# АНАЛИТИЧЕСКИЕ РЕШЕНИЯ ПО УГЛУБЛЕННОМУ АНАЛИЗУ ГРАФОВЫХ БД

*Зорко П. А., Кулевич А. О.*

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Пилецкий И. И. – канд. физ.-мат. наук, доцент*

Приводится описание аналитических решений по углубленному анализу графической БД, с целью глубокого анализа данных Web-сайтов в некоторой научной области. Описываются принятые решения по применению ML, демонстрируются, результаты работы компонента получение данных с web-сайта.

Графовые технологии позволяют преобразовать представление Web-сайта в графовую БД со свойствами. В то же время, графовая БД со свойствами может быть использована для анализа свойств тематического сайта с помощью ***графа знаний***.

На сайте Medium [1] имеется множество статей и публикаций на различную тематику. Данные этого сайта используются для демонстрации аналитических решений по углублённому анализу графовых БД. С помощью средств графовой БД Neo4j [2] встроенных плагинов APOC и Neosemantics, сериализуются данные статей в RDF и получается тематическая графовая БД. Общее представление полученной БД показано на рисунке 1, левая часть.

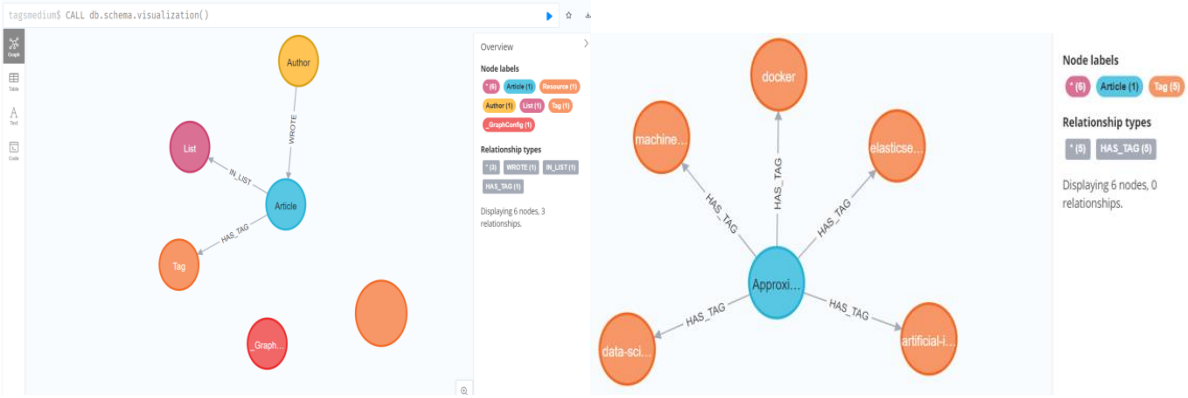


Рисунок 1 – Графовая базы данных и граф знаний

Из полученной базы данных можно получать различные ***графы знаний***. К примеру, с помощью следующего запроса MATCH (ar:Article)-[:HAS_TAG]->(t:Tag)
WHERE ar.title = “Approximate Nearest Neighbors on Elastic Search with Docker” RETURN ar, t можно получить граф знаний, состоящий из статьи и ключевых слов, которые относятся к этой статье. Результат запроса представлен на рисунке 1, правая часть.

Проведем анализ полученного графа знаний. Анализировать данные будем с помощью комбинации возможностей языка программирования Python и Cypher [3].

Посчитаем ***embedding узлов в графе***. Сделать это нам позволяют две функции Node2Vec [4] и GraphSAGE [5]. Будем использовать алгоритм GraphSAG, т.к. вместо обучения отдельных включений для каждого узла алгоритм изучает функцию, которая генерирует включения путем выборки и агрегирования признаков из локальной окрестности узла, что позволяет нам не пересчитывать включения для всего графа при добавлении нового узла, а лишь применить функцию к новому узлу.

Чтобы упростить процесс обучения модели используем алгоритм моночастичной проекции. Идея проекции состоит в том, чтобы взять граф с двумя типами узлов и вывести из него граф с одним типом узлов. Библиотека Neo4j предоставляет алгоритм построения моночастичной проекции с помощью алгоритма Node Similarity [6] из библиотеки GDS.

***Сходство узлов*** вычисляет парные сходства на основе либо показателя Жаккарда, также известного как оценка сходства Жаккарда, либо коэффициента перекрытия, также известного как коэффициент Шимкевича–Симпсона.

После применения алгоритма Node Similarity мы получим граф имеющий следующее представление (рисунок 2):

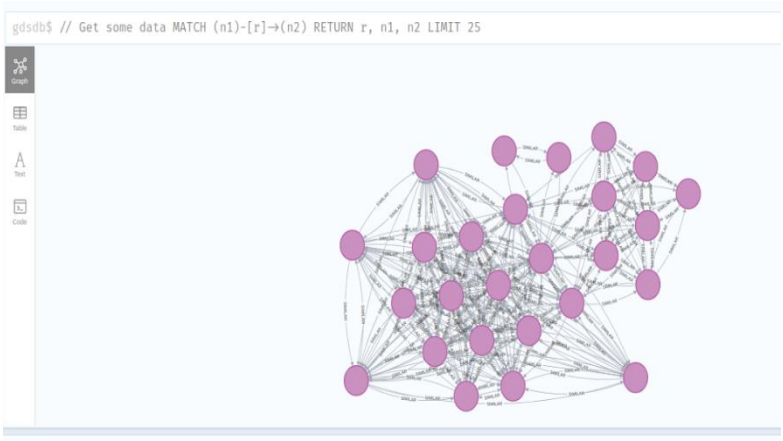


Рисунок 2 – Граф знаний после применения алгоритма Node Similarity

Обучим нашу модель GraphSAGE на выбранном подграфе и перейдем к ***ML алгоритму классификации.***

С помощью разработанной нами функции, которая принимает на вход данные и столбцы и применяет к ним алгоритм классификации с несколькими метками, а затем возвращает наиболее эффективно построенную модель, определим теги для статей, число тегов у которых ровняется нулю.

После запуска алгоритма получим следующий результат: Introduction to Data Mesh adoption in Adidas – motivation and takeaways –> ***[data]***; 3 Things to Do When You Feel Ruled by Time –> ***[productivity];*** A Data Science project start to finish -> ***[coding, programming, python, python3, software-development];*** Time series anomaly detection – in the era of deep learning -> ***[data-science, machine-learning]***; How to Optimize Your Apache Spark Application with Partitions -> ***[spark]***; Rule Execution with SHACL -> ***[knowledge-graph]***.

Проанализировав полученные данные, можно заметить, что теги действительно соответствуют тематике статей. Разработанный алгоритм можно использовать в дальнейшем для предсказания тематик, выделения ключевых слов и формирования списков с похожими статьями.

Для подсчета популярности статей будем использовать ***ML алгоритм PageRank***:

CALL gds.pageRank.write('ArticlesGraph', {maxIterations: 20, dampingFactor: 0.85,writeProperty: 'pagerank'}) YIELD nodePropertiesWritten, ranIterations

Гистограмма популярности статей на основе вычисленного PageRank представлена на рисунке 3.

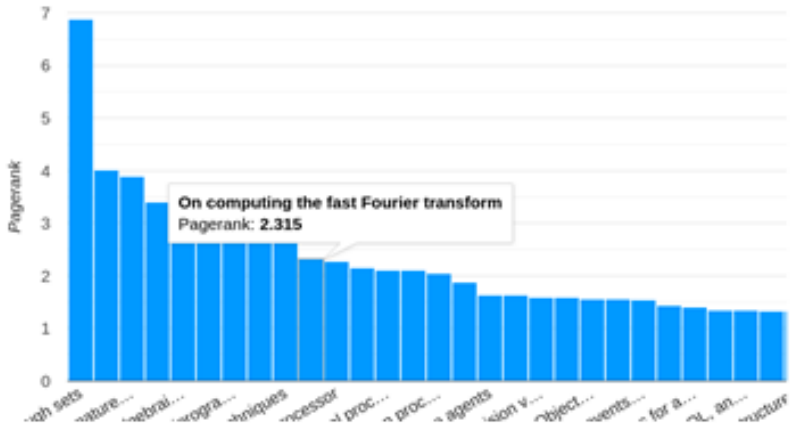


Рисунок 3 – Гистограмма популярности статей на основе вычисленного PageRank

Результатом научной работы является комплексная методологии и ее применение для глубокого анализа данных Web-сайтов.

***Список использованных источников:***

*1. Medium [Электронный ресурс] / Режим доступа: https://medium.com Дата доступа: 10.03.23.*

*2. Neo4j [Электронный ресурс] / Режим доступа: https://neo4j.com/labs/neosemantics/ Дата доступа: 10.03.23*

*3. Pypi [Электронный ресурс] / Режим доступа: https://pypi.org/project/graphdatascience/ Дата доступа: 12.03.23*

*4. Neo4j. Node2Vec [Электронный ресурс] / Режим доступа:https://neo4j.com/docs/graph-data-science/current/machine-learning/node-embeddings/node2vec/ Дата доступа: 18.03.23*

*5. Neo4j. GraphSAGE [Электронный ресурс] / Режим доступа: https://neo4j.com/docs/graph-data-science/current/machine-learning/node-embeddings/graph-sage/ Дата доступа: 18.03.23*

*6. Neo4j. Node Similarity [Электронный ресурс] / Режим доступа https://neo4j.com/docs/graph-data-science/current/algorithms/node-similarity/ Дата доступа: 19.03.23*