Р. Р. Асадулин, Н. Р. Гиясов, А. В. Мельникова
Тюменский государственный университет, г. Тюмень
УДК 004.912

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОБРАБОТКИ ОТЧЕТОВ ПРОЕКТОВ ОБУЧАЮЩИХСЯ

Аннотация. В статье представлены методы анализа студенческих отчетов: извлечение задач, выполненных студентами, именованных сущностей — Ф.И.О. лиц, выполнивших работу, стека использованных технологий для каждой из задач.

Ключевые слова: обработка естественного языка, машинное обучение, векторизация, классификация.

Введение. Каждый семестр участники образовательного процесса сталкиваются с тем, что им нужно найти студента, обладающего определенными навыками и владеющего необходимыми технологиями. С этой задачей могут столкнуться как преподаватели при поиске студентов для выполнения некоторых задач, так и студенты при формировании собственной команды, нахождении кураторов среди студентов старших курсов.

Существует множество работ, которые предлагают подходы к формированию команд или выбор участников основываясь на их предыдущий опыт и компетенции [1, 2]. Некоторые работы предлагают извлекать компетенции путем анализа документов предполагаемых членов команды, чтобы найти людей с конкретным опытом [3, 4; 61-75]. В образовательном процессе также можно применить подобный подход, основываясь на том, что студенты за время обучения выполняют различные работы, из которых можно извлечь опыт студентов [5].

Проблема исследования. При выполнении этой задачи участники образовательного процесса сталкиваются со следующими проблемами: отсутствие единой базы данных, содержащей всю необходимую информацию; отсутствие доступа к информации у некоторых участников образовательного процесса, так как отчеты хранятся в различных командах в приложении Microsoft Teams, доступ к которыми имеют научные руководители проектов; времязатратность

анализа отчетов — на поиск необходимой информации из отчета может уйти от 15 до 20 минут, а на поиск определенного студента среди всех отчетов уйдет еще больше времени.

Таким образом возникает проблема отсутствия инструмента для поиска студентов, обладающих определенными навыками и владеющих необходимыми технологиями.

Цель — разработать веб-приложение для извлечения из проектного отчета:

1) Ф.И.О. лиц, выполнивших работу;

2) выполненных задач;

3) стек использованных технологий для каждой из задач;

4) соотношения задач и использованных технологии со студентами;

1) заполнения базы данных извлеченной информацией.

Материалы и методы. Для обучения классификатора задач были использованы отчеты студентов по проектно-технологическим практикам, которые представляют из себя набор предложений, каждое из которых либо является задачей, либо не является. Задача не должна относиться к организационным событиям проектно-технологической практики. Пример подготовленных данных приведен на рис. 1.

Осознание того, что во время студенческой конференции предстоит выступить в роли спикера, было волны	0
Маргарита хорошо поработала, успешно выполнив все свои задачи	0
Мадина Куанышевна Сулейманова	0
Интересы в сфере разработки ПО	0
Системное программирование, анализ данных, DevOps	0
Во время прохождения технологической практики получены навыки работы с данными и проектирования	0
Задачи	0
Поиск способов сбора данных	1
Просмотр и изучение приложений различных банков и выписок из них по доходам и расходам	1
Анализ и обработка данные, полученных из выписок банков	1
Сбор и перенос данных в единую таблицу Excel	1
Подбор, изменение базовых категории	1
редактирование, дополнение уже имеющихся данных	1
Составление презентации для конференции	0
Помощь в составлении документов	0
Написание данных в виде SQL-запросов	1
Добавление данных для тестирования приложения	1
Создание первоначальной базы данных в SQLite	1
Знакомство со средой разработки Android Studio	1
Проблемы	0
Трудности в заказе выписок из банков за долгий период времени , т. к. банки выполняют такие запросы за	0

Рис. 1. Размеченные очищенные предложения

По исходным данным необходимо обучить бинарный классификатор, который позволит извлекать из отчетов задачи, являющиеся короткими текстами [6, 7, 8], также необходимо выделить исполнителя задачи [9] и технологии использованные для выполнения задачи.

Для удобства и последующих ссылок на фрагменты разделим предобработку текста на шаги:

Шаг 1. Изначально нам необходимо очистить исходный текст документа: удалить символы, не состоящие в таблице ASCII, числа, служебные символы.

Шаг 2. Так как наша задача состоит в извлечении подходящих под условие предложений нам требуется разделить текст на соответствующие токены. Для токенизации на предложения был использован NLTK — пакет Python библиотек и программ для символьной и статистической обработки естественного языка.

Шаг 3. Далее закрепим за каждым предложением его собственный набор слов: каждое предложение токенизируется с помощью NLTK на слова, все слова приводятся к нижнему регистру, лемматизируются с помощью Pymorphy2 — морфологического анализатора для русского языка [10], использующего словари из OpenCorpora, удаляются стоп-слова. Лемматизация — приведение слова к начальной форме, она необходима для уменьшения размерности исходного набора слов, что в будущем ускорит обучение и классификацию. Стоп-слова — это слова, которые не несут никакой смысловой нагрузки например: “и”, “или”, “это”. То есть эти слова являются лишь шумом для нашего классификатора, так как не являются признаком принадлежности к классу “задача” или “не задача”.

Шаг 4. Для дальнейшей работы нужно перевести текстовое содержимое в числовой вектор признаков — провести tf-idf векторизацию каждого набора слов [8]. Tf-idf показатель отражает важность слова, в нашем случае, в контексте предложения, являющегося частью набора предложений.

Для начала надо произвести обучение классификатора на очищенных предложениях, полученных из “Шаг 2” пункта “Предобработка отчетов”, впоследствии размеченных на классы “1” и “0”, где “1” — задача, не относящаяся к организационным событиям проектно-технологической практики, “0” — не интересующие нас

предложения. Пример данных приведен на рис. 1. После считывания размеченные по классам предложения пройдут “Шаг 3” и “Шаг 4” пункта “Предобработка отчетов”.

В качестве модели классификатора был выбран SGDClassifier из библиотеки машинного обучения scikit-learn. По умолчанию SGDClassifier представляет из себя метод опорных векторов SVM (Support Vector Machine) [6, 8] с оптимизационным алгоритмом обучения — стохастическим градиентным спуском [11; 150]. Альтернативой методу опорных векторов был наивный байесовский классификатор Naive Bayes [6, 8], который тоже демонстрирует высокую точность в решении задач обработки естественного языка [6], но исходя из нашего исследования SVM каждый раз демонстрировал более высокие показатели точности на тестовой выборке, результаты исследования показаны на рис. 2.

Так как наша выборка несбалансированная мы не можем использовать метрику “accuracy” — доля верных ответов, потому что это приведет к завышенной оценке качества работы классификатора. Поэтому мы возьмем сразу несколько метрик, устойчивых к распределению классов: “precision”, “recall”, “f1-score” [6]. Также мы можем смотреть на “macro avg” [6] — подсчитывается каждая метрика по каждому классу, далее усредняется, чтобы сразу увидеть какой из сравниваемых классификаторов показывает лучшие результаты. В нашем случае показатель “macro avg” был выше у SVM и был равен ~0.88 на протяжении 100 опытов, что показано на рис. 2.

Также была обучена модель tf-idf векторизатора — TfidfVectorizer из библиотеки машинного обучения scikit-learn для ее реализации в “Шаг 4” пункта “Предобработка отчетов”. В итоге полный алгоритм от получения документа до классификации предложений представлен на рис. 3.

Для извлечения имен студентов, выполнивших работу, необходимо выделить титульный лист отчета.

После этого с помощью набора Python библиотек для решения задач обработки естественного языка — Natasha, которая использует базы данных имен на русском языке. Извлечение Ф.И.О. лиц, выполнивших работу, происходит по ключевому для нас слову “выполнил”, которое является для нас меткой того, что человек участвовал в работе над проектом.

```
Метод опорных векторов
              precision    recall  f1-score   support

         0.0       0.96      0.94      0.95       354
         1.0       0.78      0.86      0.82        94

    accuracy                           0.92       448
   macro avg       0.87      0.90      0.88       448
weighted avg       0.92      0.92      0.92       448

Наивный Байесовский классификатор
              precision    recall  f1-score   support

         0.0       1.00      0.84      0.91       409
         1.0       0.37      0.97      0.53        39

    accuracy                           0.85       448
   macro avg       0.68      0.91      0.72       448
weighted avg       0.94      0.85      0.88       448
```

Рис. 2. Показатели точности классификаторов на тестовой выборке

Рис. 3. Алгоритм предобработки и классификации предложений

Далее требуется соотнести человека и выполненные им задачи. Для этого мы изучили в каких формах студенты прописывают эту информацию и выделили несколько паттернов:

1	2	3	4	5	6
ФИО	ФИО	ФИО	Задача	Задача	Задача
ФИО	Задача	Задача	ФИО	ФИО	Задача
ФИО	Задача	ФИО	Задача	ФИО	Задача
Задача	Задача	Задача	ФИО	ФИО	ФИО
Задача	ФИО	ФИО	Задача	Задача	ФИО
Задача	Задача	Задача	ФИО	ФИО	ФИО
ФИО	Задача	ФИО	Задача	ФИО	Задача
ФИО	Задача	Задача	ФИО	ФИО	Задача
ФИО	ФИО	ФИО	Задача	Задача	Задача
Задача	Задача	Задача	ФИО	ФИО	ФИО
Задача	Задача	ФИО	Задача	ФИО	ФИО
Задача	Задача	Задача	ФИО	ФИО	ФИО

Рис. 4. Паттерны имен студентов и выполненных ими задач

Исходя из результатов наших наблюдений, представленных на рис. 4, мы разработали алгоритм:

Создадим 2 стека: для имен и задач;

Каждое имя и задача помещаются в соответствующий им стек;

Для прохождения полного цикла обработки паттерна нам необходимо пройти все итерации. Итерация цикла состоит в добавлении в соответствующий токену стек всех предложений сначала одного затем другого вида. В нашем случае токены это предложения, которые являются либо именами, либо задачами, от незначащих предложений список токенов был очищен. Итерация цикла завершается, когда в стек были добавлены все токены 2 вида. Например, после добавления всех имен, добавления всех задач, нам снова встретилось имя и соответственно наоборот. Разделение итераций показано на рис. 5, где зелеными границами обозначена итерация цикла:

1	2	3	4	5	6
ФИО	ФИО	ФИО	Задача	Задача	Задача
ФИО	Задача	Задача	ФИО	ФИО	Задача
ФИО	Задача	ФИО	Задача	ФИО	Задача
Задача	Задача	Задача	ФИО	ФИО	ФИО
Задача	ФИО	ФИО	Задача	Задача	ФИО
Задача	Задача	Задача	ФИО	ФИО	ФИО
ФИО	Задача	ФИО	Задача	ФИО	Задача
ФИО	Задача	Задача	ФИО	ФИО	Задача
ФИО	ФИО	ФИО	Задача	Задача	Задача
Задача	Задача	Задача	ФИО	ФИО	ФИО
Задача	Задача	ФИО	Задача	ФИО	ФИО
Задача	Задача	Задача	ФИО	ФИО	ФИО

Рис. 5. Итерации цикла по обработке паттернов

После прохождения итерации к каждому полученному имени из стека имен прикрепляются все задачи из стека задач. Способ соотношения имен и задач показан на рис. 6.

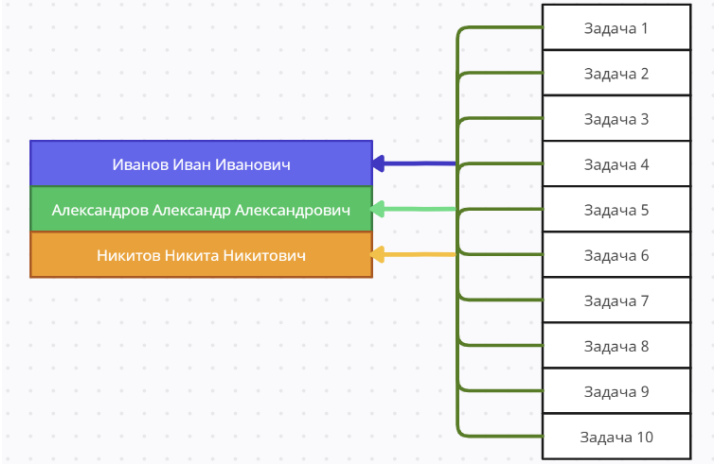
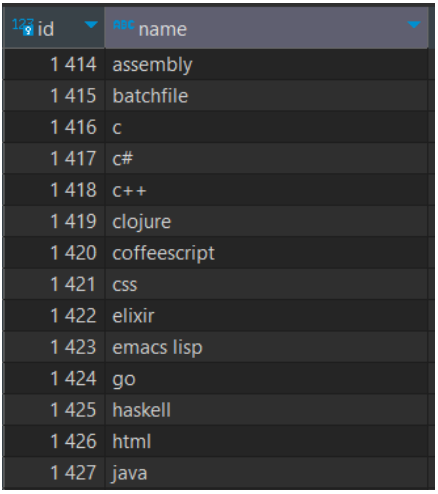


Рис. 6. Способ соотношения имен и задач из стеков

В итоге формируется словари для каждого студента, содержащие его Ф.И.О. и выполненные им задачи.

Для добавления в словарь использованных технологий будем использовать ранее заполненную нами таблицу, которая содержит языки программирования, фреймворки, СУБД, библиотеки. Пример данных показан на рис. 7:



id	name
1 414	assembly
1 415	batchfile
1 416	c
1 417	c#
1 418	c++
1 419	clojure
1 420	coffeescript
1 421	css
1 422	elixir
1 423	emacs lisp
1 424	go
1 425	haskell
1 426	html
1 427	java

Рис. 7. Пример технологий из сформированной таблицы.

Для извлечения технологий токенизируем задачи на слова и произведем поиск токенов в таблице. Для технологий, состоящих из 2, и более слов будет искать их вхождение в текст.

В итоге получим словари, каждый из которых содержит Ф.И.О студента, выполненные задачи и использованные технологии.

Для хранения информации была разработана база данных на СУБД SQLite. Схема базы данных предоставлена на рис. 8.

Таблица main_student содержит собственный номер и Ф.И.О студента;

Таблица main_task содержит собственный номер, текст задачи и номер студента, выполнившего эту задачу. Эта таблица объединяет студентов и выполненные ими задачи.

Таблица main_technology содержит собственный номер и название технологии. Эта таблица была сформирована ранее и хранит в себе общий стек технологий.

Таблица main_technology_used содержит собственный номер, номер задачи к которой относится, номер технологии. Эта таблица объединяет задачи и использованные для реализации этих задач технологии.

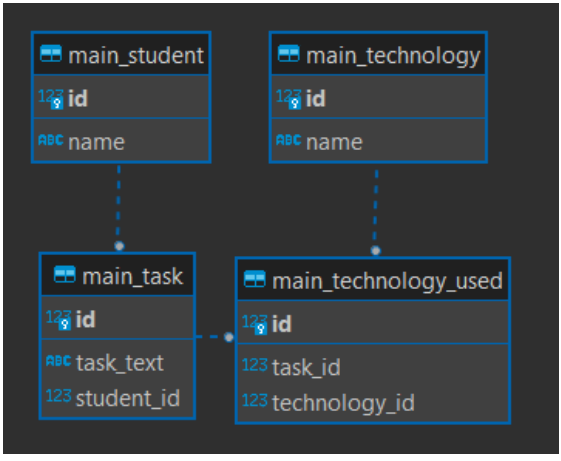


Рис. 8. Схема базы данных для хранения информации о студентах и общего стека технологий

Для реализации веб-приложения мы использовали Python фреймворк Django. Схема работы веб-приложения представлена на рис. 9.

Интерфейс и функционал веб-приложения продемонстрирован на следующих рисунках 10-13.

Результаты. Разработано веб-приложение, которое может извлекать из отчетов по проектно-технологическим практикам имена студентов, участвовавших в работе над проектом, выполненные ими задачи и использованные для решения задач технологии из загруженных на сайт отчетов. Также функционал приложения позволяет искать студента по запросам: определенному имени, задачи, технологии.

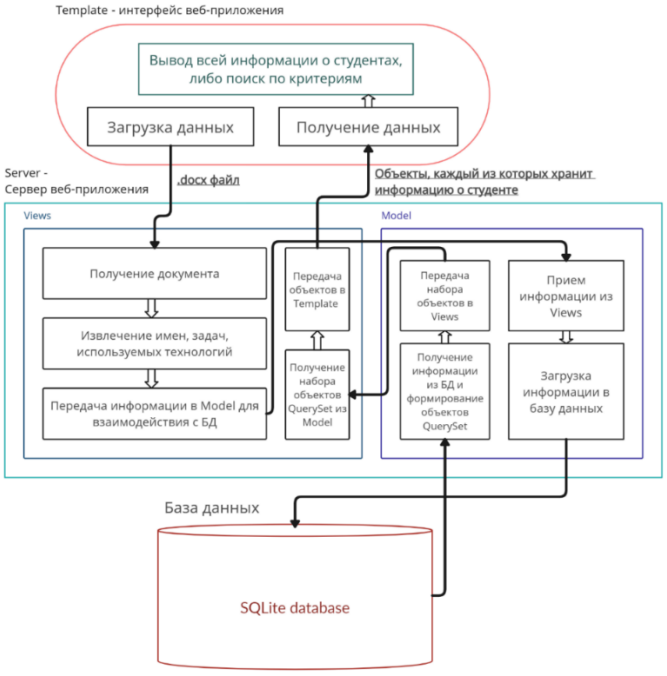


Рис. 9. Схема работы веб-приложения

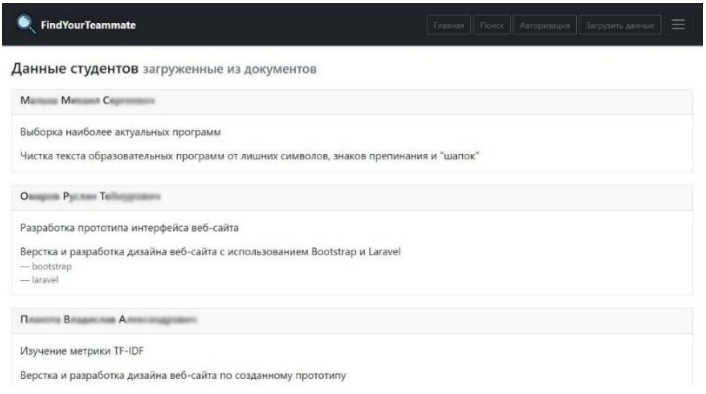


Рис. 10. Главная страница сайта

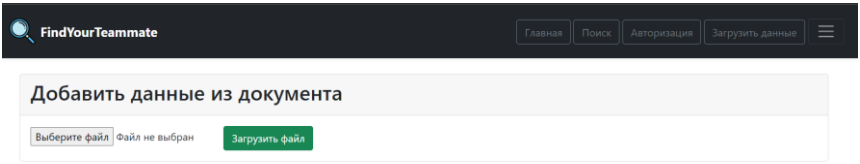


Рис. 11. Страница загрузки отчетов

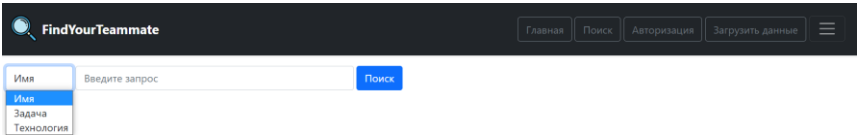


Рис. 12. Страница поиска студентов по запросу

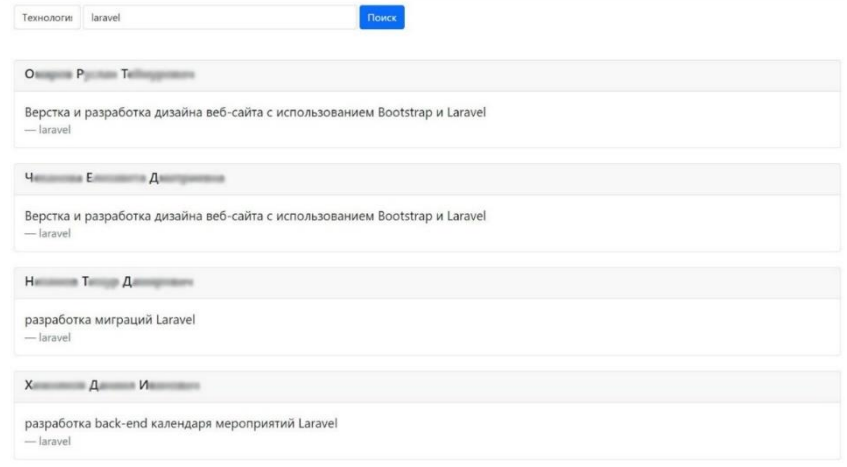


Рис. 13. Пример результатов поиска студентов
по определенному запросу

Заключение. В работе были представлены методы достижения поставленной цели и итоговое решение проблемы. Приведено обоснование выбора модели классификатора и разработанных алгоритмов. Продемонстрирована работоспособность обученного классификатора.

Для решения проблемы был разработан алгоритм предобработки текста, подходящий под наши цели, обучен бинарный классификатор, разработан алгоритм соотношения студентов и выполненных ими задач, разработана база данных и собрана таблица общего стека технологий. Все разработки были успешно внедрены в веб-приложение.

СПИСОК ЛИТЕРАТУРЫ

1. Hlaoittinun O. A team building approach for competency development / O. Hlaoittinun, E. Bonjour, M. Dulmet. — DOI 10.1109/IEEM.2007.4419343. — Direct text // 2007 IEEE International Conference on Industrial Engineering and Engineering Management. — Singapore : IEEE. — 2007. — P. 1004-1008.
2. Fapohunda T. M. Towards effective team building in the workplace / T. M. Fapohunda. — Direct text // International journal of education and research. — 2013. — Vol. 1, № 4. — P. 1-12.
3. Rodrigues S. Competence mining for team formation and virtual community recommendation / S. Rodrigues, J. Oliveira, J. M. de Souza. — DOI 10.1109/CSCWD.2005.194143. — Direct text // Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design. — Coventry : IEEE. — 2005. — P. 44-49.
4. Larusson J. A. Learning analytics: From research to practice / J. A. Larusson, B. White. — New York : Springer Publ, 2014. — 195 p. — Direct text.
5. Диагностика профессиональной компетентности студентов ИТ-направлений на основе данных цифрового следа / И. Г. Захарова, Ю. В. Боганюк, М. С. Воробьева, Е. А. Павлова. — Текст : непосредственный // Информатика и образование. — 2020. — № 4 (313). — С. 4-11.
6. Short Text Classification: A Survey / G. Song, Y. Ye, X. Du [et al.]. — Direct text // Journal of Multimedia. — 2014. — № 9. — P. 635-643.
7. Kateb F. Classifying Short Text in Social Media: Twitter as Case Study / Faris Kateb, Jugal Kalita. — Direct text // International Journal of Computer Applications. — 2015. — Vol. 111, № 9. — P. 1-12.

8. Батура Т. В. Методы автоматической классификации / Т. В. Батура // Программные продукты и системы. — 2017. — Т. 30, № 1. — С. 85-99.
9. Li S. «Named Entity Recognition with NLTK and SpaCy»: NER is used in many fields in Natural Language Processing (NLP) / S. Li. — URL: https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da. (date of the application 17.08.2018). — Text : electronic.
10. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages / M. Korobov. — DOI 10.1007/978-3-319-26123-2_31 — Direct text // Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers. — Yekaterinburg, 2015. — P. 320-332.
11. Shalev-Shwartz S. Descent Understanding Machine Learning: From Theory to Algorithms / S. Shalev-Shwartz, S. Ben-David. — Cambridge : Cambridge University Press Publ, 2014. — 397 p. — Direct text.