ИТОГОВЫЙ ПРОЕКТ «**Создание рекомендательной системы для соискателей а рынке труда»**

**1. Анализ предметной области и обзор существующих решений**

Проект направлен на решение актуальной проблемы сопоставимости компетенций соискателей и требований к ним, записанных в вакансиях на открытых дистанционных рынках труда.

В рамках финальной работы основной целью является проведение анализа неструктурированных частей вакансий, размещаемых на популярных «работных сайтах» в России. Такой анализ позволит выделить ключевые требования к соискателям на рынке труда и предложить им рекомендации при поиске подходящей работы.

Сложность исследования заключается в том, что в качестве исходной базы используется корпус текстов, пока еще с большим трудом поддающийся обработке при помощи автоматизированных методов анализа. Тем не менее, в последние годы машинный анализ текстов привлекает все большее внимание со стороны ИТ-компаний, отдельных исследователей, бизнес-практиков и консультантов, государственных и частных организаций, поскольку именно текст выступает наиболее «человеческим» из всех возможных видов деятельности и именно он считается наиболее сложным в изучении. Кроме того, объемы текстовых данных непрерывно возрастают и в настоящее время занимают значительную часть всей произведенной людьми информации. Соответственно, увеличивается потребность в развитии методов их компьютерной обработки и анализа.

Одной из важных задач автоматического анализа текстовой информации выступает классификация, которая может реализоваться разными методами. При использовании алгоритмов обучения с учителем речь идет, как правило, о разметке новых текстов на основе существующих примеров. Такие способы востребованы, например, в переводе или при определении тональности документов. Альтернативным способом является классификация данных без обучающих образцов. В этом случае наиболее распространенными методами решения задачи являются кластеризация и тематическое моделирование. Данные методы помогают обрабатывать значительные объемы информации, а в результате могут привести к получению новых знаний и обнаружению неявных связей, недоступных в иных случаях.

Обе статистические модели — кластеризация и тематическое моделирование — во многом похожи, однако, разница между ними заключается в том, что кластеризация предполагает соотнесение конкретного слова или документа с определенным набором (кластером), а тематическое моделирование содержит идею о том, что одни и те же слова и документы могут комбинироваться в разные темы.

**2. Исследование предметной области**

В неструктурированных частях вакансий и резюме, размещенных на сайтах, содержится важная информация о требованиях к соискателям со стороны фирм и потенциальных ресурсах самих работников, которые могут быть задействованы в российской экономике. Совместимость требований и ресурсов способствует тому, что работник легко находит работу, а работодатель — потенциальных кандидатов на должности. В реальности, однако, сами люди и hr-специалисты вынуждены перебирать сотни размещенных объявлений, тратить время и ресурсы на проведение собеседований, вводить испытательные сроки и совершать множество иных действий. В результате, экономике не хватает кадров, а люди либо вынуждены соглашаться на первое предложенное рабочее место, либо долго и безрезультатно продолжают свои поиски. Особая область деятельности — people analitics — в настоящее время слабо развита. Ей не хватает знаний, алгоритмов, значительных объемов информации. Так, в традиционной hr-деятельности (в некоторых случаях на предприятиях остаются еще более устаревшие отделы кадров), как правило, не используются современные озможности машинного анализа. Очень редко проводятся исследования тестовых частей информации для отбора кандидатов. В целом, как сами «работные сайты», так и работодатели чаще ориентируются на формализованные данные, например, социально-демографические сведения о соискателях, наличии образования и трудового стажа. Однако постоянная нехватка кадров показывает, что, например, полученное соискателем образование может не соответствовать конкретным требованиям, быть недостаточным или, наоборот, избыточным. Само по себе наличие образования еще не гарантирует востребованных компетенций у работника, так как оно зачастую мало адаптировано к требованиям рынка труда. В этой связи большое значение имеет работа по развитию компетенций у соискателей в составлении резюме, а для работодателей — в их всесторонней оценке и анализе.

В данном проекте реализуется в тестовом режиме следующая цель: при помощи компьютерного анализа текстовых частей вакансий, размещенных на работных сайтах в России, сформировать представление о востребованных компетенциях, которые могли бы быть использованы соискателями для согласования имеющихся знаний и навыков с искомыми рабочими местами или для их улучшения (повышения квалификации).

**3. Проектирование системы**

**3.1. Постановка задачи**

Реализация данного проекта предполагает последовательное выполнение следующих шагов:

- Определение выборочной совокупности и ее размещение в БД

- Отбор сущностей для анализа и формирование таблиц с данными

- Проведение тематического моделирования корпуса текстов с определением оптимальных метрик качества

- Определение доминирующих тематик в документах и их соотнесение с предлагаемой зарплатой

- Разработка веб-приложения, предлагающего в качестве результата рекомендации для соискателя по диапазону искомой зарплаты.

**3.2.База данных**

Источником данных для проекта служит готовая БД вакансий и резюме, собранных с сайтов HeadHunter, TrudVsem и Zarplata.ru, при помощи приложения, реализованного на языке json. Общий объем данных за период с 2019 г. по 2023 г. содержит:

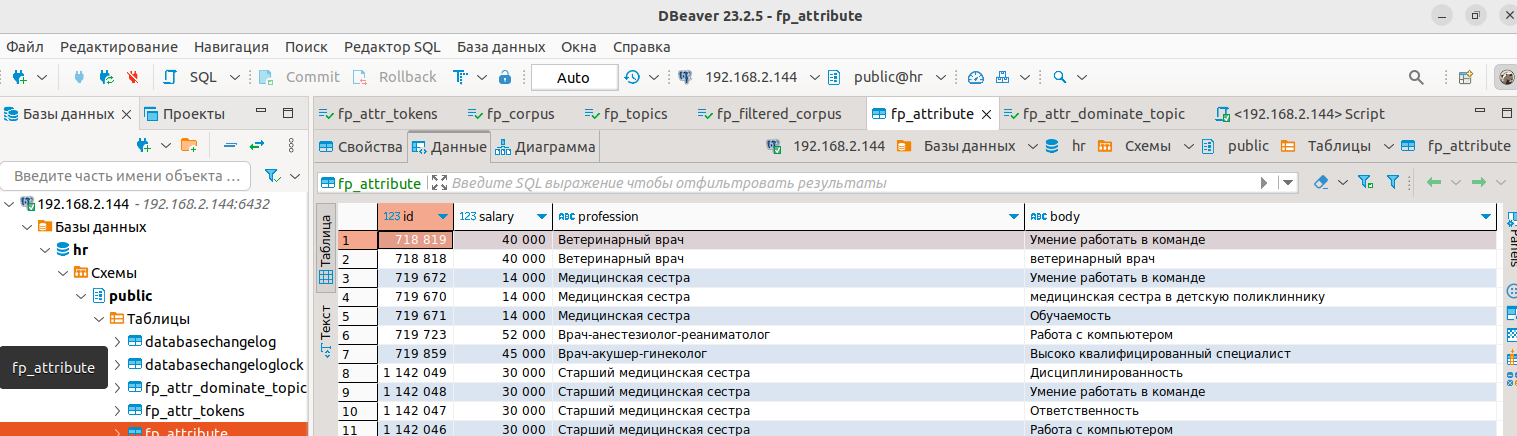
* вакансии: 3.887.121 сущностей, содержащие 28.779.613 атрибутов свободного текста (требования и должностные обязанности);
* резюме: 1.460.684 сущностей, содержащие 6.304.783 атрибутов свободного текста (знания/навыки и опыт работы);
* классификаторы суммарным объемом 3.594 записи.

Для данного проекта были отобраны

1) только некоторые служебные, формализованные и неструктурированные поля из набора вакансий, содержащие атрибуты «уникальный индентификатор», «тип» (включая текст на естественном языке), «зарплата от» и «зарплата до»;

2) набор вакансий за 1 кв.2023 г., соответствующих медицинской сфере деятельности.

Итоговый набор данных содержит 4 поля и 217861 запись. Таблица c данными (fp\_attribute), размещенная в СУБД PostgreSQL, имеет следующий вид:



Поля таблицы fp\_attribute:

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Значение |
| id | uuid | Идентификатор атрибута «Требования к работнику» вакансии |
| salary | целочисленный | Средняя зарплата по атрибутам «Зарплата от» и «Зарплата до», исключались пустые значения, округление до 1000 |
| profession | строка | Наименование должности в вакансии |
| body | строка | Свободный текст из атрибута «Требования к работнику» |

Поле «body» содержит предобработанный свободный текст, включающий:

1) Из первоначального источника данных удалились

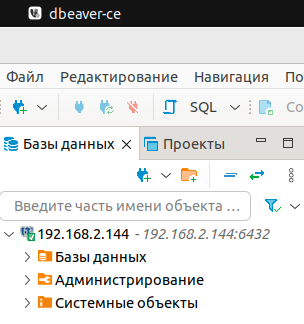
* удалялись html-теги;
* удалялись спецпоследовательности, например, "&nbsp", "@quot", "&raqu", "&laqu", "&amp", "&qt", "&bull";
* выделялся текст с фразами в соответствии с регулярным выражением "\.|;|\||\n";
* удалялись в начале и в конце фразы пустые символы из списка: ' ', ',', '\_', '!', '?', '—', '-', '●', '•', '·', ':', '✅', '\*', '✳', '▪', '└',' ', '\uF02D', 'o', '◦', '©', '®', '♦', '\uF0D8', '\uF020', '\u2028', '\uF0B7', '\uF06E', '\uF0A7', '\uF0BE', '\t', '\r', '\n'.

2) Свободный текст структурировался, включая токенизацию фразы с лемматизацией каждого токена.

**3.3. Выбор инструментов для проведения анализа данных и разработки ПО**

3.3.1. Использование корпуса данных

Использование корпуса данных осуществляется при помощи доступа к основной БД через клиент Dbeaver, позволяющего выполнять запросы на языке SQL:



3.3.2. Среда для проведения тематического моделирования и разработки проекта по созданию веб-приложения

Структура данной работы содержит два проекта, включающих исходные коды на языке Python:

1) Проект по получению статистической модели LDA с набором тематик, каждая из которых характеризует определенную компетенцию (набор знаний, умений и навыков), востребованных в вакансиях в области медицинской деятельности (FinalProject).

2) Проект по разработке веб-приложения, позволяющий получать рекомендации по требованиям работодателя на основании запроса по заработной плате (fp\_web).

Реализация проектов осуществлялась в локальной среде разработки Pycharm Community, версии 2022.3. Сохранение результатов и доступ к исходным кодам осуществляется через хостинг GitHub по следующей ссылке: <https://github.com/Sizovai/Final-work>

3.3.3. Инструменты для реализации проектов

1) Psycopg2 (версия 2.9.9) — адаптер БД PostgreSQL на языке Python, обеспечивающий поддержку связи с таблицами в БД.

2) Gensim (версия 4.3.2.) — библиотека для семантического анализа текстовых данных. Она содержит инструменты для обучения моделей без учителя, в частности, метод тематического моделирования, а также, позволяет проводить предобработку текста на естественном языке. В проекте использованы следующие возможности:

* реализация метода tf-idf
* создание n-грамм
* реализация модели LDA
* создание векторной модели токенов при помощи метода word2vec

3) Pymorphy2 (версия 0.9.1) — морфологический анализатор для русского языка. Использовался для приведения словаря к нормальной словоформе (токенизации и лемматизации).

4) NLTK (версия 3.8.1) — платформа для обработки естественного языка. Достоинством библиотеки является большое количество инструментов для предобработки текстов. В проекте инструмент использовался для сегментации фраз и импортировался метод, позволяющий определять и исключать стоп-слова.

5) Numpy (версия 1.26.2) — для работы с многомерными числовыми массивами данных, в частности, для получения числовых векторов токенов; подсчета вероятностей встречаемости слов в документах; получения метрик качества тематической модели.

6) Модуль re — использование регулярных выражений на языке Python для исключения лишних символов и пропусков в документах.

7) Flask (версия 3.0.0.) - фреймворк для создания веб-приложения.

8) tqdm — индикатор исполнения кода.

Все инсталлированные и импортируемые зависимости указаны в файлах requirements.txt, которые автоматически сконфигурированы средой Pycharm и являются частью документов проекта.

4**. Реализация проекта по тематическому моделированию неструктурированных частей вакансий**

Проект реализовался в два этапа.

4.1. Первый этап включал следующие шаги:

* обращение к сформированным таблицам в основной БД через создание соединения
* первичная обработка записей поля <body> (требования к работнику) из таблицы fp\_attribute, включая проведение токенизации и лемматизации слов, удаления спецсимволов, стоп-слов
* создание биграмм и триграмм токенов
* получение вероятностей встречаемости токенов в темах и документах при помощи метода tf-idf
* обучение модели LDA, включая получение двух сущностей, подающихся на вход модели (dictionary, corpus), выбор гиперпараметра модели (количества извлекаемых тем) и обучение модели
* получение метрик качества тематической модели (Perflexity, Coherence)

4.2. Второй этап включал следующие шаги:

* построение посредством LDA-модели тематических векторов для поля body каждой записи в выборке
* выделение в каждом тематическом векторе компоненты с наибольшим весом, которая соответствует доминантной теме
* формирование таблицы доминантных тем каждой записи

**5. Тематическое моделирование**

5.1. Исходные данные и формирование параметров для модели LDA

Обучение модели включает этап предварительной подготовки естественного языка, в рамках которого формировалась структура текста. Вначале текст сегментировался на отдельные фрагменты, которые сформировались в записи в таблице с данными. После этого фрагменты разбивались на отдельные слова-токены (токенизация) и приведение их к нормальной словоформе (лемматизация). Далее, была проведена автоматическая обработка записей (удаление цифр, спецсимволов, стоп-слов, приведение к нижнему регистру).

Анализ неструктурированных данных в выборке проводился на базе библиотеки Gensim, в которой реализуется метод LDA (Latent Dirichlet Allocation). Обычный перевод для этой очень распространенной статистической модели обозначается как «Латентное размещение Дирихле». Согласно документации, каждая запись (документ) рассматривается как набор случайных скрытых тем, где каждая тема определяется распределением слов. Распределение слов выстраивается на методе распределения Дирихле.

Каждая тема — это набор доминирующих ключевых слов. На вход модель LDA принимает две сущности — словарь и корпус. Если словарь создает уникальный идентификатор для каждого слова, то корпус является отображением частоты встречаемости слов в документе. Таким образом, корпус представляет собой массив целочисленных объектов для каждого слова, где первый является идентификатором слова в документе, второй отображает количество его упоминаний в документе.

При обучение модели LDA задавался только один гиперпараметр num\_topics, который задавался в модели двумя способами:

1) в ручную при обучении модели

2) перебором тем с шагом 10, начиная от 10.

def probe\_topics\_num(conn, selected\_topics\_num=None):  
 id2word = corpora.Dictionary.load('/home/irina/data/dictionary')  
 print('id2word = %d' % len(id2word))  
 corpus = load\_corpus(conn, 'fp\_filtered\_corpus')  
 cursor = conn.cursor()  
 cursor.execute("select id, tokens from fp\_attr\_tokens order by id")  
 data\_lemmatized = [row[1] for row in cursor]  
 print("read data\_lemmatized, size = %d" % len(data\_lemmatized))  
 if not selected\_topics\_num:  
 for topics\_num in range(10, 50, 10):  
 print('probe topics num %d' % topics\_num)  
 lda(conn, topics\_num, id2word, corpus, data\_lemmatized)  
 else:  
 lda(conn, selected\_topics\_num, id2word, corpus, data\_lemmatized)

Остальные гиперпараметры использовались по умолчанию.

После обучения модели вычислялись ее основные метрики.

Для данной модели широко используются две метрики качества:

1) Perflecity — сложность модели

2) Coherence — степень согласованности модели.

5.2. Процесс обучения модели

В процессе обучения модели использовалось несколько подходов.

1. Базовый подход содержал предварительную подготовку данных, создание входных сущностей и ручной выбор тем (100 тем). В результате реализации этого варианта получен набор тем с относительно низкими метриками качества модели и набором слов в каждой теме, трудно поддающиеся интерпретации.

2. Дополненный подход включал реализацию двух предварительных шагов для улучшения результатов моделирования.

* Вначале словарь был преобразован при помощи создания биграмм и триграмм, то есть, при помощи данного алгоритма, реализуемого в библиотеке Gensim часто встречающиеся в документе вместе токены были объединены.
* Кроме того, словари токенов и n-грамм были обработаны при помощи меры tf-idf, получив в итоге веса, показывающие, насколько часто одно слово или объединенные встретилось в одном документе относительно количества раз, когда оно/они встречались во всех остальных документах. Кроме того, из анализа исключались 5% термов, чьи веса были наиболее и наименее встречающимися в словаре.

|  |
| --- |
| def filter(conn):  id2word = corpora.Dictionary.load('/home/irina/data/dictionary')  corpus = fp\_lda.load\_corpus(conn, 'fp\_corpus')  tfidf = gensim.models.TfidfModel(corpus, id2word=id2word)  tfidf\_corpus = tfidf[corpus]  tf\_max = max([max(doc[1] for doc in corpus) for corpus in tfidf\_corpus])  tf\_min = min([min(doc[1] for doc in corpus) for corpus in tfidf\_corpus])  print(f'tfidf\_max: {tf\_max}, tfidf\_min: {tf\_min}')   doc\_ranges = [[doc[1] for doc in corpus] for corpus in tfidf\_corpus]  doc\_percentiles = []  for doc\_range in doc\_ranges:  percent\_pair = (np.percentile(doc\_range, 5), np.percentile(doc\_range, 95))  doc\_percentiles.append(percent\_pair)   print(doc\_percentiles[0]) |

После создания новых сущностей, модель LDA была обучена еще несколько раз, с использованием разных вариантов (например, один вариант мог включать: обучением со словарем биграмм и триграмм, но без tf-idf, с автоматическим подбором тем и т. д.). В итоговом варианте обученная модель с учетом полученных лучших метрик качества включала:

* предварительную обработку текстовых данных
* создание словаря токенов на основании добавления биграмм и триграмм
* включение меры tf-idf для вычисления наиболее встречающихся терм и удаление 5% самых редких и самых частых терм
* обучение модели со статичным указанием тем в количестве 50

5.3. Результаты обучения модели

Все полученные сущности записаны в базу данных:

|  |  |  |
| --- | --- | --- |
| Наименование таблицы | Назначение таблицы | Количество и наименование полей |
| fp\_attr\_tokens | Токенизированные документы | id записи  массив токенов |
| fp\_corpus | Корпус токенизированных документов | id  массив кортежей (идентификатор токена по словарю) — (частота встречаемости в документе) |
| fp\_topics | Список тем, выделенных LDA моделью | id темы  визуализация темы в виде 10 наиболее употребимых токенов |
| fp\_filtered\_corpus | Корпус токенизированных документов, отфильтрованных после применения меры tf-idf | id  массив кортежей (идентификатор токена по словарю) — (частота встречаемости в документе) |

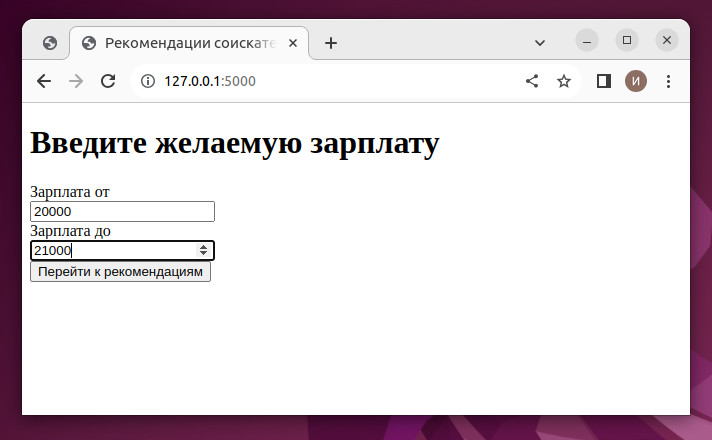
**6. Создание таблицы доминантных тем документов**

Вначале обученная и сохраненная модель LDA загружалась из файла, в который была сохранена после создания. Затем для каждой записи из таблицы отфильтрованного корпуса токенизированных документов у LDA модели вызывался метод get\_document\_topics, выдающий тематический вектор документа, состоящий из весов всех представленных в документе латентных тем. Среди последних определялась тема с наибольшим значением веса, которая считается доминантной темой в документе. Запись, содержащая id документа и номер доминантной темы, помещались в новую таблицу fp\_attr\_dominate\_topics.

**7. Разработка web-приложения**

Для разработки web-приложения был создан отдельный проект в среде Pycharm. С использованием фреймворка Flask создана главная страница, включающая окна ввода числовых данных по зарплате:

* зарплата от
* зарплата до



После этого у пользователя есть возможность получить рекомендации перейдя на соответствующую страницу приложения.

Рекомендации сформированы при помощи запроса SQL, обращающегося к таблице документов (from fp\_attribute fa), к которой присоединена таблица доминирующих тем (join fp\_attr\_dominate\_topic fadt on fa.id = fadt.id) и таблица списка тем (join fp\_topics ft on fadt.topic = ft.id). Условие запроса — попадание зарплаты вакансии, к которой относится каждый документ, во введенный пользователем диапазон (salary >= salary\_from and salary <= salary\_to).

Результаты запроса сгруппированы по темам (group by fadt.topic, ft.vis), агрегатной функцией count(\*) as cnt вычисляется количество документов в каждой теме.

Группы, соответствующие темам, сортировались в порядке убывания количества элементов в каждой группе (документов, для которых данная тема является доминантной) (order by cnt desc).

Из полученного списка пользователю выдавались первые 10 элементов (т. е. темы, отраженные в наибольшем количестве документов) (limit 10).

"""

select ft.vis, fadt.topic, count(\*) as cnt from fp\_attribute fa

join fp\_attr\_dominate\_topic fadt on fa.id = fadt.id

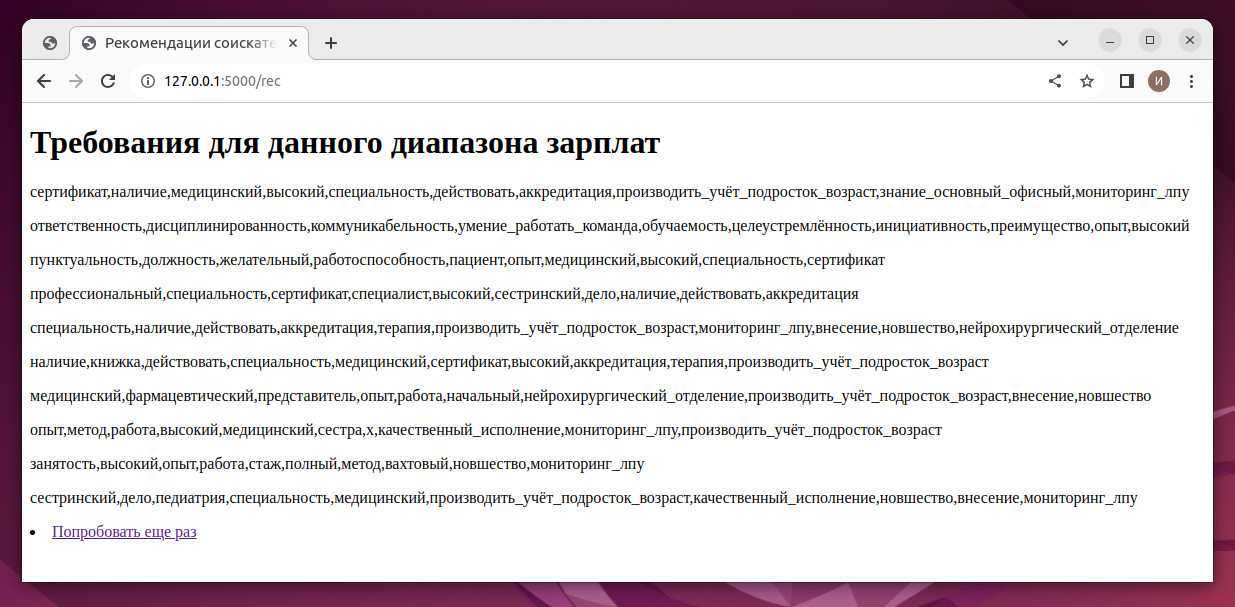
join fp\_topics ft on fadt.topic = ft.id

where salary >= %s and salary <= %s

group by fadt.topic, ft.vis order by cnt desc limit 10

""" % (salary\_from, salary\_to)

На этой же странице отображена ссылка, позволяющая пользователю вернуться на главную страницу.



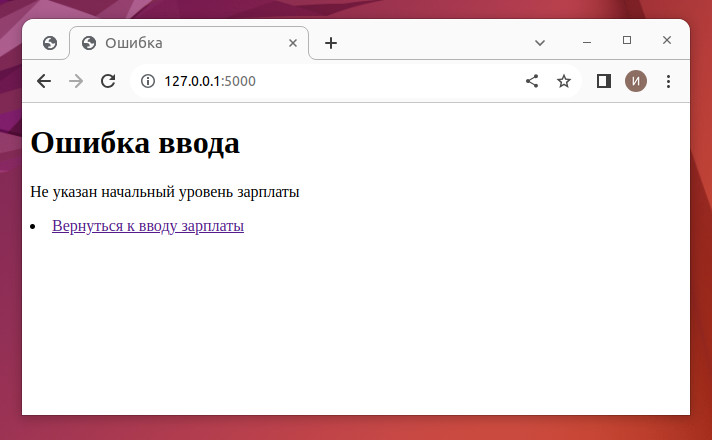
В приложении также обработаны некоторые сценарии ошибочного ввода данных по зарплате. Все использовано 4 сценария:

1) Не указан нижний уровень зарплаты

2) Не указан верхний уровень зарплаты

3) Нижний уровень зарплаты ниже минимальной, установленной на уровне в 18000 рублей.

4) Верхний уровень зарплаты ниже нижнего уровня зарплаты.



Окно ошибок предлагает пользователю вернуться к вводу данных.