ЕГОРОВА Д. К., ЗАВАРЮХИНА Ю. В.

ПРИМЕНЕНИЕ ИНСТРУМЕНТАРИЯ KNIME ANALYTICS PLATFORM ДЛЯ АНАЛИЗА СООТВЕТСТВИЯ РАБОЧИХ ПРОГРАММ УЧЕБНЫХ ДИСЦИПЛИН ТРЕБОВАНИЯМ РАБОТОДАТЕЛЕЙ

Аннотация. В статье рассматривается применение инструментария KNIME Analytics Platform для анализа текста в pdf-документах. Приведена реализация алгоритма анализа рабочих программ учебных дисциплин на соответствие требованиям работодателей.

Ключевые слова: KNIME Analytics Platform, алгоритм, workflow, text processing, узел, pdf-документ.

EGOROVA D. K., ZAVARYUKHINA YU. V.

APPLICATION OF KNIME ANALYTICS PLATFORM TOOLS TO ANALYZE THE COMPLIANCE OF SYLLABUSES WITH THE REQUIREMENTS OF EMPLOYERS

Abstract. The article considers the use of the KNIME Analytics Platform toolkit for text analysis in pdf documents. The implementation of the algorithm for analyzing the syllabuses of academic disciplines for compliance with the requirements of employers is described.

Keywords: KNIME Analytics Platform, algorithm, workflow, text processing, node, pdf-document.

Введение. Интеллектуальный анализ текста – это эффективный метод анализа данных, который можно использовать, для обработки текстовых документов, выявления ключевых терминов и тем, а также выявления скрытых отношений. Преимущества интеллектуального анализа текста очевидны, особенно в областях с большим количеством текстовых данных. Однако извлечение ценной информации, которая потенциально скрыта в текстовых данных, и явное отображение отношений между текстовыми данными, например, путем визуального представления ключевых атрибутов текста по отношению к определенным стандартам, может представлять собой довольно сложную задачу.

При решении подобного рода задач можно использовать low-code платформы визуальной разработки сценариев анализа данных. Согласно некоторым оценкам [3], к 2024 году порядка 70% приложений в мире будут разрабатываться на платформах low-code. Отличительной особенностью этих платформ, например, российских PolyAnalyst и Loginom, швейцарской KNIME, является то, что они не содержат встроенной системы анализа текстовой информации, процесс анализа следует «собирать» из своего рода узлов-nodes в которых используются библиотеки Python, R, JavaScript.

Далее будет приведено решение задачи анализа текстовой информации, извлеченной из интернет-источников и файлов pdf с использованием инструментария KNIME Analytics Platform.

Плагин Text Processing. Аналитическая платформа KNIME содержит плагин для обработки текста KNIME Text Processing [1], [2]. Плагин имеет открытый код и поддерживает многоуровневый процесс обработки текста – от чтения и синтаксического анализа через распознавание сущностей, фильтрации и манипуляции до подсчета количества слов, выделения ключевых понятий и, наконец, визуализации. Все это дает пользователю широкие возможности работы с текстом.

На рисунке 1 представлен репозиторий узлов обработки и анализа текстовой информации KNIME.

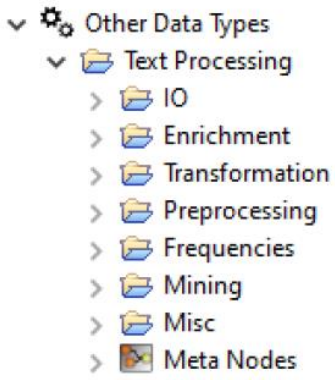

Рис. 1. Репозиторий узлов Text Processing.

Рассмотрим некоторые возможности указанного плагина для обработки текста.

Категория IO содержит узлы, позволяющие выполнить первоначальный анализ и чтение данных, а именно узлы синтаксического анализа, которые могут анализировать тексты из различных форматов, таких как DML, SDML, PubMed (формат XML), PDF, Word и др. Здесь текстовые документы представляются, в том числе, экземпляром класса Java Document, с использованием типов данных KNIME DocumentCell и DocumentValue.

Категория Enrichment содержит узлы, которые распознают стандартные именованные объекты (имена собственные, специфические обозначения, например, биомедицинские и др.), присваивая каждому распознанному именованному объекту значение тега. Тип тега представляет домен или тип маркера, а значение представляет конкретную характеристику в этом домене. Документ после процесса пометки называют «обогащенным», так как он содержит больше информации, чем первоначально.

Категория Transformation содержит узлы (вектор документа, вектор термина), позволяющие преобразовать текстовые данные в числовые. Эти узлы создают двоичное или числовое векторное представление для каждого термина.

Узлы категории Preprocessing используются на этапе глубокой предварительной обработки, заключающейся в очистке данных от стоп-слов, слов не имеющих содержания, знаков препинания и т.д. В итоге остаются только те термины, которые впоследствии используются для создания числового вектора или визуализируются в облаке тегов.

Узлы, осуществляющие вычисление терминальной частоты tf (относительной или абсолютной), обратной частоты документа idf, фильтрацию терминов с использованием показателя оценки релевантности, графическое представление документа для поиска ключевых слов, находятся в категории Frequency.

Еще более тонкая работа с текстом осуществляется с помощью узлов категорий Mining (Извлечение), Misc (Разное) и Meta Nodes (Мета узлы).

Реализация алгоритма. Использование узлов плагина для обработки текста KNIME Text Processing позволяет решить актуальную задачу анализа рабочих программ учебных дисциплин на соответствие требованиям работодателей. Требования работодателей можно получить, например, из описания вакансий, размещенных на соответствующих интернет площадках. Рабочие программы учебных дисциплин, в свою очередь, содержат перечень компетенций, знаний и умений, которыми должен обладать студент, в результате освоения той или иной дисциплины. Эти данные являются текстовыми. Применяя алгоритмы кластеризации и анализа текстовых данных определим, к какому запросу вакансий из области ИТ относится выбранная рабочая программа.

Для реализации задачи необходимо выполнить следующие шаги:

1. создать запросы для поиска данных о вакансиях;
2. присвоить каждой вакансии свой класс;
3. сформировать пакет слов для извлеченных вакансий и pdf-документа рабочей программы;
4. из пакетов слов построить векторы документов;
5. применить наивный байесовский классификатор для сопоставления рабочей программы и запроса.

Поиск данных о вакансиях осуществлялся на сайте https://hh.ru. Для реализации алгоритма были выбраны следующие вакансии в г.о. Саранск: мобильный разработчик, разработчик игр, веб-разработчик.

Шаг 1. Создаем в KNIME Analytics Platform рабочий процесс (Workflow). Считываем данные при помощи узла GET Request (рис. 2). Первоначально работаем с вакансиями. Для

этого создаем три одинаковые ветки, так как планируется извлечение трех разных типов вакансий.

Рис. 2. Узел GET Request.

В настройках конфигурации узла добавляем запрос о вакансиях написанный на Python, имеющий, для каждой ветки следующий вид:

https://api.hh.ru/vacancies?area=63&per_page=100&only_with_salary=true&text=вебANDразработчик&search_field=name

https://api.hh.ru/vacancies?area=63&per_page=100&only_with_salary=true&text=разработчикANDигр&search_field=name

https://api.hh.ru/vacancies?area=63&per_page=100&only_with_salary=true&text=мобильныйANDразработчик&search_field=name

Данные запросы реализуют извлечение по сто вакансий по г.о. Саранск, названия которых содержат слова «веб-разработчик» (первый запрос), «разработчик игр» (второй запрос), «мрбильный разработчик» (третий запрос) с указанными размерами заработной платы. При необходимости данные параметров извлечения информации можно менять, руководствуясь документацией HeadHunter API.

Далее выполняем узел и непосредственно извлекаем в каждой ветке по 100 вакансий, воспользлвавшись узлом JSON Path. Соединяем его с узлом GET Request (Рис. 3).

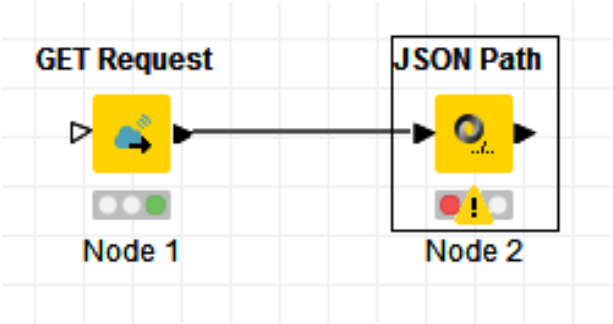

Рис. 3. Соединение узла JSON Path.

Переходим в конфигурацию узла JSON Path. Во вкладке Setting выбираем параметр идентификатора записи (id) и с помощью правой кнопки мыши добавляем путь (Add JSONPath). В строках содержащих название вакансии (name) и параметры заработной платы (currency – валюта, gross – сумма до уплаты подоходного налога, to и from в salary – размер заработной платы от/до, responsibility в snippet – требования к кандидату, name в schedule – рабочий режим) меняем значение «0» после ['items'] на «*» (Рис. 4), для того, чтобы иметь возможность работать со всеми вакансиями. В противном случае из всей базы будет извлечена только одна вакансия.

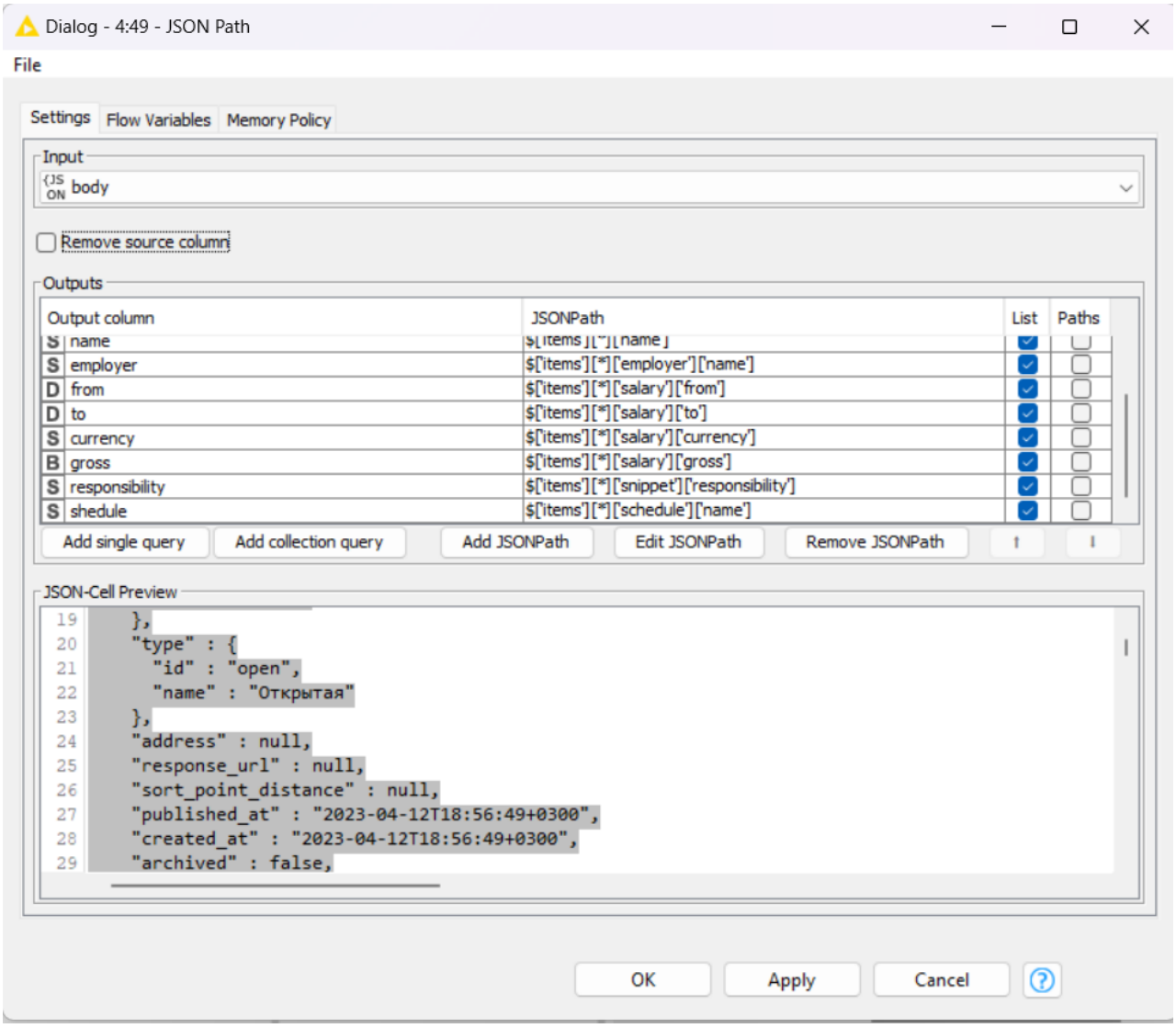

Рис. 4. Настройка конфигурации узла JSON Path.

Приводим узел в действие. Получаем список запрашиваемых данных, который необходимо разделить на отдельные документы (вакансии). Для этого воспользуемся узлом

Ungroup. Данный узел создает список строк с извлеченными вакансиями и их параметрами. Соединяем узлы JSON Path и Ungroup, затем запускаем их выполнение.

Шаг 2. Далее используем узел Math Formula для присвоения номера класса ветки. После запуска процесса получаем список с добавленым столбцом номера класса в таблице данных каждой ветки. Затем дважды используем узел Concatenate для объединения таблиц полученных на разных ветках.Таблица на входе 0 задается как первая входная таблица (верхний входной порт), таблица на входе 1 является второй таблицей, соответственно. Получаем одну общую таблицу, содержащую весь перечень извлеченных вакансий с присвоенными им классами.

Далее используем узел Column Filter, он необходим для фильтрации данных, так как для решения задачи необходимо оставить класс ваканссии (class) и перечень требований к кандидату (responsibility) (рис. 5).

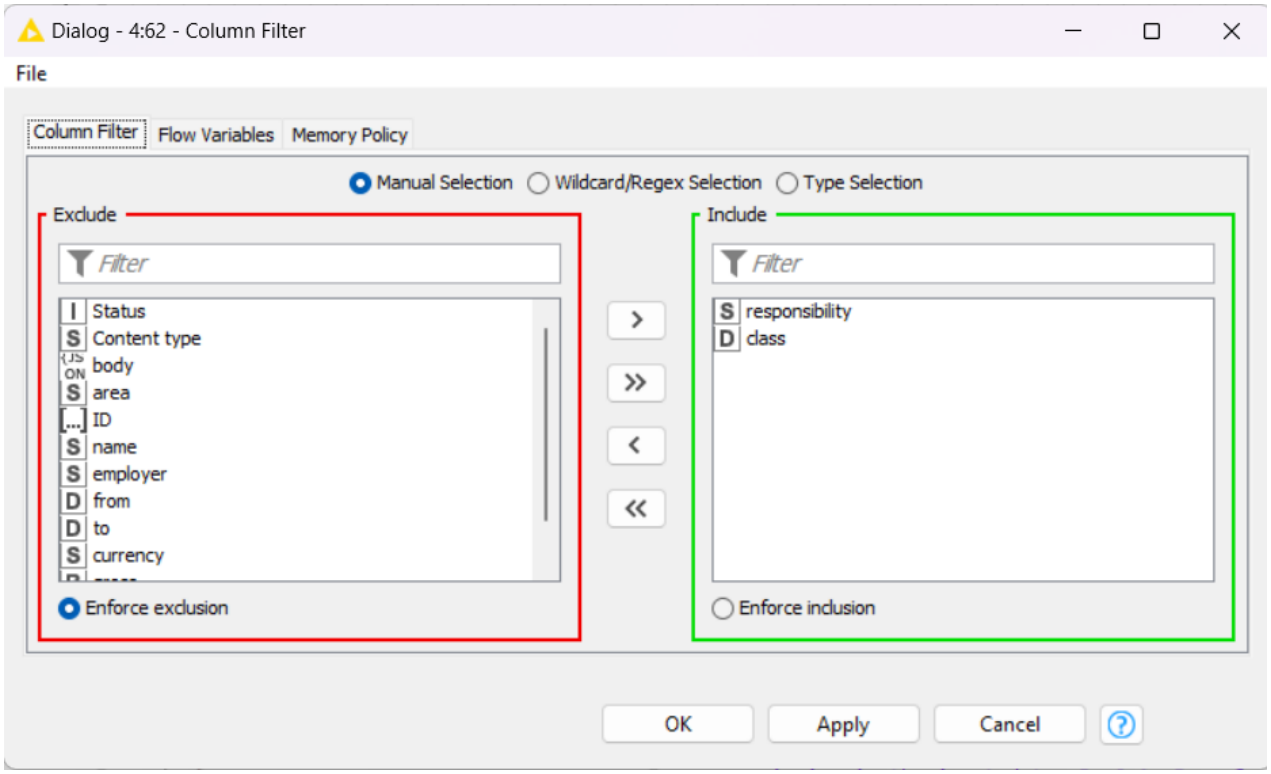

Рис. 5. Настройки узла Column Filter.

Используем узел Missing Value для заполнения недостающих значений, если таковые имеют место быть. Затем необходимо преобразовать указанные строки в документы. Используем узел Strings To Document (рис. 6).

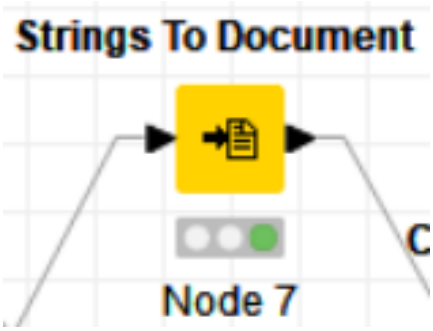

Рис. 6. Узел Strings To Document.

Шаг 3. Переходим к формированию пакета слов для вакансий. Используем узел Bag Of Words Creator. Пакет слов состоит из столбцов, содержащих термины, встречающиеся в соответствующем документе (вакансии). Все столбцы, связанные с терминами, такие как столбец документа, можно выбрать в диалоговом окне конфигурации узла. В нашем случае в конфигурации выбираем параметры «Document», «class» и «responsobility» (рис. 7).

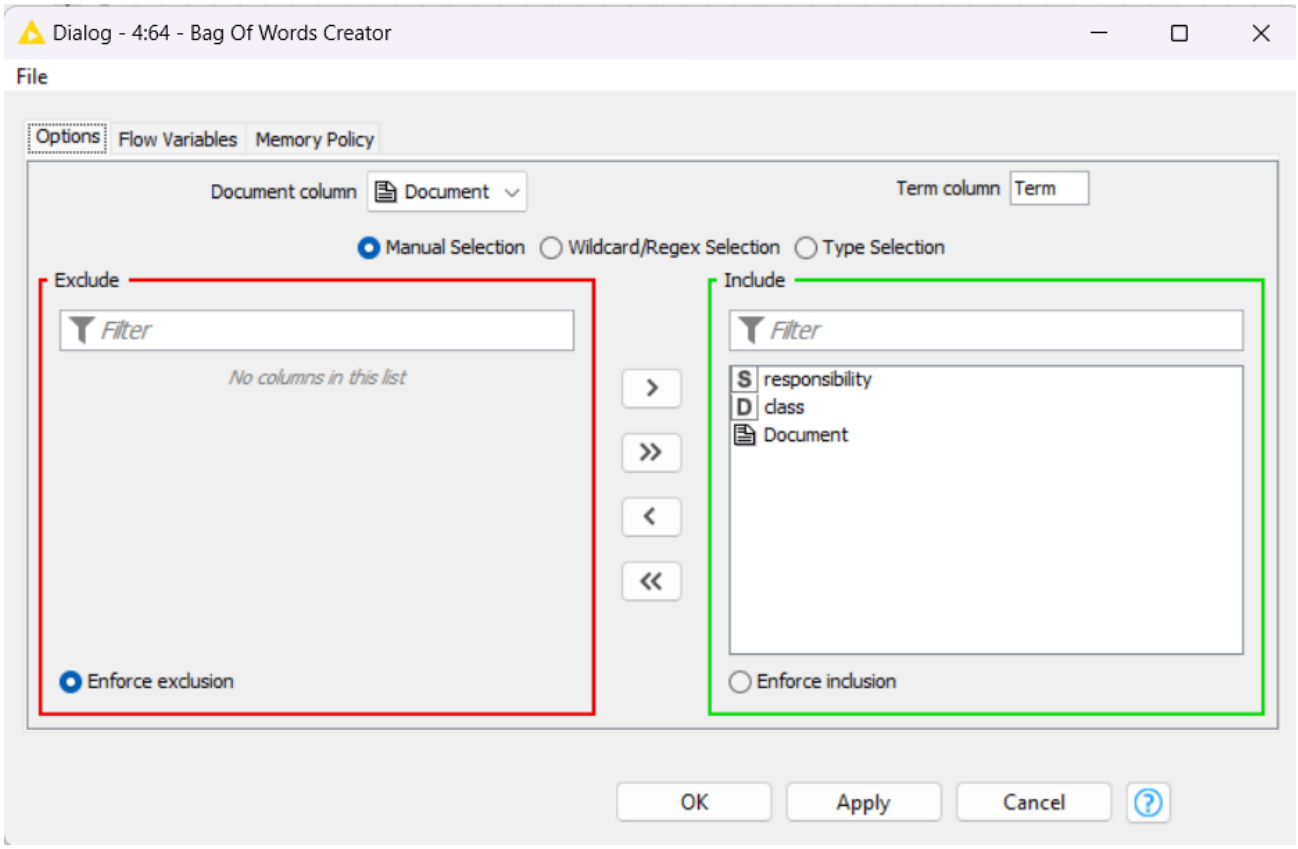

Рис. 7. Настройки узла Bag Of Words Creator.

Шаг 4. Создаем вектор документа для каждой вакансии с помощью узла Document Vector. Этот узел создает вектор документа, представляющего его в пространстве терминов.

Размерность векторов будет равна количеству различных термов полученных в пакете слов. В конфигурации данного узла выбираем параметр «Document» (рис. 8).

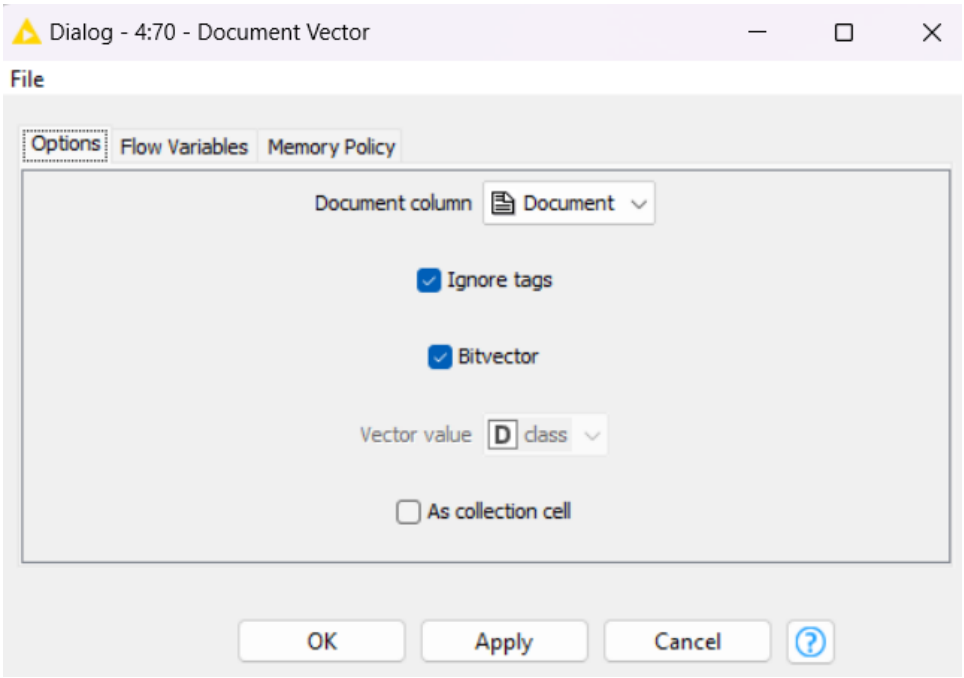

Рис. 8. Настройки узла Document Vector.

Также необходимо выполнить фильтрацию столбцов только по классу (узл Column Filter) и преобразование чисел в строки (узел Number to string) для формирования таблицы и их объединение (узел Column Appender) .

Параллельно создаем ветку для загрузки рабочей программы учебной дисциплины, например, «Работа с удаленными базами данных» и реализации описанных выше шагов 3 и 4. Здесь используем узел PDF Parser, в настройках конфигурации которого выбираем путь к файлу. Затем используем узел RegEx Filter для удаления ненужных символов, прописывая их в настройках конфигурации узла (рис. 9).

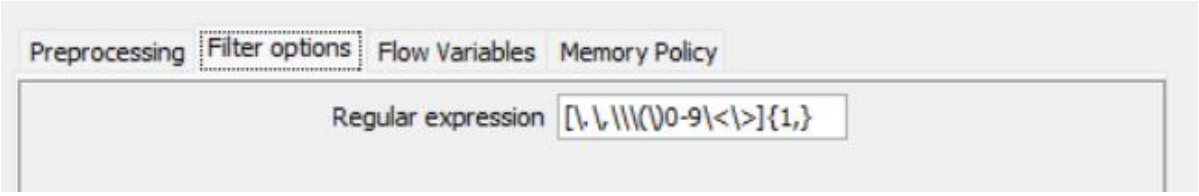

Рис. 9. Настройки узла RegEx Filter.

Шаг 5. Далее используем узел Naive Bayes Learner для «обучения» байесовской модели предсказывающей класс для каждой вакансии на основе соответствующего вектора

документа (вакансии). В результате получили 3 класса соответствующие трем выбранным типам вакансий (таб. 1).

Таблица 1

Соответствие класса выбранному типу вакансии

Класс	Соответсвующий тип ваканси на hh.ru
1	Мобильный разработчик
2	Веб-разработчик
3	Разработчик игр

Узел Naive Bayes Learner является входным для узла Naive Bayes Predictor, который прогнозирует класс документа содержащего рабочую программу на основе обученной байесовской модели.

В завершении соединяем две ветки реализующие обработку вакансий и текста рабочей программы в узле Naive Bayes Predictor.

Результатом выполнения узла Naive Bayes Predictor является тип вакансий с сайта hh.ru, которому соответствует загруженная программа учебной дисциплины. Открывая вкладку The classified data (с анг. – секретные данные) конфигурации узла Naive Bayes Predictor в последнем столбце Prediction (с анг. – прогноз) получили класс 3 (рис. 10), что соответствует запросу работодателей по типу вакансий «Разработчик игр» (табл. 1).

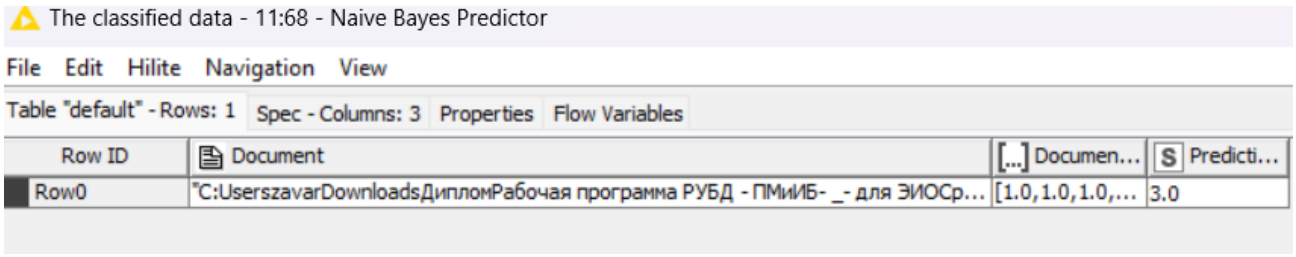

Рис. 10. Результат работы алгоритма.

Предлагаемый алгоритм, состоящий из 28 узлов (рис. 11), разработан согласно модели low-code и не является desktop-приложением, вследствие чего может быть полезен узкому кругу специалистов, занимающихся анализом текстовой информации, получаемой из различных источников.

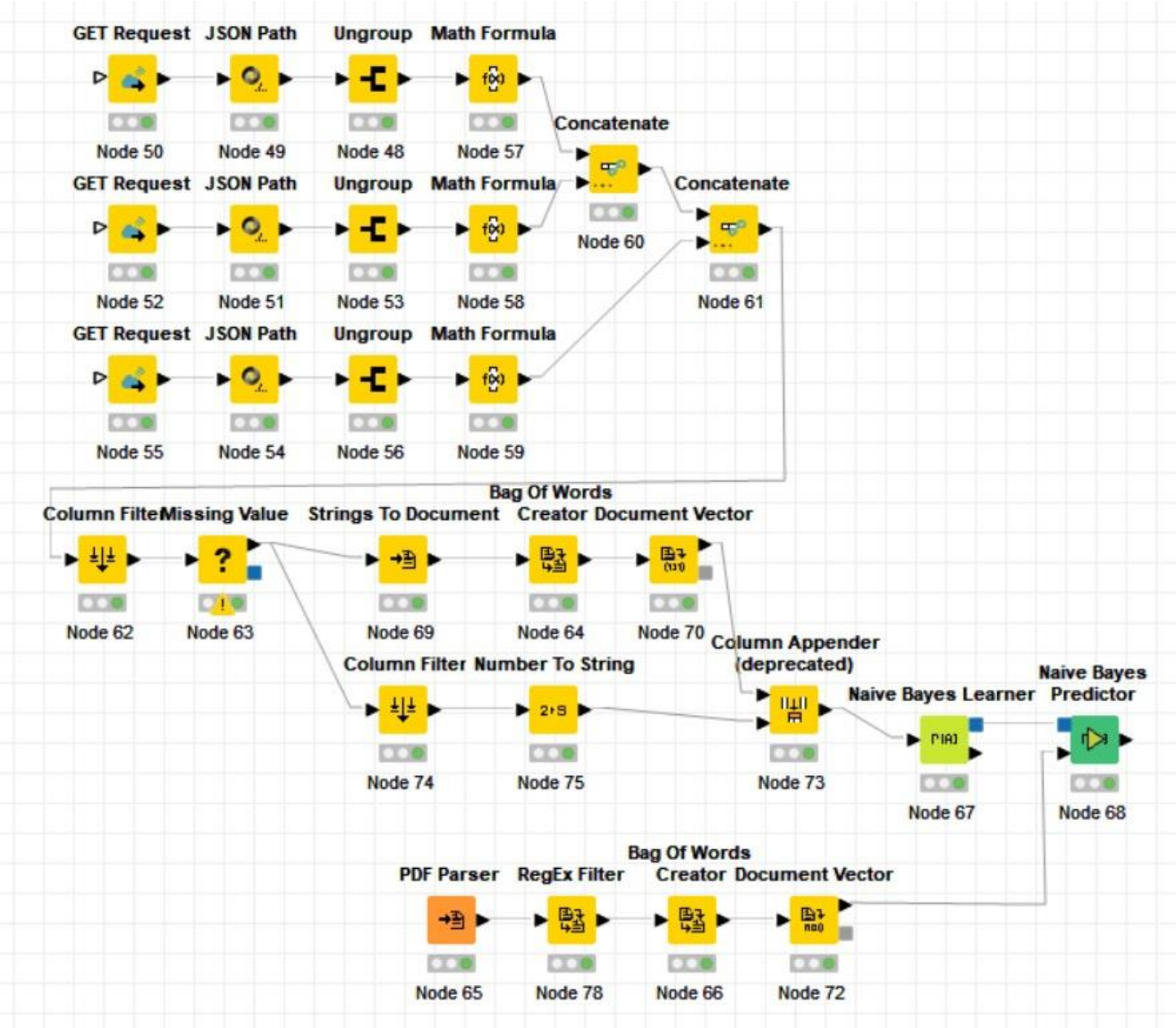

Рис. 11. Итоговый рабочий процесс.

СПИСОК ЛИТЕРАТУРЫ

1. From data collection to text mining [Электронный ресурс]. – Режим доступа: https://www.knime.com/blog/data-collection-to-text-mining (дата обращения 01.09.2023).
2. Knime Analytics Platform [Электронный ресурс]. – Режим доступа: https://www.knime.com/ (дата обращения 01.09.2023).
3. Low-code в России: Быстрый рост без иностранных конкурентов [Электронный ресурс]. – Режим доступа: https://startpack.ru/articles/low-code-v-rossii (дата обращения 30.10.2023).