Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Соломатин Роман Игоревич

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ПОИСКА ИСПОЛНИТЕЛЕЙ ПО ТЕХНИЧЕСКОМУ ЗАДАНИЮ ПРИКЛАДНОГО ПРОЕКТА

Курсовая работа

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Руководитель
к.т.н., доцент кафедры Информационных технологий в бизнесе НИУ ВШЭ-Пермь

А. В. Бузмаков

Пермь, 2021 год

Аннотация

Оглавление

Введение

Задача поиска исполнителей очень актуальна для многих сфер жизни. В каждой компании появляется много заданий и не понятно, кто лучше с ними справится. Поиск исполнителей необходим для того, чтобы как можно более эффективно использовать ресурсы. В «Высшую школу экономики» приходит множество проектов для выполнения, и не понятно кто их может выполнить. Иногда поиск нужного исполнителя, знания которого подходят для данного проекта, проходит очень долго.

В этой работе будет проверятся гипотеза можно ли из выпускных курсовых работ студентов получить сферу компетенций преподавателя и подобрать по тексту ближайший профиль преподавателя.

Объект исследования - процесс поиска исполнителей по тех заданию.

Предмет исследования - автоматизация процесса из объекта.

Цель работы – создать информационную систему для поиска исполнителя по техническому заданию.

Для достижения поставленной цели нужно сделать:

Во второй главе описание проектирования системы. В третьей главе пример работы приложения.

Глава 1. Анализ предметной сферы

1.1. Обзор существующих решений

Не существует программных продуктов, которые решают данную задачу в явном виде. Потому что это очень особенная ситуация, когда есть много исполнителей с разными компетенциями и надо для них подбирать задания. Также редко в каких организациях по текстам, которые пишут исполнители можно представить их компетенции. Сейчас это проблема решается вручную.

Приходит в ВШЭ много технических заданий, из отбирает человек, который знает компетенции многих исполнителей. После этого если он думает, что компетенции исполнителя подходят для проекта, то пишет ему. Сотрудник отвечает готов или не готов. Потом за короткий промежуток времени нужно подготовить заявку на проект, задать уточняющие вопросы организатору, найти недостающих исполнителей. Данный процесс занимает много времени. Эта система имеет недостатки:

- Много проектов теряется, так как человек не знает все компетенции всех исполнителей
- Сложно искать проекты
- Тяжело масштабировать, потому что необходимо знать много про разных людей
- Тратится много времени

Иногда применятся способ массовой рассылки требующихся исполнителей для проекта. У данного подхода недостатки:

- Массовую рассылку не все читают
- Не дает полную информацию

Разрабатываемая система поможет автоматизировать процессы:

- Определения компетенций сотрудника
- Поиск сотрудников для выполнения задания

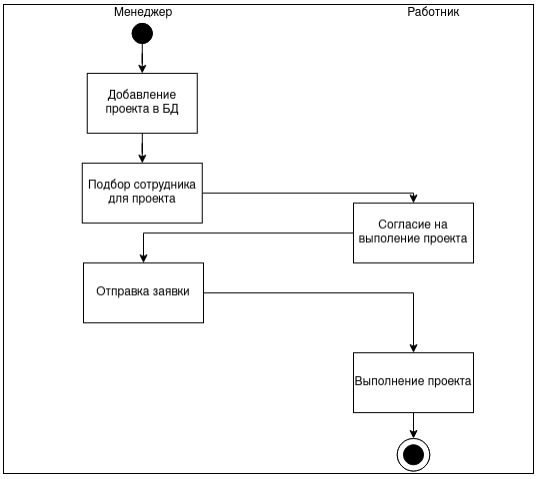

Рис. 1.1. Автоматизация бизнес-процесса "Подбор сотрудника для проекта"

Варианты использования реализуемой системы:

- Просмотр активных проектов
- Поиск компетенций исполнителя
- Подбор сотрудника для проекта
- Редактирование информации о исполнителя

Так как в данной работе нужно проверить гипотезу, будет разрабатываться прототип программы. Диаграмма вариантов использования конечного программного продукта 1.2.

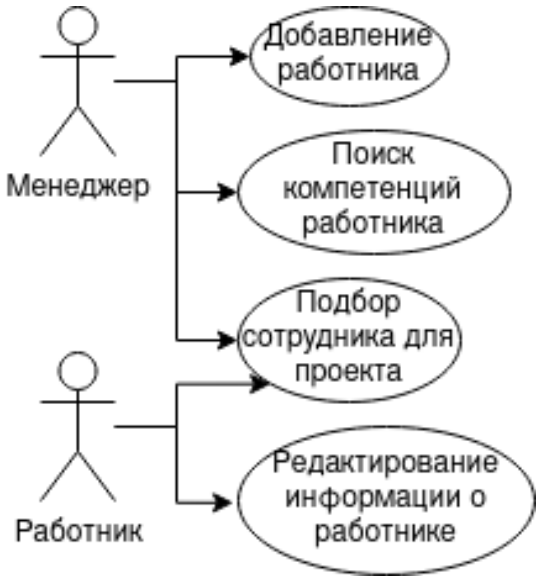

Рис. 1.2. Диаграмма вариантов использования

1.2. Выбор языка программирования

Для прототипирования подходят следующие языки: Python, C#, Java, JavaScript. Основными критериями для выбора языка были:

- Библиотеки для обучения моделей. Для работы с моделями машинного обучения.
- Предобученные модели. Так как для обучения модели надо иметь большой объем данных и много вычислительных мощностей.
- Библиотека для обработки docx и pdf файлов и обработки сайтов.
- Простота написание и отладки программы.
- Интерактивный режим.

Под эти критерии подходит Python. Он простой для написание и отладки, так как это интерпретируемый язык программирования, у него есть интерактивный режим. Также есть фреймворк PyThorch и библиотека с предобученными моделями HuggingFace Transformers, в частности RuBERT [1], а также есть библиотека Bert Extractive Summarizer [2] для удобной работы с моделей.

1.3. Выбор СУБД

Для Python существуют библиотеки для работы с любыми системами управления базами данных. Для курсовой работы был выбран SQLite потому что его легко настраивать, нет необходимости устанавливать ничего дополнительно, что достаточно для прототипирования с небольшим количеством данных.

Глава 2. Проектирование системы

2.1. Проектирование базы данных

Для хранение информации о преподавателях и выпускных квалификационных работах, где они были руководителями была разработана база данных. Необходимо было хранить:

Код преподавателя – уникальный код преподавателя

ФИО преподавателя – фамилия имя и отчество преподавателя

Ссылка на профиль преподавателя – ссылка на профиль преподавателя на сайте Высшей школы экономики

Компетенции – полученные компетенции преподавателя

Эмбеддинги – компетенции преподавателя в векторном пространстве

Код ВКР – уникальный код выпускной квалификационной работы

Название ВКР – название выпускной квалификационной работы

Код преподавателя – код преподавателя

Ссылка на ВКР – ссылка на выпускную квалификационную работу на сайте ВШЭ

Ссылка на полный текст ВКР – ссылка на файл для загрузки с полным текстом ВКР

ФИО студента – фамилия имя и отчество студента

Код ОП студента – код образовательной программы студента

Код статуса – уникальный код статуса

Наименования статуса – доцент, старший научный сотрудник и тд.

Код кафедры – уникальный код кафедры

Наименование кафедры – название кафедры

Код кампуса – уникальный код

Наименования кампуса – название кампуса

Код образовательной программы – уникальный код

Код факультета – код факультета

Наименование ОП – название образовательной программы

Код факультета – уникальный код факультета

Наименование факультета – название факультета

Описание данных для проектирования БД 2.1.

2.1.1. Приведение к 1НФ

Отношение находится в первой нормальной форме, если выполнены все свойства реляционных отношений, в частности все атрибуты отношения принимают простые значения (атомарные или неделимые), не являющиеся множеством или кортежем из более элементарных составляющих, все кортежи уникальны (отсутствуют дубли).

1. Код преподавателя
2. ФИО преподавателя
3. Ссылка на профиль преподавателя
4. Компетенции
5. Эмбеддинги
6. Код статуса преподавателя
7. Код кафедры преподавателя
8. Код ВКР
9. Название ВКР

Таблица 2.1. Таблица атрибутов

Имя атрибута	Тип данных	Значение по умолчанию	Формат ввода	Ограничение на значения
Код преподавателя	Число	Нет	Нет	Нет
ФИО преподавателя	Строка	Нет	Нет	Нет
Ссылка на профиль преподавателя	Строка	Нет	Нет	Нет
Компетенции	Строка	Нет	Нет	Нет
Эмбеддинги	Число	Нет	Нет	Нет
Код ВКР	Число	Нет	Нет	Нет
Название ВКР	Строка	Нет	Нет	Нет
Ссылка на ВКР	Строка	Нет	Нет	Нет
Ссылка на полный текст ВКР	Строка	Нет	Нет	Нет
ФИО студента	Строка	Нет	Нет	Нет
Код ОП студента	Число	Нет	00.00.00	Нет
Код статуса	Число	Нет	Нет	Нет
Наименования статуса	Строка	Нет	Нет	Нет
Код кафедры	Число	Нет	Нет	Нет
Наименование кафедры	Строка	Нет	Нет	Нет
Код кампуса	Число	Нет	Нет	Нет
Наименования кампуса	Строка	Нет	Нет	Нет
Код образовательной программы	Число	Нет	00.00.00	Нет
Код факультета	Число	Нет	Нет	Нет
Наименование ОП	Строка	Нет	Нет	Нет
Код факультета	Число	Нет	Нет	Нет
Наименование факультета	Строка	Нет	Нет	Нет

10. Код преподавателя

11. Ссылка на ВКР

12. Ссылка на полный текст ВКР
13. ФИО студента
14. Код кампуса
15. Код ОП студента
16. Код статуса
17. Наименование статуса
18. Код кафедры
19. Наименование кафедры
20. Код кампуса
21. Наименования кампуса
22. Код образовательной программы
23. Код факультета
24. Наименование ОП
25. Код факультета
26. Наименование факультета

Данные атрибуты находятся в 1НФ.

2.1.2. Приведение к 2НФ

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый неключевой атрибут функционально полно зависит от всего ключа в целом, то есть отсутствует частичная функциональная зависимость не ключевых атрибутов от ключа. В соответствии с описанными выше зависимостями можно сделать вывод, что в описанном отношении имеются следующие частичные зависимости:

1. Код преподавателя определяет:
    - ФИО преподавателя

- Ссылка на профиль преподавателя
- Компетенции
- Эмбеддинги
- Код статуса преподавателя
- Код кафедры преподавателя

2. Код ВКР определяет:
   - Название ВКР
   - Код преподавателя
   - Ссылка на ВКР
   - Ссылка на полный текст ВКР
   - ФИО студента
   - Код кампуса
   - Код ОП студента
3. Код статуса преподавателя определяет:
   - Наименование статуса
4. Код факультета определяет:
   - Наименование факультета
5. Код кампуса определяет:
   - Наименование кампуса
6. Код кафедры определяет:
   - Наименование кафедры
7. Код образовательной программы определяет:
   - Название образовательной программы

2.1.3. Приведение к 3НФ

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме, и каждый неключевой атрибут не является транзитивно зависимым от первичного ключа.

2.2. Сбор данных

Для сбора данных надо было обрабатывать страницы сайтов. Для этого использовалась библиотека BeautifulSoap.

В первую очередь надо было получить список преподавателей и ссылки на их страницы. Для этого обрабатывался сайт со списком преподавателей, но там можно получить преподавателей, фамилии которых начинаются на 1 букву. Что бы решить эту проблему с сайты получались ссылки на буквы. А потом с каждой такой ссылки собирался список преподавателей и ссылки на их страницы. Для этого брались все элементы страницы с атрибутом "a" и классом "link" , в которых значение атрибута "href" было "/org/persons/" или "/staff/". В базу данных записывались ФИО преподавателя и ссылку на их страницу.

Далее надо было получить список ВКР для каждого преподавателя. Для этого брались все элементы страницы с атрибутом "a" и классом "link" , в которых значение атрибута "href" было "/edu/vkr/". Итого получилось собрать информацию о 321 преподавателе и 1708 ВКР.

Потом надо было получить информацию о ВКР, но их данные ВКР, в отличие от данных преподавателей, подгружались после загрузки основой страницы. Чтобы решить эту проблему пришлось эмулировать работу браузера с помощью библиотеки Selenium и драйвера Gecko (движок браузера FireFox). Для полной загрузки страницы ставилась задержка в 2 секунды. И бралась информация атрибута "p" с классом "vkr-card__item". Также бралась информация о научном руководителе, он искался в базе данных, и потом в базу данных уже записывался Код преподавателя.

Чтобы получить тексты из файлов, они обрабатывались с помощью textract и помещались в базу данных.

2.3. Создание профиля человека

Для создание профиля человека рассматривались TFiDF и суммаризация текста с помощью BERT[3] (Bidirectional Encoder Representations from Transformers).

Подход TFiDF заключается в том что бы для каждого слова посчитать его отношение числа вхождений некоторого слова к общему числу слов документа (TF) и частоты, с которой некоторое слово встречается в документах коллек-

ции (DF). Данный подход часто используется для кластеризации текстов. Но данный метод плохо подходит для данной задачи, потому что он не рассматривает контекст используемых слов. Это значит у него проблемы с омонимами и синонимами.

BERT рассматривает контекст, в котором находится слово. Каждое слово преобразуется в векторное пространство модели. И модель идет в каждом предложении с лева направо и с право налево пытаясь предсказать, какое будет следующее слово. Для того что бы модель не смотрела в 'будущее' убирается примерно 15% слов из предложения и модель пытается предсказать, что находится на этом месте.

Для получение компетенций преподавателя использовались тексты ВКР, которые у него писали студенты. Для этого обрабатывался сайты Высшей школы экономики и скачивались тексты ВКР. Потом эти тексты группировались по преподавателю и обрабатывались с помощью BERT Extractive summarizer, в качестве модели для обучения использовался предобученный RuBERT от DeepPavlovAI.

BERT Extractive summarizer – использует предпредпоследний слой модели в качестве данных. Потом использует алгоритм K-Means для кластеризации текстов.

Глава 3. Поиск человека

Заключение

Библиографический список

1. *Kuratov Y.*, *Arkhipov M.* Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language. — 2019.
2. *Miller D.* Leveraging BERT for Extractive Text Summarization on Lectures. — 2019.
3. *Devlin J.*, *Chang M.-W.*, *Lee K.*, *Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. — 2019.

Приложения