ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(ФИНАНСОВЫЙ УНИВЕРСИТЕТ)**

Департамент анализа данных и машинного обучения

**Дисциплина: «Теория сложных сетей в экономике»**

Направление подготовки: «Прикладная математика и информатика»

Профиль: «Анализ данных и принятие решений в экономике и финансах»

Факультет информационных технологий и анализа больших данных

Форма обучения очная

Учебный 2021/2022 год, 6 семестр

Курсовая работа

на тему:

**«Создание и анализ графового набора данных по заданной предметной области на основе крупномасштабной базы знаний»**

Изображение выглядит как текст

Автоматически созданное описаниеДопущена к защите

31 мая 2022 г.

Выполнила:

студентка группы ПМ19-2

Гераськина Н.С.

Руководитель:

к.т.н., доцент, Петросов Д.А.

Изображение выглядит как силуэт

Автоматически созданное описание

Москва 2022

Содержание

[ВВЕДЕНИЕ 2](#_Toc104646728)

[Подробное разъяснение темы 2](#_Toc104646729)

[Актуальность 3](#_Toc104646730)

[1. Постановка задачи 4](#_Toc104646731)

[2. Теоретическая справка по созданию и анализу графа 5](#_Toc104646732)

[2.1. Базовые понятия из теории графов 5](#_Toc104646733)

[2.2. Базовые понятия из теории сложных сетей 5](#_Toc104646734)

[2.3. Графы знаний 5](#_Toc104646735)

[2.4. Характеристики графов 6](#_Toc104646736)

[2.5. Выявление сообществ 8](#_Toc104646737)

[3. Выбор инструментальных средств программной реализации 10](#_Toc104646738)

[4. Описание программной реализации построения графа знаний 11](#_Toc104646739)

[5. Анализ полученного графа 12](#_Toc104646740)

[5.1. Основные характеристики графа 12](#_Toc104646741)

[5.2. Выделение сообществ 16](#_Toc104646742)

[ЗАКЛЮЧЕНИЕ 23](#_Toc104646743)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ 24](#_Toc104646744)

[ПРИЛОЖЕНИЕ 26](#_Toc104646745)

[Характеристики компьютера и время выполнения программ 26](#_Toc104646746)

[Код программ 27](#_Toc104646747)

[Список файлов 38](#_Toc104646748)

# **ВВЕДЕНИЕ**

## **Подробное разъяснение темы**

В данной работе предлагается формализовать определенную экспертную систему за счет использования графов для создания моделей и экспертных систем на основе предметных областей знаний.

В современном мире существует множество математических подходов, которые помогают объективно описать определенную предметную область. Самыми распространенными методами являются:

* графоаналитический подход;
* функциональный подход;
* графический подход и т. д.

В данной работе предложено использование графоаналитического подхода для решения поставленной задачи.

Данный подход позволяет создать модели базы знаний для различных предметных областей и обладает рядом преимуществ, связанных с программной реализацией.

База знаний – это база данных, содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. [4]

В исследовании, для программной реализации, предлагается использовать интерпретируемый язык программирования Python, который позволит смоделировать и провести анализ без привязки к платным платформам. Для оптимальной работы кода с применением графоаналитического подхода предлагается использование таких библиотек Python, как: NetworkX; Pandas; Pyvis и другие. [2-3]

Данный подход позволяет оптимизировать не только процесс программной реализации, но и непосредственно исполнение программного кода на вычислительном устройстве пользователя.

Применение данного подхода позволяет подробно изучить предметную область при помощи формализованной модели, которая может масштабироваться на более сложные системы для решения реальных задач. К реальной задаче можно отнести решения задачи сокращения расходов на производство/транспортировку товаров и услуг из выбранной предметной области.

## **Актуальность**

Данное исследование как никогда актуально ввиду появления быстроразвивающихся систем, работу которых необходимо оптимизировать, чтобы они приносили большие доходы владельцам и не нагружали имеющиеся в запасе ресурсы. Одно из основных особенностей современного мира – огромное количество необработанной информации (Big Data) во всех сферах жизнедеятельности человека. Поэтому исследование данных сложных систем на основе исторических данных позволит обнаружить общие закономерности, которые помогут заранее предотвращать роковые ошибки в работе, приводящие к снижению производительности и доходности.

Также стоит отметить, что база знаний — важный компонент интеллектуальной системы. Наиболее известный класс таких программ — это экспертные системы. Они предназначены для поиска способов решения проблем из некоторой предметной области, основываясь на записях базы знаний и на пользовательском описании ситуации.

# **Постановка задачи**

Для стартапов и маркетологов в современном мире затруднительно найти потенциальных рекламодателей или заказчиков. Рынок услуг разнообразен и изменчив, поэтому необходимо быть в курсе актуальных изменений. В этом может быть полезна информация из Википедии.

Википедия – общедоступный ресурс, который может обновляться как официальными источниками, так и простыми пользователями. Именно эта открытость для всех позволяет Википедии обновляться быстрее многих информационных ресурсов. Для своевременных эффективных маркетинговых действий будут информативными даже неподтвержденные факты, которые позволят провести успешную рекламную кампанию.

Однако в Википедии есть много «информационного шума», который может замедлить процесс своевременного принятия решений. Необходимо немало профессионализма, чтобы быстро выделять основные сущности из объемных статей. В этом может помощь обработка естественного языка (Natural Language Processing).

Итак, если проанализировать данный популярный ресурс и извлечь категории статей или другие их признаки, можно составить граф знаний (Knowledge graph), который поможет выявить неочевидные связи между современными компаниями и особенностями их коммерческой деятельности.

В данной курсовой работе были выбраны в качестве анализируемой предметной области современные бренды и транснациональные компании.

# **Теоретическая справка по созданию и анализу графа**

## **Базовые понятия из теории графов**

При выполнении поставленной задачи необходимо ознакомиться с понятием неориентированного графа:

«Неориентированным графомназывается пара, где— множество вершин, а— множество рёбер» [7]

В работе будет строиться мультиграф то есть неориентированный граф, в котором допускается построение параллельных ребер (рёбер, имеющих те же самые конечные вершины) [7]. Данный выбор обоснован тем, что в сети связи являются двунаправленными, а значит смысл показа данных связей пропадает.

## **Базовые понятия из теории сложных сетей**

Построенный в данной работе граф, по сути, представляет из себя сложную систему, то есть систему, состоящую из множества компонентов, которые могут взаимодействовать друг с другом.

Сложные системы обладают следующими свойствами:

* Нелинейность (nonlinearity);
* Эмерджентность (emergence) - свойства характерные для системы в целом и не характерные для ее частей;
* Предрасположенность к самоорганизации - упорядочения элементов одного уровня в системе за счёт внутренних факторов, без специфического внешнего воздействия;
* Адаптивность (adaptation) - способность меняться и учиться под воздействием окружающей среды;
* Наличие петлей обратной связи (feedback loops).

## **Графы знаний**

В данной курсовой работе необходимо построить и проанализировать так называемый граф знаний (Knowledge graph). Граф знаний является семантической сетью и представляет собой сеть сущностей из реального мира [5]. Данное математическое представление окружающего мира является сетью объектов, событий, ситуаций или понятий, и иллюстрирует взаимосвязь между ними. Эта информация обычно хранится в базе знаний. База знаний представляет собой смысловые отношения между, объектами реального мира, понятиями.

Одними из самых известных графов знаний являются DBpedia, состоящая из данных информационных блоков Википедии, и Wikidata, фокусирующаяся на вторичных и третичных объектах. Google Knowledge Graph представлен через страницы результатов поисковой системы Google (SERP), собирая информацию на основе того, что люди ищут в поисковике. Этот граф знаний состоит из более чем 500 миллионов объектов, полученных из Freebase, Wikipedia, CIA World Factbook и других источников [1].

## **Характеристики графов**

Анализ в данной курсовой будет осуществляться по нескольким направлениям. Во-первых, будет построено распределение степеней узлов:

Величинуможно интерпретировать как вероятность того, что случайно выбранный узел сети имеет степень. Рассмотрениевместопозволяет сравнивать распределения степеней узлов для сетей разных размеров.

Во-вторых, будет подсчитана плотность сети:

Также будут анализированы различные меры центральности.

Нормированная центральность по степени (degree centrality) – центральность узла равна количеству связей у узла. Показывает, кто является наиболее "популярным" в сети.

Нормированная центральность по близости (closeness centrality) определяется как величина, обратная сумме расстояний от узладо всех остальных узлов. Характеристика показывает, насколько быстро распространяется информация в сети от данного узла к остальным.

Нормированная центральность по посредничеству (betweenness centrality) рассчитывается следующим образом: для всех пар узлов (не включая рассматриваемый) вычисляется доля кратчайших путей, проходящих через рассматриваемый узел, затем эти доли суммируются. Это мера контроля, определяющего важность того или иного узла как передаточного звена.

## **Выявление сообществ**

Для детального рассмотрения полученного графа и выявления дополнительных смысловых связей будет применен алгоритм выявлений сообществ на полученном графе. Сообщества обладают следующими свойствами: взаимностью связей, компактностью (малое расстояние между членами группы), разделенностью (высокой плотностью связей внутри группы по сравнению со связями с внешними членами). Выявленные сообщества могут использоваться для:

* Понимания взаимосвязей между узлами;
* Разметки сетей для визуализации и навигации по большим (и гигантским) сетям;
* Предсказание связей;
* Основа для других задач, таких как data-mining в сетях.

Исходя из того, что сообщества в сетях это: множества узлов с большим количеством внутренних связей, небольшим количеством внешних связей (с остальными узлами сети), будет применен Лувенский алгоритм - эвристический метод жадной оптимизации модулярности. Алгоритм устроен так, что рассчитываемая модулярность относится к исходной топологии сети: выполняя операции над суперузлами, рассчитываются изменения модулярности для разбиения исходной сети. На некотором уровне модулярность более не возрастает и алгоритм останавливается. Все исходные вершины, которые входят в финальный суперузел, принадлежат одному сообществу.

Лувенский алгоритм является очень быстрым, сложность имеет порядок. На реальных сетях алгоритм демонстрирует качество (величину достигнутой модулярности), лучшее чем все ранее созданные жадные алгоритмы [6].

Мера модулярности для разбиения сети на сообщества считается следующим образом

# **Выбор инструментальных средств программной реализации**

Все вычисления будут проводиться на языке Python. В качестве крупномасштабной базы знаний в данной курсовой будет взята Википедия. Подключение к данной энциклопедии будет осуществляться при помощи библиотеки Wikipedia-api. Данная библиотека поддерживает извлечение текстов, разделов, ссылок, категорий, переводов со страниц Википедии [3].

Также полученные данные будут храниться и обрабатываться при помощи таких библиотек Python как Pandas, Numpy, NetworkX. Визуализация результатов нетрудозатратных вычислений будет проводиться при помощи Matplotlib. Визуализация графов будет осуществляться при помощи библиотеки Pyvis. Данная библиотека способна интерактивно визуализировать сложные сети, в отличие от NetworkX, так как была разработана как оболочка вокруг популярной библиотеки Javascript visJS. Также при анализе полученного графа будет проводиться выявление сообществ при помощи модуля python-louvain, в котором реализовать Лувенский алгоритм [2].

# **Описание программной реализации построения графа знаний**

Для выгрузки информации было совершено подключение к Wikipedia API. Для получения вершин графа, между которыми будут установлены связи, были выбраны названия статей и их категории. Таким образом граф состоит из связей «Статья N имеет категорию M» и «категория M относится к статье N».

После выполнения кода «Код 01. Извлечение информации из Википедии.ipynb» был получен массив данных по теме «brand». Далее его необходимо обработать, что было сделано по средствам библиотеки Pandas. В итоге при помощи библиотеки NetworkX по полученной таблице данных был построен граф. Далее будет описан анализ полученного графового набора.

# **Анализ полученного графа**

## **Основные характеристики графа**

В результате работы программы «тут название» был получен граф с 3067 вершинами и 3985 ребрами. Плотность сети составила 0.0008. Это является низким показателем, что свидетельствует о потенциально большом количестве «изолированных» подграфов и что число рёбер не близко к максимально возможному у полного графа с таким же числом вершин, то есть граф близок к разряженному.

Плотность распределения степеней вершин в линейных координатах можно увидеть на Рисунке 1.

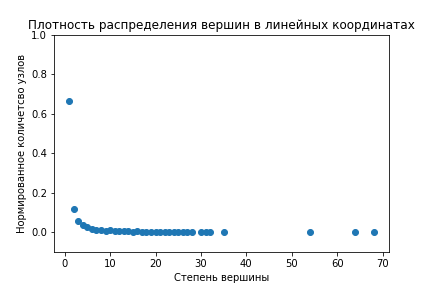
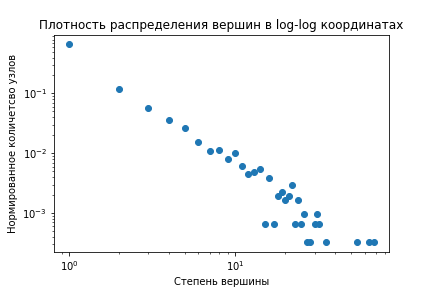
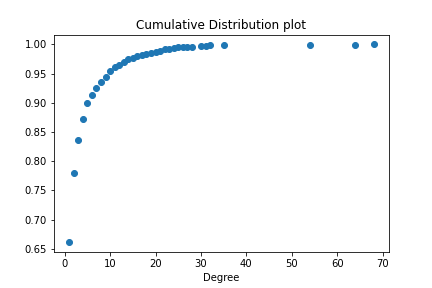


Рисунок 1 – Плотность распределения степеней вершин в линейных координатах

Плотность распределения степеней вершин в log-log координатах можно увидеть на Рисунке 2.

 Рисунок 2 – Плотность распределения степеней вершин в log-log координатах

В свою очередь, распределение степеней вершин можно увидеть на Рисунке 3. В графе приблизительно 70% вершин имеет степень 1–2, оставшиеся 30% имеют степени 3–70. Это свидетельствует о том, что в сети есть несколько «центральных» узлов.

Рисунок 3 – Распределение степеней вершин

Наблюдение о наличии популярных узлов подтверждается мерами центральности (Таблица 1–3).

Таблица 1 – Лучшие вершины по нормированная центральность по степени

|  |  |
| --- | --- |
| **Название узла** | **Нормированная центральность по степени** |
| Lady Gaga | 0.009 |
| Naomi Klein | 0.008 |
| Brand management | 0.007 |
| Katy Perry | 0.004 |
| Hewlett-Packard | 0.004 |
| Honda | 0.004 |
| Toyota | 0.004 |
| Sony | 0.004 |
| Promotion and marketing communications | 0.004 |
| Microsoft | 0.004 |

Через узел «Multinational companies headquartered in the United States» быстрее всего распространяется информация в полученном графе.

Таблица 2 – Лучшие вершины по нормированная центральность по близости

|  |  |
| --- | --- |
| **Название узла** | **Нормированная центральность по близости** |
| Multinational companies headquartered in the United States | 0.150 |
| Hewlett-Packard | 0.148 |
| Walmart | 0.146 |
| Companies listed on the New York Stock Exchange | 0.145 |
| Citation overkill | 0.144 |
| American brands | 0.143 |
| Kraft Foods | 0.143 |
| OpenCorporates groupings | 0.142 |
| Fake news website | 0.139 |
| Corporate spin-offs | 0.139 |

Узел «Fake news website» является самым важным передаточным звеном в полученном графе.

Таблица 3 – Лучшие вершины по нормированная центральность по посредничеству

|  |  |
| --- | --- |
| **Название узла** | **Нормированная центральность по посредничеству** |
| Fake news website | 0.168 |
| Citation overkill | 0.156 |
| Multinational companies headquartered in the United States | 0.148 |
| Starbucks | 0.139 |
| Howard Schultz | 0.133 |
| Coffee in Seattle | 0.131 |
| Brand management | 0.110 |
| Living people | 0.096 |
| Hewlett-Packard | 0.094 |
| Corporate spin-offs | 0.094 |

В результате визуализации был получен граф (Рисунок 4).

**Изображение выглядит как наружный объект, ночное небо

Автоматически созданное описание** Рисунок 4 – Граф по теме «brand»

## **Выделение сообществ**

Для возможности более детального рассмотрения графа и выявления дополнительных смысловых связей было принято решение разбить граф на сообщества. В результате работы Лувенского алгоритма было выявлено 113 сообществ (Рисунок 5). Цветовая карта была сгенерирована при помощи сайта [https://medialab.github.io/iwanthue/](https://medialab.github.io/iwanthue/.) (дата обращения 03.05.2022).

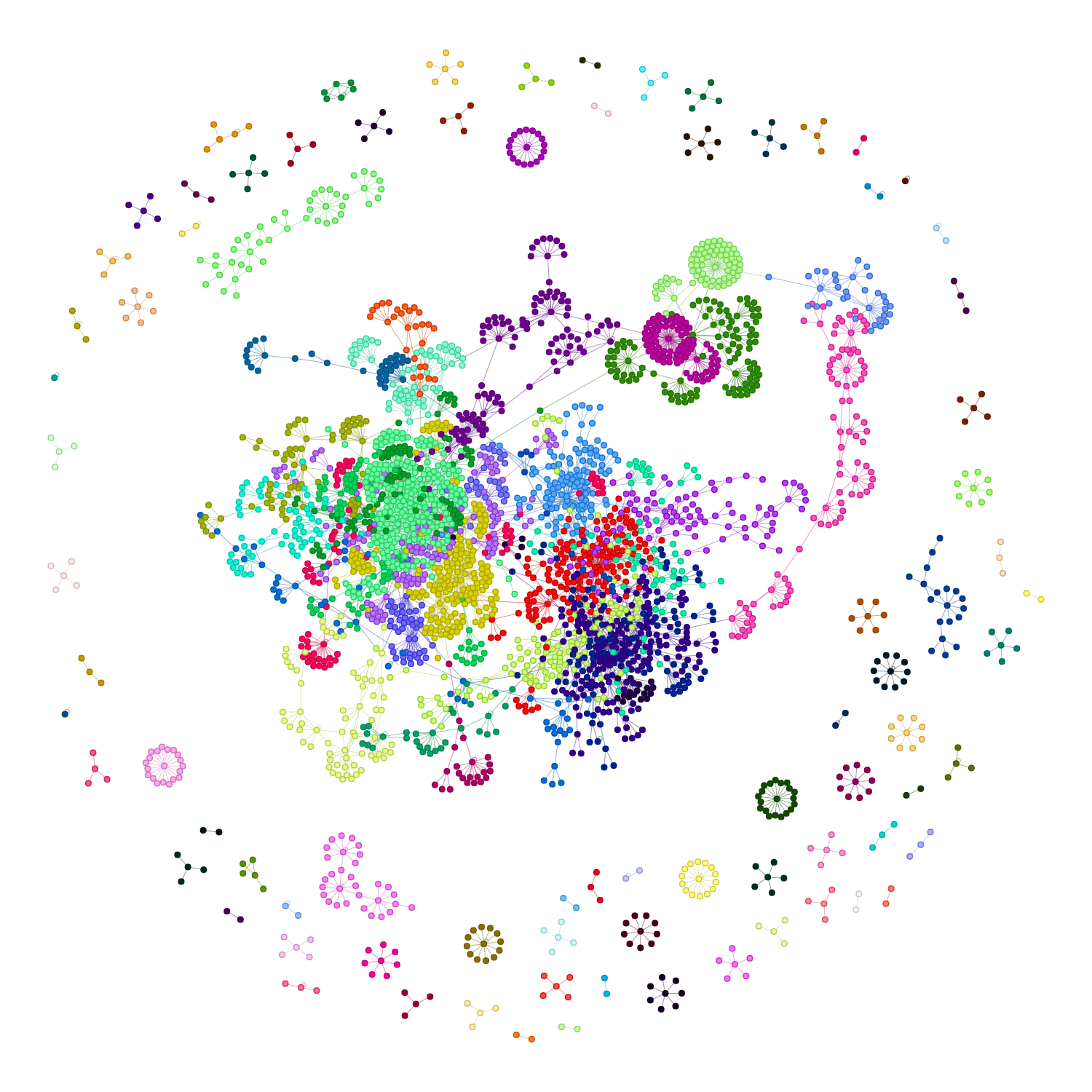
****

Рисунок 5 – Сообщества в полученном графе

Можем наблюдать, что в центре графа расположилось сообщество вокруг ключевого слова «brand» (Рисунок 6). При детальном рассмотрении подграфа можно увидеть, что вокруг вершины «brand» сосредоточились такие понятия данной предметной области, как «Brand Management», «Store Brands», «Types of branding», «Trademarks», «Music industry» и др. Действительно, бренды могут быть как в сфере производства товаров (машин, техники, продуктов питания), так и в сфере услуг (музыка, мода и другие сервисы).

Изображение выглядит как паутина

Автоматически созданное описание Рисунок 6 – Сообщество с вершиной «brand» в полученном графе

Также были рассмотрены сообщества, которые включают в себя некоторые лучшие вершины с точки зрения центральности по степени (Рисунок 7 и 8). Действительно, Lady Gaga и Toyota имеют большое количество связей. Также стоит отметить, что выявленные сообщества были выделены вполне логично и закономерно. Lady Gaga попала в одно сообщество с Katy Perry – они являются американскими поп-исполнительницами, выступающими в одно и то же время.

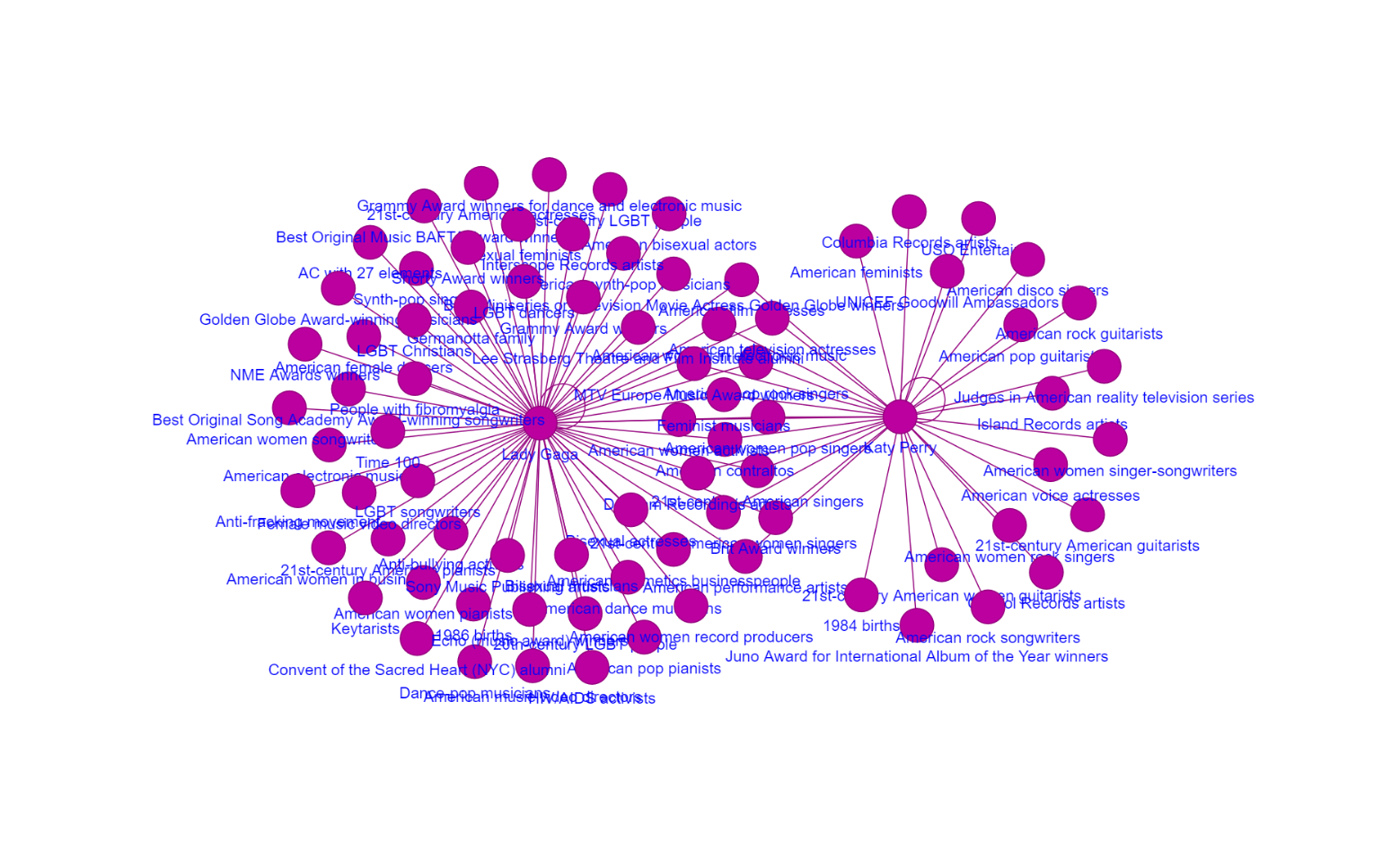
****

Рисунок 7 – Сообщество с вершиной «Lady Gaga» в полученном графе

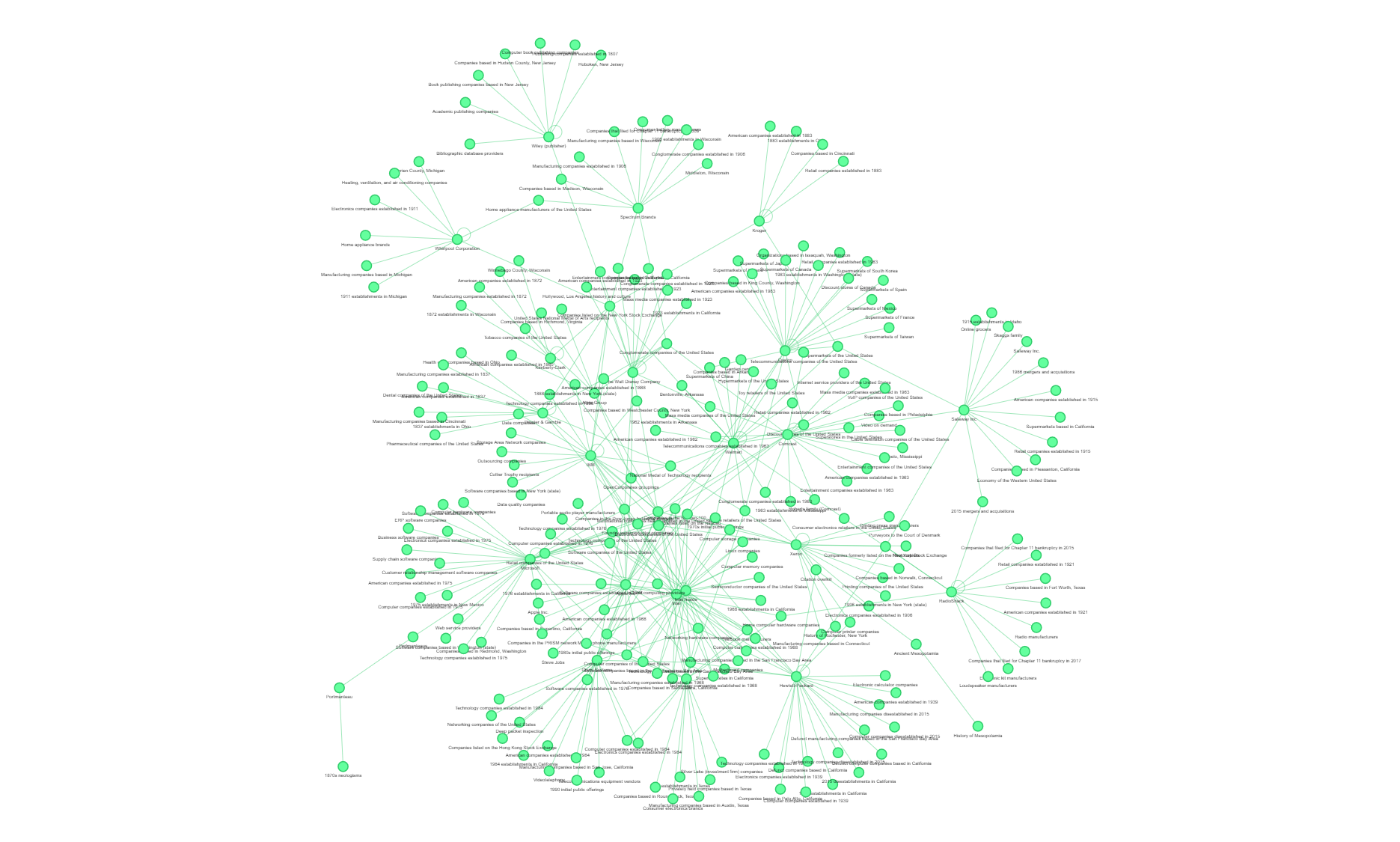
Toyota попала в одно сообщество с другими производителями автомобилей (Mercedes-Benz, Honda, Lexus, Porsche, Chrysler). Однако также в это сообщество попали японские корпорации (Seiko, Fujifilm, Sony), что логично, ведь Toyota является японским автоконцерном.

Изображение выглядит как наружный объект, звезда

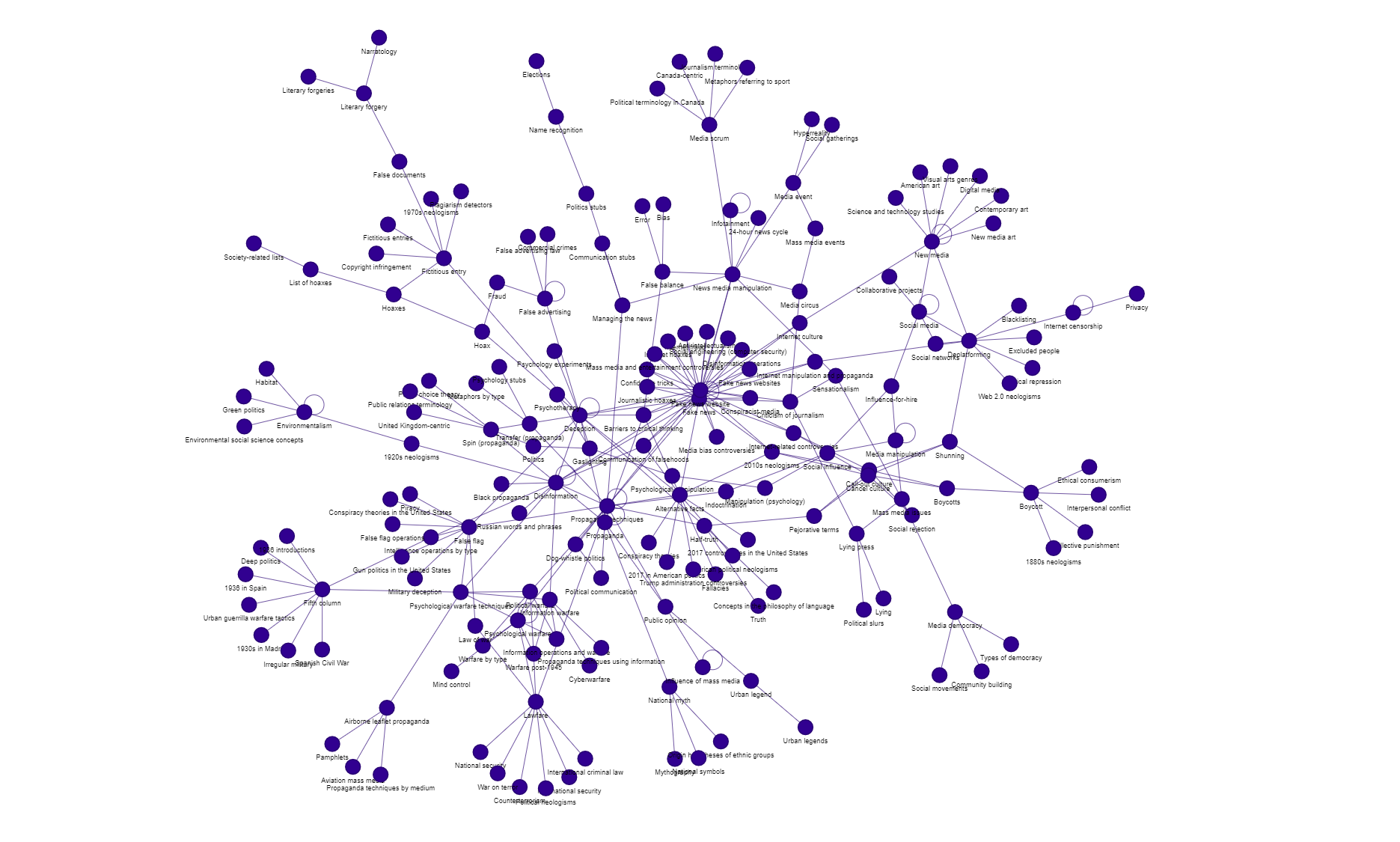
Автоматически созданное описание

Рисунок 8 – Сообщество с вершиной «Toyota» в полученном графе

Дополнительно были рассмотрены сообщества, которые включают в себя лучшие вершины с точки зрения центральности по близости и по посредничеству (Рисунок 9 и 10). «Multinational companies headquartered in the United States» и «Fake news website» действительно находятся посередине многих путей, что позволяет быстро распространять информацию от данных вершин и что свидетельствует о важности этих узлов как предаточных звеньев. Также стоит отметить, что в одно сообщество с «Multinational companies headquartered in the United States» попали Microsoft, IBM, Intel, Hewlett-Packard, Xerox, Apple Inc. Эти компании действительно имеют головные офисы в США. По данному подграфу можно узнать о компании Safeway – американской сети супермаркетов, которой нет в России. Она имеет связь с вершиной «2015 mergers and acquisitions», что означает, что данная компания был поглощена в 2015 году.

Рисунок 8 – Сообщество с вершиной «Multinational companies headquartered in the United States» в полученном графе

В подграфе с вершиной «Fake news website» содержится информация о «News media manipulation» и «Disinformation». Данные понятия действительно помогают понять суть «fake news».

**** Рисунок 9 – Сообщество с вершиной «Fake news website» в полученном графе

# **ЗАКЛЮЧЕНИЕ**

В данной курсовой работе с помощью средств Jupyter Notebook был построен граф знаний по ключевому слову «brand». Анализ графового набора показал, что выявленные связи в большинстве являются закономерными и логичными, однако есть и неожиданные взаимосвязи, что лишний раз доказывает эффективность графа знаний для изучения какой-либо предметной области.

Благодаря данной модели можно узнать особенности современных брендов и известных личностей, что поможет маркетологам своевременно проводить рекламные кампании и исследования современного рынка.

Изображение выглядит как силуэт

Автоматически созданное описание

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ**

1. Knowledge Graph// ibm.com [Электронный ресурс] URL: <https://www.ibm.com/cloud/learn/knowledge-graph#:~:text=A%20knowledge%20graph%2C%20also%20known,the%20term%20knowledge%20%E2%80%9Cgraph.%E2%80%9D> (дата обращения: 03.05.2022).
2. Revisiting the Network of Influential Rap Albums: Community Detection with the Louvain Algorithm// methodmatters.github.io [Электронный ресурс] URL: https://methodmatters.github.io/network-community-detection/ (дата обращения: 03.05.2022).
3. Knowledge Graphs With Machine Learning [Guide] // neptune.ai [Электронный ресурс] URL: <https://neptune.ai/blog/web-scraping-and-knowledge-graphs-machine-learning> (дата обращения: 01.05.2022).
4. База знаний // ru.wikipedia.org [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B7%D0%BD%D0%B0%D0%BD%D0%B8%D0%B9> (дата обращения: 20.04.2022).
5. Knowledge Graph // ru.wikipedia.org [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/Knowledge_Graph> (дата обращения: 20.04.2022).
6. Louvain method // en.wikipedia.org [Электронный ресурс] URL: <https://en.wikipedia.org/wiki/Louvain_method> (дата обращения: 02.05.2022).
7. Граф (математика) // ru.wikipedia.org [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)> (дата обращения: 21.04.2022).
8. Документация библиотеки NetworkX // networkx.org [Электронный ресурс] URL: https://networkx.org/documentation/stable/index.html (дата обращения: 02.05.2022).
9. Документация библиотеки Pyvis // pyvis.readthedocs.io [Электронный ресурс] URL: https:// pyvis.readthedocs.io/en/latest/documentation.html (дата обращения: 02.05.2022).
10. Документация библиотеки Pandas // pandas.pydata.org [Электронный ресурс] URL: https://pandas.pydata.org/docs/ (дата обращения: 02.05.2022).
11. Документация библиотеки Wikipediaapi // wikipedia.readthedocs.io [Электронный ресурс] URL: https://wikipedia.readthedocs.io/en/latest/code.html#api (дата обращения: 02.05.2022).
12. Документация библиотеки Tqgm // tqdm.github.io [Электронный ресурс] URL: https://tqdm.github.io/ (дата обращения: 02.05.2022).
13. Документация библиотеки Matplotlib // matplotlib.org [Электронный ресурс] URL: https://matplotlib.org/stable/index.html (дата обращения: 02.05.2022).
14. Документация библиотеки python-louvain // python-louvain.readthedocs.io [Электронный ресурс] URL: https://python-louvain.readthedocs.io/en/latest/ (дата обращения: 02.05.2022).

# **ПРИЛОЖЕНИЕ**

## **Характеристики компьютера и время выполнения программ**

Тип процессора: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz, 1190 МГц, ядер: 4, логических процессоров: 8

Память: 8,00 ГБ

Кэш-память: 6 MB Intel® Smart Cache

|  |  |
| --- | --- |
| Название программы | Время работы |
| Код 01. Извлечение информации из Википедии.ipynb | 726 s |
| Код 02. Обработка полученных данных из Википедии и построение графа.ipynb | 1.33 s |
| Код 03. Расчёт основных характеристик графа.ipynb | 42.2 s |
| Код 04. Расчет распределения и плотности распределения степеней вершин графа.ipynb | 2.93 s |
| Код 05. Выделение сообществ в графе и их отрисовка.ipynb | 9.26 s |

## 

## **Код программ**

**Код 01. Извлечение информации из Википедии.ipynb**

# подготовка библиотек

import pandas as pd

import wikipediaapi

from tqdm import tqdm

def scrape\_wikipedia(name\_topic, language = 'en'):

'''

функция, которая производит выгрузку названий и категорий статей Википедии по заданной теме (name\_topic)

также предусмотрена возможность выгружать данные не только с англоязчной Википедии

'''

def link\_to\_wikipedia(link):

'''

функция, которая выгружает необходимую информации о статье Википедии по ссылке (link), при условии ее существования

'''

try:

page = api\_wikipedia.page(link)

if page.exists() and page.categories:

return {'page': link, 'categories': list(page.categories.keys())}

except:

return None

# производим подключение к Wikipedia ARI, данные извлекаются в формате WIKI, а не HTML

api\_wikipedia = wikipediaapi.Wikipedia(language=language, extract\_format=wikipediaapi.ExtractFormat.WIKI)

# достаем страницы по топику, то есть по моей предметной области

name\_of\_page = api\_wikipedia.page(name\_topic)

# если не существует страниц по топику, то выводит на экран, что статей по топику не существует

if not name\_of\_page.exists():

print(f'Page {name\_of\_page} is not present')

return

# сохраняем ссылки на страницы по топику

links\_to\_page = list(name\_of\_page.links.keys())

# делает бегущую заполняющуюся полоску, чтоб отслеживать процесс

# в origin записываем набор данных о первой странице по топику, затем сюда записываются остальные результаты выгрузки статей по топику

# если они существуют

origin = [{'page': name\_topic, 'categories': list(name\_of\_page.categories.keys())}]

for link in tqdm(links\_to\_page):

if link\_to\_wikipedia(link)!=None:

origin.append(link\_to\_wikipedia(link))

# сохраняем собранные данные в таблицу Pandas

origin = pd.DataFrame(origin)

namespaces = ('Wikipedia', 'Special', 'Talk', 'LyricWiki', 'File', 'MediaWiki',

'Template', 'Help', 'User', 'Category talk', 'Portal talk','Category','Portal')

# отбираем только те топики, название которых не содержится в namespaces, тк это не нужные статьи

origin = origin[~(origin['page'].str.startswith(namespaces, na=True))]

# в категориях отрезаем первые символы до последнего двоеточия

# (Caregory:):

origin['categories'] = origin.categories.apply(lambda a: [b[len(b)-b[::-1].index(':'):] for b in a])

# выводим сколько страниц было просмотрено

print('Scraped pages', len(origin))

# в результате выводим таблицу с собранными данными

return origin

#запускаю функцию для выгрузки информации из Википедии по теме brand

data = scrape\_wikipedia('brand', language = 'en')

# распаковка списка в ячейках

data\_new = data.explode('categories')

# сохранение результатов выгрузки в файл формата csv

data\_new.to\_csv('data for work.csv')

**Код 02. Обработка полученных данных из Википедии и построение графа.ipynb (Рисунок 4)**

# подготовка библиотек

import pandas as pd

from pyvis.network import Network

import networkx as nx

# считывание данных из файла с выгруженными данными

data\_new = pd.read\_csv('data for work.csv').drop(['Unnamed: 0'],axis=1)

# удаление неинформативных строк и удаление ненужных знаков препинания

data\_new = data\_new[(data\_new.categories.str.find('Article')==-1)

&(data\_new.categories.str.find('article')==-1)

&(data\_new.categories.str.find('URL')==-1)

&(data\_new.categories.str.find('pages')==-1)

&(data\_new.categories.str.find('url')==-1)

&(data\_new.categories.str.find('title')==-1)

&(data\_new.categories.str.find('from')==-1)

&(data\_new.categories.str.find('sources')==-1)

&(data\_new.categories.str.find('Pages')==-1)

&(data\_new.categories.str.find('Wiki')==-1)

&(data\_new.categories.str.find('links')==-1)

&(data\_new.categories.str.find('errors')==-1)

&(data\_new.categories.str.find('AC with 0 elements')==-1)

&(data\_new.categories.str.find('All accuracy disputes')==-1)

&(data\_new.categories.str.find('missing periodical')==-1)

&(data\_new.categories.str.find('Julian–Gregorian uncertainty')==-1)

&(data\_new.categories.str.find('long volume value')==-1)

&(data\_new.categories.str.find('authors list')==-1)

&(data\_new.categories.str.find('others')==-1)

&(data\_new.categories.str.find('expanded')==-1)

&(data\_new.categories.str.find(' location')==-1)

&(data\_new.categories.str.find('generic name')==-1)

&(data\_new.categories.str.find('use')==-1)

&(data\_new.categories.str.find('Use')==-1)]

data\_new.page = data\_new.page.str.replace(',','')

data\_new.page = data\_new.page.str.replace('.','')

# удаление дубликатов

data\_new.drop\_duplicates()

# создаем граф из таблицы Pandas, где субъектами будут данные из столбца page, а объектами - из столбца categories

graph = nx.from\_pandas\_edgelist(data\_new, "page", "categories")

# сохранение графа в файл для длаьнейней работы

nx.write\_edgelist(graph, "data.edgelist", delimiter='\_', encoding = 'utf-8')

# визуализация графа с помощью библиотеки Pyvis

net = Network(height='750px', width='100%', font\_color='black',notebook=True)

net.from\_nx(graph, show\_edge\_weights=False)

net.toggle\_physics(True)

# сохранение и вывод полученного графа в виде файла html

net.show('mygraph.html')

**Код 03. Расчет основных характеристик графа.ipynb (Таблица 1, 2, 3)**

# считывание графа для анализа из файла edgelist

graph = nx.read\_edgelist("data.edgelist", delimiter='\_')

# количество вершин

N = len(graph.nodes())

# количество связей

L = len(graph.edges())

print(f'Количество вершин: {N}, Количество связей: {L}')

# плотность сети

pho = L/(N\*(N-1)/2)

print(f'Плостность сети: {pho:0.4f}')

# центральность по посредничеству

d = dict(nx.degree(graph) )

N = sum(d.values())

# сохранение в переменную лучших 10 значений по данной центральности

top = sorted(d, key=d.\_\_getitem\_\_)[-10:]

dd = {}

# нормировка и округление значений

for i in top[::-1]:

dd[i] = round(d[i]/N,3)

# создание таблицы с данными и сохранение в файл csv

df = pd.DataFrame.from\_dict(dd, orient='index')

df.to\_csv('top\_10\_degree\_centrality.csv',sep = ';')

# центральность по посредничеству

d = nx.betweenness\_centrality(graph)

# сохранение в переменную лучших 10 значений по данной центральности

top = sorted(d, key=d.\_\_getitem\_\_)[-10:]

dd = {}

# округление значений

for i in top[::-1]:

dd[i] = round(d[i],3)

# создание таблицы с данными и сохранение в файл csv

df = pd.DataFrame.from\_dict(dd, orient='index')

df.to\_csv('top\_10\_betweenness\_centrality.csv',sep = ';')

# центральность по близости

d = nx.closeness\_centrality(graph)

# сохранение в переменную лучших 10 значений по данной центральности

top = sorted(d, key=d.\_\_getitem\_\_)[-10:]

dd = {}

# округление значений

for i in top[::-1]:

dd[i] = round(d[i],3)

# создание таблицы с данными и сохранение в файл csv

df = pd.DataFrame.from\_dict(dd, orient='index')

df.to\_csv('top\_10\_closeness\_centrality.csv',sep = ';')

**Код 04. Расчет распределения и плотности распределения степеней вершин графа.ipynb (Рисунок 1, 2, 3)**

# подготовка библиотек

import numpy as np

import matplotlib.pyplot as plt

# создание словаря частот степеней вершин в графе

d = {}

for i in graph.degree():

d[i[1]] = d.get(i[1],0)+1

key = np.array(list(d.keys()))

val = np.array(list(d.values()))

# отрисовка полученного результата в линейных координатах

plt.scatter(key,val/sum(val))

plt.xlabel('Степень вершины')

plt.ylabel('Нормированное количетсво узлов')

plt.title('Плотность распределения вершин в линейных координатах')

plt.ylim(-0.1,1)

plt.savefig('Плотность распределения степеней вершин в линейных координатах.png')

plt.show()

# отрисовка полученного результата в log-log координатах

plt.scatter(key,val/sum(val))

plt.yscale('log')

plt.xscale('log')

plt.xlabel('Степень вершины')

plt.ylabel('Нормированное количетсво узлов')

plt.title('Плотность распределения вершин в log-log координатах')

plt.savefig('Плотность распределения вершин в log-log координатах')

plt.show()

key,val = [],[]

d = {}

# создание словаря частот степеней вершин в графе

for i in graph.degree():

d[i[1]] = d.get(i[1],0)+1

# создание упорядоченных рядов значений для отрисовки cdf

for i in sorted(d):

key.append(i)

val.append(d[i])

cs = np.cumsum(val)

# отрисовка cdf

plt.scatter(sorted(key), cs/max(cs))

plt.title("Cumulative Distribution plot")

plt.xlabel("Degree")

plt.savefig('Cumulative Distribution plot')

plt.show()

**Код 05. Выделение сообществ в графе и их отрисовка.ipynb (Рисунки 5 – 9)**

# установка библиотеки с Лувенским алгоритмом

!pip install python-louvain

#подготовка библиотек

from community import community\_louvain

from pyvis.network import Network

import networkx as nx

def draw\_graph3(networkx\_graph,notebook=True

,output\_filename='graph.html',show\_buttons=True,only\_physics\_buttons=False,

height=None,width=None,bgcolor=None,font\_color=None, pyvis\_options=None):

"""

Функция принимает на вход граф networkx и преобразует его в pyvis граф с множеством атрибутов

, затем отрисовывает полученный граф

notebook - показывать ли граф в Jupyter Notebook

output\_filename - как назвать файл с получившейся визуализацией

show\_buttons - показывать ли кнопки управления визуализацией

only\_physics\_buttons - показывать ли кнопки управления визуализацией отвечающие за движения графа

height - высота окна визуализации

width - ширина окна визуализации

bgcolor - цвет фона

font\_color - цвет подписей

pyvis\_options - доп опции pyvis

"""

# импортируем библиотеку

from pyvis import network as net

# создаем pyvis граф

network\_class\_parameters = {"notebook": notebook, "height": height, "width": width, "bgcolor": bgcolor, "font\_color": font\_color}

pyvis\_graph = net.Network(\*\*{parameter\_name: parameter\_value for parameter\_name, parameter\_value in network\_class\_parameters.items() if parameter\_value})

# добавляем в граф вершины с атрибутами

for node,node\_attrs in networkx\_graph.nodes(data=True):

pyvis\_graph.add\_node(node,\*\*node\_attrs,default\_node\_size=15, default\_edge\_weight=1)

# добавление связей в граф с атрибутами

for source,target,edge\_attrs in networkx\_graph.edges(data=True):

if not 'value' in edge\_attrs and not 'width' in edge\_attrs and 'weight' in edge\_attrs:

edge\_attrs['value']=edge\_attrs['weight']

pyvis\_graph.add\_edge(source,target,\*\*edge\_attrs)

# включение кнопок управления графом

if show\_buttons:

if only\_physics\_buttons:

pyvis\_graph.show\_buttons(filter\_=['physics'])

else:

pyvis\_graph.show\_buttons()

# добавление специальных опций графу

if pyvis\_options:

pyvis\_graph.set\_options( pyvis\_options)

# возвращает html файл, который также сохраняется и выводится на экран

return pyvis\_graph.show(output\_filename)

# применение Лувенского алгоритма к графу, получаем словарь вершин со значениями номеров сообщества

partition\_object = community\_louvain.best\_partition(graph)

# количество выделенных сообщества

values = [partition\_object.get(node) for node in graph.nodes()]

print(f'Количество выявленных сообществ: {len(set(values))}')

# список цветов для раскрашивания

color\_list = ["#51a8ff","#f80a0e","#01f3aa","#b63ff1","#95d700","#310090","#a3ff67","#ac00bc","#01d658","#fa6bff",

"#85ff7e","#4c008c","#cbff69","#6f68ff","#d9d200","#012294","#fdff62","#710093","#64ff9e","#f800a2","#00a028","#b871ff",

"#2b8e00","#bc009f","#b2ff91","#42006c","#fffc73","#004ec0","#a4b400","#7876ff","#5c9600","#ff7ef3","#019737","#ff50be",

"#01f8d5","#ef0020","#58f7ff","#fe005b","#7cffd1","#e1006e","#00a467","#b40066","#e8ff81","#200046","#ffe76a","#013e98",

"#b5a800","#016fdc","#e69300","#6c9bff","#ff7c18","#02b2f1","#ff5c21","#00d9cf","#ff4c47","#02a1a1","#a11a00","#66c8ff",

"#ac0022","#c6ffa0","#560053","#e7ff9e","#1f0035","#ffd55a","#002b63","#cc9700","#0289d0","#c57900","#0066a6","#ffbf62",

"#170027","#ffee9c","#001b30","#ffd574","#005088","#b15000","#8fc1ff","#792000","#c2fffc","#a0003a","#d1ffcd","#710049",

"#00743b","#ff5594","#114b00","#ffa8ee","#5e7100","#a9adff","#8d6e00","#c5cbff","#1b3b00","#ff90d0","#005831","#ff6d97",

"#00856f","#900053","#b5e3ff","#581400","#fbf8ff","#00181b","#ff7b6c","#002e16","#ffc0f4","#2a2c00","#ffd9f0","#261300",

"#ffece8","#4c0023","#ffe0b4","#00324f","#ffbc85","#002e28","#ff8496","#017695","#ffa686","#431d00","#685e00","#64001f",

"#7c5400","#703500"]

# создание палитры цветов для нашего разделения на сообщества

color\_list = color\_list[0:len(set(values))]

# создание массива Pandas для присвоения цвета к номеру сообщества

color\_dict = pd.Series(color\_list,

index=np.arange(0,len(set(values)))).to\_dict()

# заменяем номера сообществ на цвета в словаре

for key, value in partition\_object.items():

partition\_object[key] = color\_dict[value]

# добавляем к вершинам графа атрибут цвета

nx.set\_node\_attributes(graph, partition\_object, 'color')

# применянем функцию к нашему графу с выявленными сообществами

draw\_graph3(graph, height = '1000px', width = '1000px',

show\_buttons=False,

output\_filename='graph\_output\_communities.html', notebook=True)

from pyvis.network import Network

import networkx as nx

# отбираем сообщество, где находится вершина brand

selected\_nodes = [n for n,v in graph.nodes(data=True) if v['color'] == list(graph.nodes['brand'].values())[0]]

# сохраняем в переменную подграф с сообществом около вершины brand

sub\_graph = graph.subgraph(selected\_nodes)

# рисуем подграф

net = Network(height='750px', width='100%', font\_color='black',notebook=True)

net.from\_nx(sub\_graph,default\_node\_size=15, default\_edge\_weight=1, show\_edge\_weights=False)

net.toggle\_physics(True)

# выводим на экран и сохраняем рисунок

net.show('mysubgraph\_brand.html')

# отбираем сообщество, где находится вершина Toyota

selected\_nodes = [n for n,v in graph.nodes(data=True) if v['color'] == list(graph.nodes['Toyota'].values())[0]]

# сохраняем в переменную подграф с сообществом около вершины Toyota

sub\_graph = graph.subgraph(selected\_nodes)

# рисуем подграф

net = Network(height='750px', width='100%', font\_color='black',notebook=True)

net.from\_nx(sub\_graph,default\_node\_size=15, default\_edge\_weight=1, show\_edge\_weights=False)

net.repulsion(node\_distance=200, spring\_length=100)

net.toggle\_physics(True)

# выводим на экран и сохраняем рисунок

net.show('mysubgraph\_Toyota.html')

# отбираем сообщество, где находится вершина Lady Gaga

selected\_nodes = [n for n,v in graph.nodes(data=True) if v['color'] == list(graph.nodes['Lady Gaga'].values())[0]]

# сохраняем в переменную подграф с сообществом около вершины Lady Gaga

sub\_graph = graph.subgraph(selected\_nodes)

# рисуем подграф

net = Network(height='750px', width='100%', font\_color='blue',notebook=True)

net.from\_nx(sub\_graph,default\_node\_size=15, default\_edge\_weight=1, show\_edge\_weights=False)

net.toggle\_physics(True)

# выводим на экран и сохраняем рисунок

net.show('mysubgraph\_lady\_gaga.html')

# отбираем сообщество, где находится вершина Multinational companies headquartered in the United States

selected\_nodes = [n for n,v in graph.nodes(data=True) if v['color'] == list(graph.nodes['Multinational companies headquartered in the United States'].values())[0]]

# сохраняем в переменную подграф с сообществом около вершины Multinational companies headquartered in the United States

sub\_graph = graph.subgraph(selected\_nodes)

# рисуем подграф

net = Network(height='750px', width='100%', font\_color='black',notebook=True)

net.repulsion(node\_distance=250, spring\_length=150)

net.from\_nx(sub\_graph,default\_node\_size=15, default\_edge\_weight=1, show\_edge\_weights=False)

net.toggle\_physics(True)

# выводим на экран и сохраняем рисунок

net.show('mysubgraph\_usa\_brands.html')

# отбираем сообщество, где находится вершина Fake news website

selected\_nodes = [n for n,v in graph.nodes(data=True) if v['color'] == list(graph.nodes['Fake news website'].values())[0]]

# сохраняем в переменную подграф с сообществом около вершины Fake news website

sub\_graph = graph.subgraph(selected\_nodes)

# рисуем подграф

net = Network(height='750px', width='100%', font\_color='black',notebook=True)

net.from\_nx(sub\_graph,default\_node\_size=15, default\_edge\_weight=1, show\_edge\_weights=False)

net.repulsion(node\_distance=200, spring\_length=50)

net.toggle\_physics(True)

# выводим на экран и сохраняем рисунок

net.show('mysubgraph\_fake\_news.html')

## **Список файлов**

|  |  |
| --- | --- |
| Имя файла | Имя программы |
| Рисунок 1. Плотность распределения степеней вершин в линейных координатах | Код 04. Расчет распределения и плотности распределения степеней вершин графа.ipynb |
| Рисунок 2. Плотность распределения степеней вершин в log-log координатах | Код 04. Расчет распределения и плотности распределения степеней вершин графа.ipynb |
| Рисунок 3. Распределение степеней вершин | Код 04. Расчет распределения и плотности распределения степеней вершин графа.ipynb |
| Рисунок 4. Граф по теме «brand» | Код 02. Обработка полученных данных из Википедии и построение графа.ipynb |
| Рисунок 5. Сообщества в полученном графе | Код 05. Выделение сообществ в графе и их отрисовка.ipynb |
| Рисунок 6. Сообщество с вершиной «brand» в полученном графе | Код 05. Выделение сообществ в графе и их отрисовка.ipynb |
| Рисунок 7. Сообщество с вершиной «Lady Gaga» в полученном графе | Код 05. Выделение сообществ в графе и их отрисовка.ipynb |
| Рисунок 8. Сообщество с вершиной «Toyota» в полученном графе | Код 05. Выделение сообществ в графе и их отрисовка.ipynb |
| Рисунок 8. Сообщество с вершиной «Multinational companies headquartered in the United States» в полученном графе | Код 05. Выделение сообществ в графе и их отрисовка.ipynb |
| Рисунок 9. Сообщество с вершиной «Fake news website» в полученном графе | Код 05. Выделение сообществ в графе и их отрисовка.ipynb |
| Таблица 1. Лучшие вершины по нормированная центральность по степени | Код 03. Расчет основных характеристик графа.ipynb |
| Таблица 2. Лучшие вершины по нормированная центральность по близости | Код 03. Расчет основных характеристик графа.ipynb |
| Таблица 3. Лучшие вершины по нормированная центральность по посредничеству | Код 03. Расчет основных характеристик графа.ipynb |