
h3>
114                   <p class="card-text">
115                     A Bert topic model implemented into the
background to understand themes of opinions. Also you can
read about latent Dirichlet allocation.
116                   </p>
117                   <a href="#" class="btn btn-outline-light">
Read More</a>
118                 </div>
119               </div>
120             </div>
121           </div>
122         </div>
123   </section>
124
125   <!-- Youtube -->
126
127   <section id="try" class="bg-danger text-light p-5">
128       <div class="container">
129         <div class="d-md-flex justify-content-between align-
items-center">
```

```
130             <h3 class="mb-3 pr-5 mb-md-0">YouTube
131             </h3>
132
133             <div class="input-group news-input">
134               <input type="text" id="link-holder" class="form-
    control" placeholder="Link" />
135               <button class="btn btn-dark btn-lg" type="button"
    onclick="analyse();">Analyse</button>
136             </div>
137           </div>
138         </div>
139     </section>
140
141     <!-- Twitter -->
142
143
144     <section id="about" class="p-5 bg-light">
145       <h2>About</h2>
146         <p>
147           Here is my first experience in the web development. I
    decided to start by implementing my diploma work deep into
    the web-service to analyse a social opinion in the video
    hosting YouTube. You can contact me, if you want to suggest
    something or prevent some liks can be existed. Thank you for
    watching this.
148         </p>
149     </section>
150
151     <!-- Contacts -->
152     <section id='contacts' class="p-5">
153         <div class="container">
154           <div class="row g-4">
155             <div class="col-md">
```

```
156             <h2 class="text-center text-light mb-4">Contact
    Info</h2>
157             <ul class="list-group list-group-flush lead">
158               <li class="list-group-item">
159                 <span class="fw-bold">Phone:</span> (999)
    608-92-52
160               </li>
161               <li class="list-group-item">
162                 <span class="fw-bold">Email:</span> ras.
    unlucky@yandex.ru
163               </li>
164               <li class="list-group-item">
165                 <span class="fw-bold">Telegram:</span>
166                 @addicted-by
167               </li>
168             </ul>
169           </div>
170         </div>
171       </div>
172     </section>
173
174   <!-- Footer -->
175     <footer class="p-5 bg-dark text-white text-center position-
    relative">
176       <div class="container">
177         <p class="lead">Copyright &copy; 2022, Ryabykin Aleksey
    </p>
178
179         <a href="#" class="position-absolute bottom-0 end-0 p
    -5">
180           <i class="bi bi-arrow-up-circle h1"></i>
181         </a>
182       </div>
```

```
183     </footer>
184
185         <script src="https://code.jquery.com/jquery-3.2.1.slim.
    min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/
    Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></
    script>
186         <script src="https://cdn.jsdelivr.net/npm/popper.js@1
    .12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+
    Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
187         <script src="https://cdn.jsdelivr.net/npm/bootstrap@4
    .0.0/dist/js/bootstrap.min.js" integrity="sha384-
    JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76
    PVCmYl" crossorigin="anonymous"></script>
188         <script src="js/darkmode.js"></script>
189     </body>
190 </html>
191 <!-- ANALYSIS -->
192 <!DOCTYPE html>
193 <html>
194     <head>
195         <meta charset="utf8">
196         <link rel="preconnect" href="https://fonts.gstatic.com">
197         <link href="https://fonts.googleapis.com/css2?family=
    Montserrat:wght@300;400&display=swap" rel="stylesheet">
198         <link rel="preconnect" href="https://fonts.googleapis.
    com">
199         <link rel="preconnect" href="https://fonts.gstatic.com"
    crossorigin>
200         <link href="https://fonts.googleapis.com/css2?family=
    Montserrat+Alternates:ital,wght@0,200;0,500;1,300&display=
    swap" rel="stylesheet">
201         <meta name="viewport" content="width=device-width,
```

```
initial_scale=1">
202         <link rel="stylesheet" href="https://cdn.jsdelivr.net/
npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="
sha384-Gn5384xqQ1aoWXA+058
RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="
anonymous">
203
204         <!-- Latest compiled and minified CSS -->
205         <link rel="stylesheet" href="{{ url_for('static',
filename='style.css')}}">
206         <nav class="navbar navbar-expand-lg navbar-dark">
207             <div class="container">
208                 <a class="navbar-brand" href="#">Sentiment App</
a>
209             </div>
210         <button class="navbar-toggler" type="button" data-toggle
="collapse" data-target="#navbarNavAltMarkup" aria-controls="
navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle
navigation">
211             <span class="navbar-toggler-icon"></span>
212         </button>
213         <div class="collapse navbar-collapse" id="
navbarNavAltMarkup">
214             <div class="navbar-nav">
215             <a class="nav-item nav-link active" href="#analysis
">Analysis <span class="sr-only">(current)</span></a>
216             <a class="nav-item nav-link" href="#home">Home</a>
217             </div>
218         </div>
219         </nav>
220         <title>Sentiment analysis</title>
221     </head>
222     <body>
```

```
223
224         <!-- Showcase -->
225         <section
226       class="bg text-light p-5 p-lg-0 pt-lg-5 text-center text-
    sm-start"
227     >
228       <div class="container">
229         <div class="d-sm-flex align-items-center justify-content
    -between">
230           <div>
231             <h1>YouTube analysis </span></h1>
232             <p class="lead my-2">
233               Almost everything you need <br>to analyse social
    opinions
234             </p>
235           </div>
236           <img
237             class="img-fluid w-50 d-none d-sm-block"
238             src="{{ url_for('static', filename='showcase1.svg')
    }}"
239             alt="photo tuta"
240           />
241         </div>
242       </div>
243     </section>
244
245
246     <section class="analysis bg text-dark p-5 pt-lg-5 text-left
    text-sm-start">
247         <div class="container">
248             <h2>{{youtube_title}}</h2>
249             <!-- <iframe class="embed-responsive-item"
250             frameborder="0"
```

```
251         allow="accelerometer; autoplay; encrypted-media;
    gyroscope; picture-in-picture"
252         allowfullscreen
253         src="{{context_str}}">
254         </iframe> -->
255
256         <iframe width="560" height="315" src="https://www.
    youtube.com/embed/MxCRgtdAuBo" title="YouTube video player"
    frameborder="0" allow="accelerometer; autoplay; clipboard-
    write; encrypted-media; gyroscope; picture-in-picture"
    allowfullscreen></iframe>
257         <img class="img-fluid w-50 rounded mx-auto float-
    right d-flex d-sm-inline-block" src="{{path_word_clouds}}"
    alt="Word cloud">
258         <div class="row align-items-start">
259           <div class="col-sm"><img
260             class="img-fluid w-25 rounded mx-auto float-
    right d-flex d-sm-inline-block"
261             src="{{ url_for('static', filename='positive-
    vote.png')}}"
262             alt="Positive Icon"
263           />
264           </div>
265           <div class="col-sm"><img
266             class="img-fluid w-25 rounded mx-auto float-
    right d-flex d-sm-inline-block"
267             src="{{ url_for('static', filename='negative-
    vote.png')}}"
268             alt="Negative Icon"
269           />
270           </div>
271         </div>
272         <div class="row align-items-center">
```

```
273                 <div class="col-sm">
274                     {{positive_percentage}}
275                 </div>
276                 <div class="col-sm">
277                     {{negative_percentage}}
278                 </div>
279             </div>
280             <div class="row align-items-center">
281                 <div class="col-sm">
282                     {{positive_count}}
283                 </div>
284                 <div class="col-sm">
285                     {{negative_count}}
286                 </div>
287             </div>
288         </div>
289
290         <div class="row align-items start">
291             <div class="col-sm">
292                 <h3>Most liked</h3>
293             </div>
294             <div class="col-sm"><img
295               class="img-fluid w-25 rounded mx-auto float-
    right d-flex d-sm-inline-block"
296               src="{{ url_for('static', filename='heart.png')
    }}"
297               alt="Heart"
298             /></div>
299             <div class="col-sm"><img
300               class="img-fluid w-25 rounded mx-auto float-
    right d-flex d-sm-inline-block"
301               src="{{ url_for('static', filename='comment.png
    ')}}"
```

```
302                 alt="Comment"
303             /></div>
304         </div>
305         <div class="row align-items center">
306             <div class="col-sm">
307                 {{ml_comment_formated}}
308             </div>
309             <div class="col-sm">
310                 {{ml_value_of_likes}}
311             </div>
312             <div class="col-sm">
313                 {{ml_value_of_comments}}
314             </div>
315         </div>
316         <div class="row align-items center">
317             <div class="col-sm">
318                 <h3>Most commented</h3>
319             </div>
320             <div class="col-sm"><img
321                 class="img-fluid w-25 rounded mx-auto float-
    right d-flex d-sm-inline-block"
322                 src="{{ url_for('static', filename='heart.png')
    }}"
323                 alt="Heart"
324             /></div>
325             <div class="col-sm"><img
326                 class="img-fluid w-25 rounded mx-auto float-
    right d-flex d-sm-inline-block"
327                 src="{{ url_for('static', filename='comment.png
    ')}}"
328                 alt="Heart"
329             /></div>
330         </div>
```

```
331         <div class="row align-items end">
332             <div class="col-sm">
333                 {{mc_comment_formated}}
334             </div>
335             <div class="col-sm">
336                 {{mc_value_of_likes}}
337             </div>
338             <div class="col-sm">
339                 {{mc_value_of_comments}}
340             </div>
341         </div>
342
343
344   </section>
345
346 <!-- Footer -->
347   <footer class="p-5 bg-dark text-white text-center position-
    relative">
348     <div class="container">
349       <p class="lead">Copyright &copy; 2022, Ryabykin Aleksey
    </p>
350
351       <a href="#" class="position-absolute bottom-0 end-0 p
    -5">
352         <i class="bi bi-arrow-up-circle h1"></i>
353       </a>
354     </div>
355   </footer>
356
357       <script src="https://code.jquery.com/jquery-3.2.1.slim.
    min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/
    Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></
    script>
```

```
358         <script src="https://cdn.jsdelivr.net/npm/popper.js@1
    .12.9/dist/umd/popper.min.js" integrity="sha384-ApNbgh9B+
    Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
359         <script src="https://cdn.jsdelivr.net/npm/bootstrap@4
    .0.0/dist/js/bootstrap.min.js" integrity="sha384-
    JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76
    PVCmYl" crossorigin="anonymous"></script>
360         <script src="js/darkmode.js"></script>
361     </body>
362
363 </html>
364
365 <p style='background:green;'>
366     <font color='black' style='padding-bottom:1%; border-bottom
    :4px solid green'>1</font>.
367     <font color='black' style='padding-bottom:1%; border-bottom
    :4px solid red'> But the script is awful.</font>.
368 </p>
369
370 <script>
371     function generateCommentHtml(comment_json) {
372         let comment = JSON.parse(comment_json);
373         let color = comment.color
374         let sentences = comment.sentences
375         let sentenceColors = comment.sentenceColors
376
377         result_html = `<span style='background-color: ${color
    }'>`
378
379         for (let i = 0; i < sentences.length; ++i) {
380             result_html +=
381                 `<span style='box-sizing: border-box; border-
```

```
    bottom: 4px solid ${sentenceColors[i]}'>
                        ${sentences[i]} + ' '
                    </span>`
        }

        return result_html + "</span>"
    }

    function addComment(id, html) {
        document.getElementById(id).innerHTML = html;
    }

    function addMostLiked(html) {
        addComment("most-liked", html)
    }

    function addMostCommented(html) {
        addComment("most-commented", html)
    }

</script>
```