Для работы web-приложения необходима база данных (БД). Под базой данных принято понимать, имеющий свою четкую структуру и порядок набор данных. Под системой управления базами данных (СУБД) понимается программа, которая требуется для того, чтобы организовать и вести БД [4]. Надежность, скорость, расширяемость и открытость исходного кода определили выбор СУБД MySQL.

Инструментом для разработки сайта является язык программирования JavaScript на платформе Node.js. Шаблон Reactor в Node.js обеспечивает обработку операций ввода/вывода, блокируя выполнение до момента доступности новых событий из набора наблюдаемых ресурсов с последующей обработкой каждого события вызовом связанного с ним обработчика.

Для разработки под Node JS достаточно иметь простейший текстовый редактор. При этом для более удобного процесса разработки больше подходит среда разработки, одной из которых является PHPStorm.

Таким образом, разработанный программный комплекс, включает три основных компонента: клиентское приложение, серверное приложение и чат-бот. Взаимодействие компонентов обеспечивает корректную работу веб-приложения. Хранилищем необходимой информации является база данных.

Совершенствование процессов учета материальных ценностей возможно с применением информационно-технических средств, в том числе чат-ботов. Чат-боты обеспечивают оперативность доступа пользователей к данным за счет возможности использования мобильных средств коммуникации и наличия устройств ввода (камера) и вывода (экран).

Список литературы

1. Генератор QR-кода [Электронный ресурс]. URL: https://trustthisproduct.com/free_qr_code_generator.php?lang=ru/ (Дата обращения: 18.03.2019)

2. Людоговский, А. Моделирование бизнес-процессов [Электронный ресурс] / А. Людоговский // Инф. портал «Script coding». – Режим доступа: http://www.script-coding.com/bp.htm l (Дата обращения: 01.01.2021).

3. Складской учет и торговля. Руководство пользователя [Электронный ресурс] – режим доступа: http://www.softstudio.ru/astock/ aStock.pdf (Дата обращения: 10.01.2021).

4. Голицына О. Л. Базы данных: учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – М.: ФОРУМ: ИНФРА-М, 2016. – 400 с.

5. Дронов, В. А. PHP, MySQL, HTML5 и CSS3. Разработка современных динамических Web-сайтов / В. А. Дронов. – СПб.: BHV, 2016. – 688 с.

6. Документация Telegram: Примеры ботов [Электронный ресурс]. URL: https://tlgrm.ru/docs/bots/samples#php/
(Дата обращения: 18.03.2019)

СОЗДАНИЕ КЛАССИФИКАТОРА НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

Митина Ольга Алексеевна
кандидат пед. наук.,
МИРЭА – Российский технологический университет,
г. Москва
Марков Александр Борисович
студент 4 курса,
МИРЭА – Российский технологический университет,
г. Москва

CREATING A CLASSIFIER BASED ON NEURAL NETWORKS

Mitina Olga Alekseevna
Candidate of Science
MIREA – Russian Technological University
Moscow
Markov Alexander Borisovich
4nd year student
MIREA – Russian Technological University
Moscow

АННОТАЦИЯ

В современном мире человечество производит колоссальное количество текстовой информации, и людям становится все сложнее проанализировать и отфильтровать всю эту информацию вручную. С такой проблемой сталкиваются пользователи сайта единой информационной системы в сфере закупок.

Пользователям единой информационной среды в сфере закупок приходится вручную просматривать весь список закупок для поиска необходимых. Несмотря на то, что на сайте предусмотрено множество различных фильтров, возможность выбрать конкретную сферу услуг отсутствует. Автоматическая классификация текста может решить данную проблему.

ANNOTATION

In the modern world, humanity produces a huge amount of textual information, and it is becoming increasingly difficult for people to analyze and filter all this information manually. This problem is faced by users of the site of the unified information system in the field of procurement.

Users of the unified information environment in the field of procurement have to manually view the entire list of purchases to find the necessary ones. Despite the fact that the site provides many different filters, the ability to choose a specific service sector is not available. Automatic text classification can solve this problem.

Ключевые слова: задача классификации текста; нейронные сети.

Keywords: classification task; neural networks.

Классификация – один из разделов машинного обучения с учителем, посвящённый решению следующей задачи [1]. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классификация текстов (документов) (англ. Document classification) – задача компьютерной лингвистики, заключающаяся в отнесении документа к одной из нескольких категорий на основании содержания документа [5].

С задачей классификации текста хорошо справляются методы на основе искусственных нейронных сетей.

Искусственная нейронная сеть (ИНС) – математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. На рисунке 1 представлен пример простой ИНС.

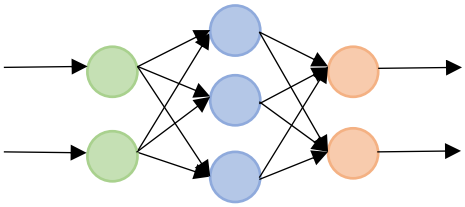

Рисунок 9 Простая искусственная нейронная сеть

На входной слой, имеющий два нейрона, подаются сигналы x_1 и x_2 – входные параметры нейронной сети. Далее сигналы распространяются по всем слоям сети, и на выходе выходного слоя получается результат работы сети – параметры y_1 и y_2.

В зависимости от состава слоев, порядка их следования и способа вычисления выходного сигнала выделяются различные архитектуры ИНС. Рассмотрим основные архитектуры, которые применяются для классификации текста: одномерную сверточную сеть, сеть LSTM и сеть GRU.

Одномерная сверточная ИНС – одна из архитектур глубоких нейронных сетей, применяемая для анализа последовательностей, например, текста. Основной операцией, применяемой в сверточных нейронных сетях, является операция свертки, извлекающая одномерные шаблоны из последовательностей.

На рисунке 2 представлен принцип действия одномерной сверточной нейронной сети.

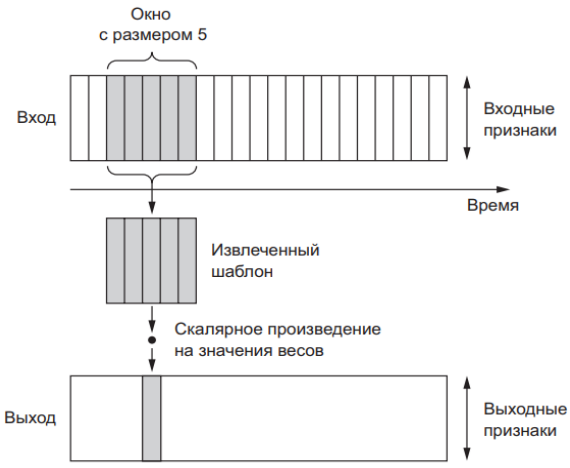

Рисунок 2 Принцип действия одномерной сверточной ИНС

Одномерные сверточные слои позволяют распознавать локальные шаблоны в последовательности. Так как к каждому шаблону применяются одни и те же преобразования, тот или иной шаблон, найденный в некоторой позиции в предложении, позднее может быть выявлен в другой позиции, что делает преобразования, выполняемые одномерными сверточными сетями, инвариантными во времени [2].

Другая архитектура, применяемая для классификации текста – сеть LSTM (Long short-term memory). Она является частным случаем рекуррентной нейронной сети.

Рекуррентные нейронные сети (РНС, англ. Recurrent neural network; RNN) – вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины [3].

Простая рекуррентная сеть не способна обрабатывать длинные последовательности из-за проблемы затухания градиента, которая возникает при обучении сети с большим количеством слоев. Для решения этой проблемы была создана архитектура LSTM.

В основе слоя LSTM лежит алгоритм долгой краткосрочной памяти (Long Short-Term Memory, LSTM), разработанный Хохрейтером и Шмидхубером в 1997 году. Этот слой основан на простом рекуррентном слое с некоторыми модернизациями (рисунок 3). Он сохраняет информацию для последующего использования, тем самым предотвращая постепенное затухание старых сигналов во время обработки [4].

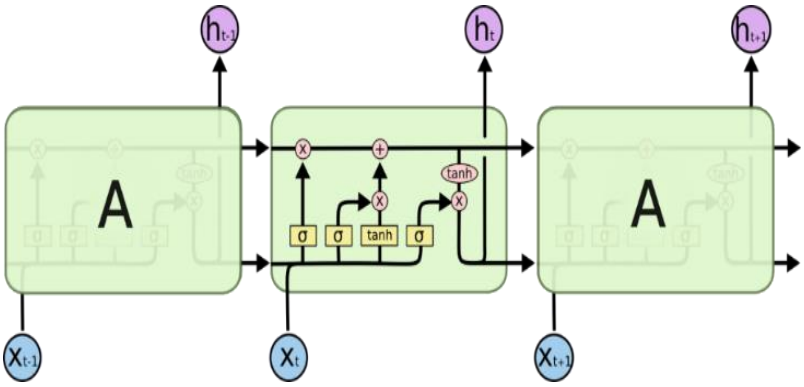

Рисунок 3 Архитектура сети LSTM

Несмотря на то, что архитектура LSTM гораздо лучше справляется с длинными последовательностями, чем простая рекуррентная сеть, она сложнее в обучении и требует больше весов. Поэтому существует другой вариант рекуррентной сети, которая получила название GRU.

Архитектура GRU, как и архитектура LSTM, решает проблему длинных зависимостей, но обладает меньшим количеством весов и легче обучается. Но преимуществом LSTM является то, что она может эффективнее обрабатывает более длинные последовательности, на которых GRU обучается хуже. На рисунке 4 показана архитектура GRU.

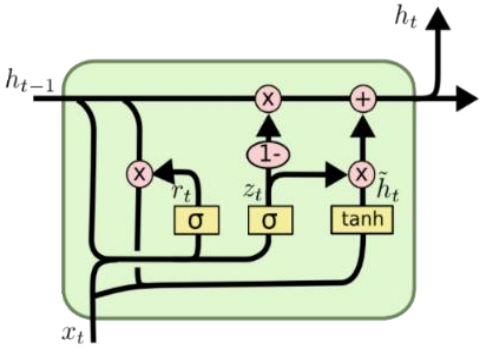

Рисунок 4 Архитектура GRU

Итак, классификация текстовых последовательной является распространённой задачей машинного обучения, для решения которой было создано множество архитектур. Рассмотрим пример классификации текста на наборе данных госзакупок.

Для обучения и тестирования модели используется набор данных, состоящий из 107702

примеров. Каждый пример представляет собой пару значений: номер класса и текст закупки.

Набор содержит 32 категории закупок: строительство и ремонт; медицинские и оздоровительные услуги; охрана, сигнализация, противопожарное оборудование; связь, информационные технологии и другие.

На рисунке 5 показано, сколько примеров тендеров в процентах приходится на каждую категорию. На 6-ю категорию приходится 18987 примеров (17.63%) от всего набора данных, тогда как на 30-ю категорию приходится лишь 114 примеров (0.11%).

Средняя длина описания тендера составляет 72 символа, минимальная длина – 2 символа, максимальная длина – 3633 символа.

Для тренировки классификатора необходимо разбить набор данных на две части: набор для обучения и набор для тестирования.

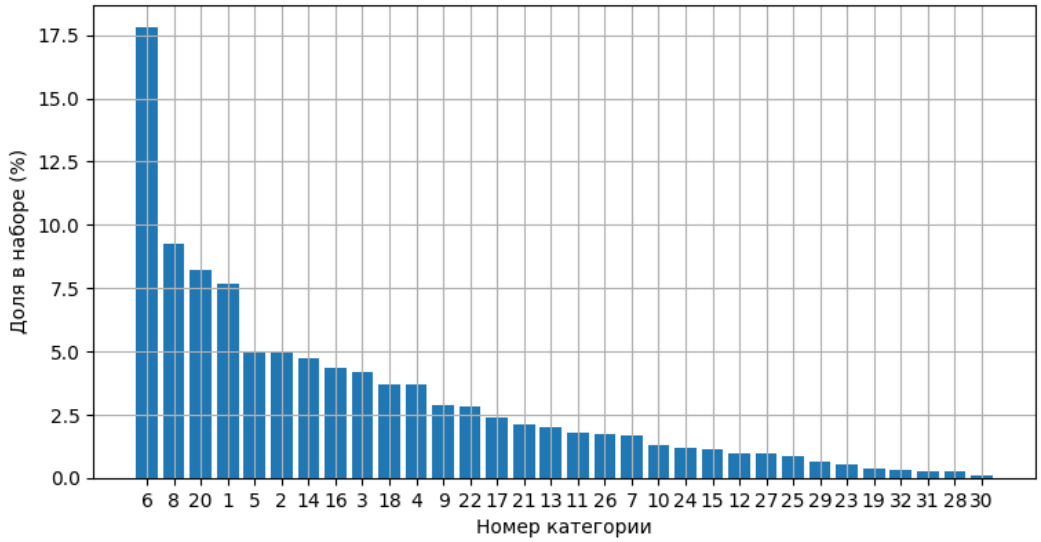

Рисунок 5 Распределение примеров в наборе данных на каждый класс

Набор для обучения используется для обучения модели, тогда как набор для тестирования используется для проверки точности работы модели на новых данных, то есть на данных, на которых она не обучалась.

Для подготовки данных и обучения модели воспользуемся библиотеками глубокого машинного обучения keras и tensorflow.

Преобразуем категории, состоящие из десятичных чисел – номеров категорий, в унитарный код (one hot encoding). Таким образом, класс с номером 2 преобразуется в унитарный код (0, 1, 0, 0, …. 0, 0), количество элементов которого равно числу категорий, то есть 32.

Далее преобразуем описание тендеров в числовое представление. Для этого воспользуемся классом Tokenizer из библиотеки tensorflow. Этот класс позволяет векторизовать корпус текста, превращая каждый текст либо в последовательность целых чисел на основе частоты встречаемости слов в тексте.

Работа с токенизатором состоит из двух этапов. Первый этап заключается в обучении самого токенизатора на словах из обучающего набора. Во время обучения токенизатор составляет словарь частоты встречаемости слов в наборе данных, затем сопоставляет каждому слову индекс от 1 до n, где n – количество слов в словаре, все остальные слов отбрасываются.

В результате обучения токенизатора, самым часто встречающимся словом является слово «поставка». Количество слов в токенизаторе было выбрано равным 10000. Результат работы токенизатора представлен на рисунке 6.

Вторым этапом работы с токенизатором является преобразование исходных текстов в числовой формат. Каждое слово текста заменяется в соответствии со словарем токенизатора на соответствующее число. Неизвестные слова заменяются на специально зарезервированные числа. В результате работы токенизатора текст «оказание услуг по техническому обслуживанию, ремонту систем автоматической пожарной сигнализации (апс) и систем оповещения и управления эвакуацией (соуэ)» преобразуется в последовательность чисел:

[6, 2, 21, 19, 16, 27, 211, 61, 74, 718, 3, 27, 242, 3, 149, 588].

```
{"'поставка": 1,
 'для': 2,
 'по': 3,
 'и': 4,
 "'": 5,
 'услуг': 6,
 "'оказание": 7,
 'в': 8,
 'на': 9,
 'работ': 10,
 "'выполнение": 11,
 'с': 12,
 'медицинского': 13,
 "'закупка": 14,
 'ремонту': 15,
 'лекарственного': 16,
 'препарата': 17,
 'материалов': 18,
 'к': 19,
```

Рисунок 6 Результат работы токенизатора

Последним шагом в подготовке данных является приведение обучающих примеров (текстов) к одной длине. Слишком длинные тексты обрезаются с конца, слишком короткие дополняются с начала нулями. Установим максимальную длину примеров равную 30 словам. На рисунке 7 показано преобразование 3го примера в вид, пригодный для тренировки классификатора.

Последним шагом в подготовке данных является приведение обучающих примеров (текстов) к одной длине. Слишком длинные тексты обрезаются с конца, слишком короткие дополняются с начала нулями. Установим максимальную длину примеров равную 30 словам. На рисунке 7 показано преобразование 3го примера в вид, пригодный для тренировки классификатора.

```
[263] purchases[2]

      'ремонт и промывка теплообменника тепловой сети цтп-20'

[264] x_train[2]

  array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,   20,    3, 5629,
         5630,  471,   71, 2002,  279], dtype=int32)
```

Рисунок 7 Преобразование примера в вид пригодный для тренировки

Обучим классификаторы, основанные на трёх различных архитектурах: одномерной свёрточной сети, сети LSTM и сети GRU.

На рисунке 8 показан состав одномерной сверточной нейронной сети.

```
model_cnn = Sequential()
model_cnn.add(Embedding(TOKENIZER_POWER, 32, input_length=SAMPLE_MAX_LEN))
model_cnn.add(Conv1D(250, 5, padding='valid', activation='relu'))
model_cnn.add(GlobalMaxPooling1D())
model_cnn.add(Dense(128, activation='relu'))
model_cnn.add(Dense(CLASS_COUNT, activation='softmax'))
model_cnn.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

Рисунок 8 Состав одномерной сверточной нейронной сети

В результате обучения сети в течении 10 эпох, модель обучалась до 5 эпохи, после чего начала переобучаться: доля правильных ответов на обучающем наборе продолжала расти, а на проверочном – падать. Модель, обученная на 5 эпохе, показала результат точности классификации на проверочном наборе 79%.

На рисунке 9 показан состав сети LSTM.

```
model_lstm = Sequential()
model_lstm.add(Embedding(TOKENIZER_POWER, 32, input_length=SAMPLE_MAX_LEN))
model_lstm.add(LSTM(32))
model_lstm.add(Dense(CLASS_COUNT, activation='softmax'))
model_lstm.compile(optimizer='adam',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
```

Рисунок 9 Состав сети LSTM

Модель на основе архитектуры LSTM показала результат правильных ответов 77% на проверочном наборе данных.

Последняя рассматриваемая архитектура для обучения классификатора – архитектура на основе ячеек GRU. Сеть на основе ячеек GRU состоит из аналогичных слоев, используемых в сети LSTM, рассмотренной ранее. Отличие заключается в замене ячеек LSTM на ячейки GRU (рисунок 10).

```
model_gru = Sequential()
model_gru.add(Embedding(TOKENIZER_POWER, 32, input_length=SAMPLE_MAX_LEN))
model_gru.add(GRU(32))
model_gru.add(Dense(CLASS_COUNT, activation='softmax'))
model_gru.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

Рисунок 10 Параметры сети GRU

Классификатор на основе GRU показал результат правильных ответов также 77%.

В результате обучения были получены три классификатора, основанные на разных архитектурах ИНС. Протестируем работу моделей на тестовом наборе данных, который содержит примеры, неиспользуемые при обучении моделей.

Тестовый набор данных состоит из 10771 примера. Точность классификации моделей на тестовом наборе представлена в таблице 1.

Таблица 2

Результаты проверки классификаторов на тестовом наборе данных

Название модели	Точность, %
Сверточная	78,29
LSTM	76,57
GRU	77,08

Таким образом, получены 3 классификатора, основанные на различных архитектурах ИНС. Самым точным оказался классификатор на основе одномерной сверточной ИНС. Он показал точность 78,29% на тестовом наборе данных.

На этапе подготовки данных к обучению была намеренно пропущена отчистка данных от лишних символов, коротких слов, и не было произведено приведение слов к начальной форме.

После обучения моделей на очищенном наборе данных точность работы классификаторов возросла на 3-4% (таблица 2).

Таблица 3

Сравнение результатов работы моделей до и после отчистки текста

Название модели	Точность, %	
	До отчистки	После отчистки
Сверточная	78,29	81.26
LSTM	76,57	79.93
GRU	77,08	79.31

На рисунке 5 видно, что на одни классы в обучающем наборе приходится по несколько тысяч примеров для обучения, тогда как на другие – лишь пару сотен. Это не может негативно не сказаться на качестве обучения моделей.

В таблице 3 показано, как зависит точность работы моделей от числа примеров в обучающем наборе данных, при условии, что на каждый класс в обучающем наборе приходится равное число примеров.

Таблица 4

Точность работы моделей в зависимости от числа примеров на каждый класс

Название модели	Точность, %								
	Число примеров на класс								
	1000	2000	3000	4000	5000	6000	7000	8000	9000
Сверточная	75	81	82.5	86.9	90	92.9	93.6	92.6	97.8
LSTM	71	78	80.3	83.3	88.8	93.5	93.1	92.8	95.8
GRU	68	78	81.8	84.1	89.1	92.8	92.8	93.3	97.4

Таким образом, на данный момент точность классификации моделей составляет около 80%. Для достижения точности порядка 90% и более необходимо собрать по 5000-6000 примеров на каждый класс или более.

Благодаря внедрению классификатора на сайт госзакупок, пользователи сайта смогут меньше времени тратить на поиск тендеров, им придётся меньше отфильтровывать лишнюю информацию, а, значит, они останутся довольны.

Список литературы

1. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А.А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.

2. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер, 2018. – 400 с.

3. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. – СПб.: Питер, 2018. – 480 с.

4. Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. – СПб.: Питер, 2019. – 368 с

5. http://karpathy.github.io [Электронный ресурс]. – Режим доступа: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ Дата доступа: 30.04.2021.

ЭФФЕКТИВНОЕ ИСПОЛЬЗОВАНИЕ ПРОИЗВОДСТВЕННОЙ МОЩНОСТИ НА ПРИМЕРЕ ЦЕХА ХИМВОДООЧИТСКИ ТЭЦ

Спиричев Владислав Дмитриевич
Бакалавр
Северный (Арктический) федеральный университет имени М.В. Ломоносова
г. Архангельск
Петухов Сергей Васильевич
кандидат технических наук, доцент
Северный (Арктический) федеральный университет имени М.В. Ломоносова
г. Архангельск

EFFICIENT USE OF PRODUCTION CAPACITY ON THE EXAMPLE OF A CHEMICAL WATER TREATMENT PLANT

Spirichev Vladislav Dmitrievich
Bachelor
Northern (Arctic) Federal University named after M.V. Lomonosov
Arkhangelsk
Petukhov Sergey Vasilievich
candidate of technical sciences, associate professor
Northern (Arctic) Federal University named after M.V. Lomonosov
Arkhangelsk

АННОТАЦИЯ

Эффективность производства - очень важный показатель, отвечающий за производительность. Одним из методов оценивания эффективности производства является коэффициент использования активной мощности оборудования. Чем ближе этот показатель к 100%, тем больше используются производственные мощности, тем меньше разница между фактическим объемом производства и потенциальным объемом производства, и тем выше качество экономического развития. В данной статье проводится анализ эффективности использования производственной мощности цеха химводоочистки (ХВО) ТЭЦ;

ABSTRACT

Production efficiency is a very important metric for productivity. One of the methods for evaluating production efficiency is the active power utilization ratio of equipment. The closer this figure is to 100%, the more production capacity is used, the smaller the difference between the actual volume of production and the potential