Федеральное государственное бюджетное образовательное учреждение высшего образования

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

**Департамент анализа данных и машинного обучения**

Пояснительная записка к курсовой работе

по дисциплине «Технологии анализа данных и машинное обучение»

на тему:

«Машинное обучение в задачах автоматизации наукометрической деятельности»

Выполнил(а):

студент(ка) группы ПМ21-5 факультета информационных технологий и анализа больших данных

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Здоронок Е.М.

Научный руководитель:

ассистент Шаталова А.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2023 г

Содержание:

[Введение 3](#_Toc135436661)

[Выбор издательства 4](#_Toc135436662)

[Процесс парсинга издательства 5](#_Toc135436663)

[Очистка датасета 11](#_Toc135436664)

[Выбор модели 16](#_Toc135436665)

[Оценка модели 22](#_Toc135436666)

[Подбор параметров 24](#_Toc135436667)

[Интерпретация и визуализация лучшего результата 25](#_Toc135436668)

[Заключение 28](#_Toc135436669)

# **Введение**

В настоящее время наукометрика, т.е. научное исследование и анализ научной деятельности, получает все большее значение в связи с растущей конкуренцией в научной среде и необходимостью повышения эффективности научного процесса. Машинное обучение представляет собой мощный инструмент для решения задач наукометрии, таких как анализ научных публикаций, оценка влияния научных работ, прогнозирование научных достижений, кластеризация статей, поиск похожих статей и т. д.

Тема курсовой работы звучит как “Машинное обучение в задачах автоматизации наукометрической деятельности”. Эта тема довольно обширная, и к ней можно довольно много придумать задач, связанных с ml. Посоветовавшись с коллегами из Альфа-Банка, я выделил для себя задачу: кластеризовать статьи по темам, имея только текст и название статьи.

Цель работы: исследовать применение методов машинного обучения для кластеризации статей по темам в наукометрической деятельности.

Задачи работы:

* Собрать набор данных, содержащий тексты и названия статей, для проведения исследования.
* Предобработать данные, включая очистку текста, токенизацию и удаление стоп-слов.
* Исследовать различные методы представления текстовых данных.
* Разработать модель машинного обучения для кластеризации статей на основе выбранного метода представления данных.
* Обучить модель на подготовленных данных и провести ее оценку с использованием подходящих метрик.
* Проанализировать результаты кластеризации и интерпретировать полученные кластеры.
* Предложить возможные пути улучшения результатов исследования.
* Сформулировать выводы и подвести общие итоги выполненной работы.

# **Выбор издательства**

Перед тем как приступать к применению методов машинного обучения, необходимо получить наукометрические данные о статьях. Для этого нужно найти подходящее издательство, данные о статьях которого будут лежать в открытом доступе. Коллеги из Альфа-банка предоставили большой список издательств и предложили выбрать самостоятельно издательство.

После тщательного процесса поиска издательства мною было выбрано “ The Catholic University of America Press”. Сайт этого издательства хорошо структурирован, вся необходимая информация о статьях доступна.

На этой странице(https://www.cuapress.org/journals/) представлены все журналы, которые публикуются данным издательством. Проваливаясь глубже в каждый журнал, можно получить основную информацию о нем. Нажимая на “Preview on Project Muse”, мы получаем список всех выпусков данного журнала (пример https://muse.jhu.edu/journal/708), переходя на каждый из которых получаем список статей, включенных в конкретный выпуск (пример https://muse.jhu.edu/issue/42755). Там уже была представлена вся необходимая наукометрическая информация о каждой статье (пример https://muse.jhu.edu/pub/16/article/762408).

# **Процесс парсинга издательства**

Этот этап заключается в том, что нужно вытянуть наукометрические данные из каждой статьи каждого выпуска.

Первым делом, я изучил структуру сайта, чтобы понимать, где именно лежит нужная мне информация. Для этого я открыл код страницы и тщательно разобрал его, выделив все основные HTML теги и атрибуты.

После этого я принялся за написание кода, который должен спарсить целое издательство.

Для начала я написал функцию get\_proxies(). Она предназначена для получения бесплатных прокси-серверов с нескольких источников. Конкретно, она обращается к сайтам hidemy.name и free-proxy-list.net.

Первый источник (hidemy.name) возвращает таблицу с IP-адресами и портами. Функция сначала загружает страницу с помощью библиотеки requests и парсит ее с помощью BeautifulSoup. Затем функция проходится по каждому элементу таблицы и извлекает IP-адрес и порт, объединяя их в формат IP-адрес:порт. Эти прокси-серверы добавляются в список proxies\_hd.

Второй источник (free-proxy-list.net) также предоставляет таблицу с прокси-серверами. Функция снова загружает страницу с помощью requests и парсит ее с помощью BeautifulSoup. Затем функция проходится по каждому элементу таблицы и извлекает IP-адрес и порт, а также уровень анонимности прокси. Все прокси добавляются в список proxies\_levels.

Затем функция объединяет все полученные прокси-серверы в один список и удаляет дубликаты. Результат возвращается в виде списка уникальных прокси-адресов.

Таким образом, функция get\_proxies() позволяет получить большое количество бесплатных прокси-серверов из разных источников и использовать их для анонимного и безопасного доступа к веб-ресурсам.

Также я написал функцию get\_protected\_request(), которая отправляет защищенный запрос на сервер, используя случайный прокси-сервер из списка переданных прокси-серверов. Заголовки также передаются в запросе.



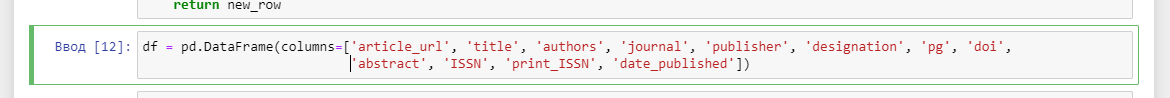
Если аргумент sleep имеет значение True, функция перед выполнением запроса задерживается на случайный промежуток времени от 0 до 2 секунд.

Функция возвращает объект BeautifulSoup, который содержит HTML-код ответа от сервера.

Затем я вручную собрал все ссылки журналов, у которых был открыт доступ к выпускам, получив файл ssylki.xlsx.

Далее, я написал функцию all\_info, которая собирает с сайта, нужную информацию, такую как: название статьи, автора статьи, журнал, страницы статьи, doi и т.д. На вход данная функция принимает ссылку на статью, на выходе получаем словарь, с нужными полями.

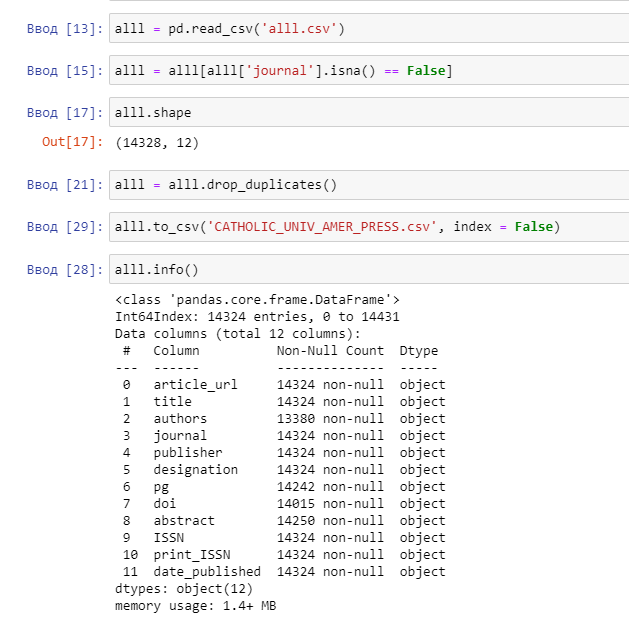


Далее я создаю pandas DataFrame, в котором и будут храниться все спаршенные статьи.

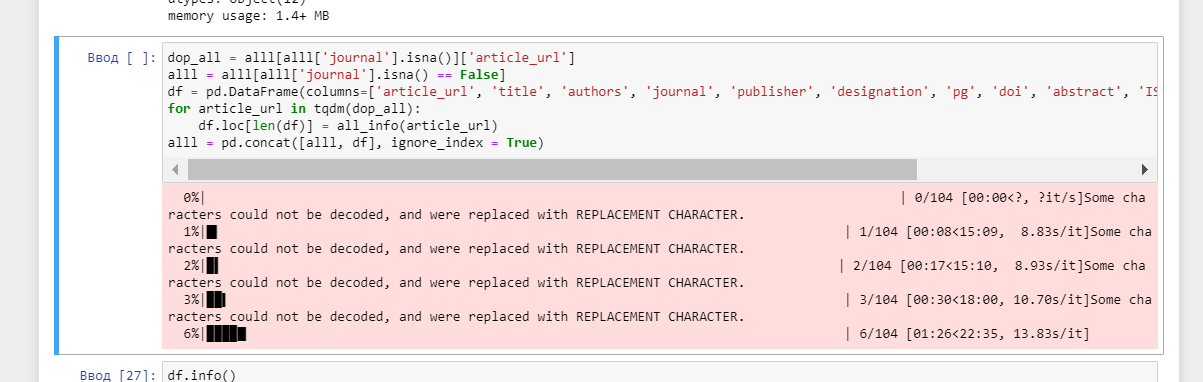
Затем, я написал код, который по ссылке на журнал будет доставать мне все его выпуски и статьи этих выпусков.

Я его запускал десятки раз, так как после определенной попытки спарсить журнал, сайт включал проверку на человека, прося ввести капчу. А затем и вовсе блокировал ip на определенное время.

Но тем не менее, у меня получилось собрать датасет из 14к статей, в котором изначально было много пропусков, опять же из-за безопасности сайта.



На скриншоте представлен финальный вид полученного датасета, в котором уже нет пропусков. Чтобы восполнить пропуски в нем, я заново парсил те ссылки, которые не получились в предыдущие разы с помощью данного кода.



В итоге получилось восполнить пропуски и получить датасет, в котором храниться вся доступная информация о статьях издательства “The Catholic University of America Press”.

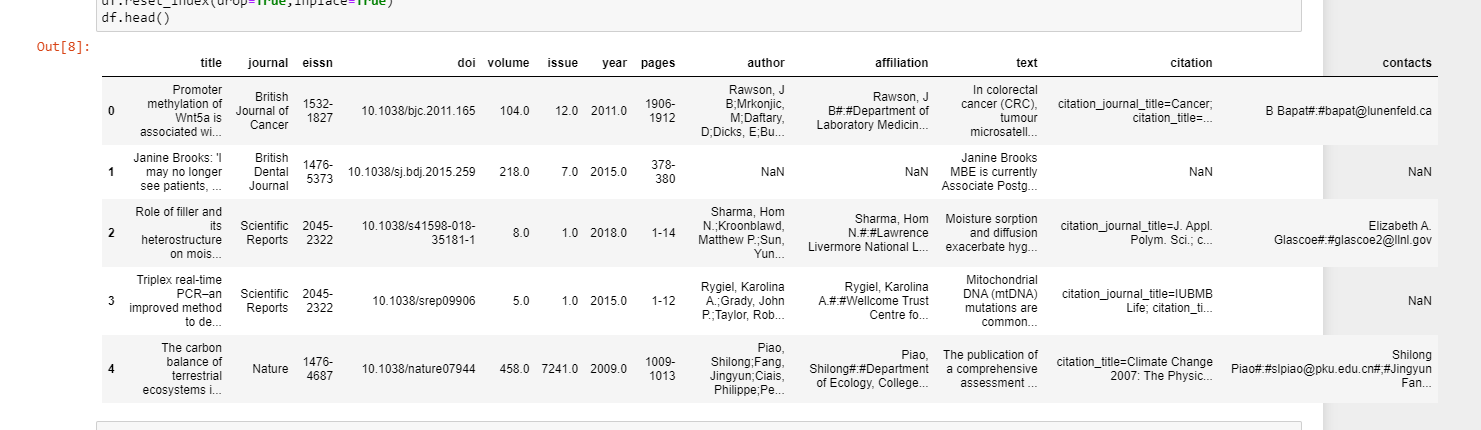
*Ссылка на код:* <https://colab.research.google.com/drive/1JA5C2kEf-iwu7LLm2zw_wGehfKMB7l7T?usp=sharing>

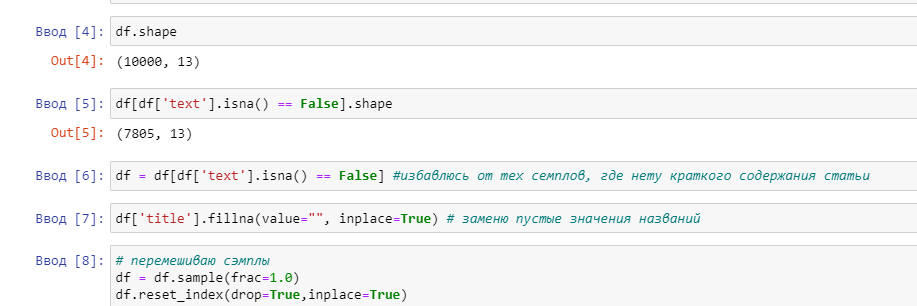
*Ссылка на датасет:* <https://drive.google.com/drive/folders/1O730GzGNy3ANLKxFTUAR-yK5J0iFeFxj>

# **Очистка датасета**

Сбор данных, содержащий тексты и названия статей, для проведения исследования.

Получив датасет издательства “ The Catholic University of America Press”, я хотел использовать его, однако обнаружил, что в нем у большей части статей вместо краткого содержания, дан большой отрывок самой статьи. Поэтому я использовал предложенный коллегами из Альфа-Банка датасет журнала “Nature”.

Я считал этот датасет, в нем оказалось 10000 статей с кратким содержанием и названием. Выглядит он следующим образом : . 

Так как мне интересно лишь 2 поля: text и title, то я рассматривал пропуски лишь в них. .

Я удалил все статьи, в которых не оказалось краткого содержания вовсе, и заменил пропуски в названиях на пустую строку. После следующих преобразования у меня осталось 7805 статей.

Также, перед тем как приступать к обработке данных, я перемешал их. На это есть 2 основные причины:

1. Устранение возможной зависимости порядка данных: Если данные в исходном датасете упорядочены по какому-либо признаку (например, по дате или алфавитному порядку), то это может привести к нежелательной корреляции между признаками и целевыми значениями.
2. Улучшение обобщающей способности модели: При обучении моделей машинного обучения важно, чтобы модель обучалась на разнообразных примерах и не становилась чувствительной к конкретному порядку данных.

Тем самым снизил потенциальный эффект нежелательной корреляции и достиг более обобщающей модели, способной обрабатывать новые данные более эффективно.

Далее, я принялся за предобработку текста. Этот процесс включает в себя три функции: initial\_clean, remove\_stop\_words и stem\_words, а также функцию apply\_all, которая объединяет эти три функции вместе для обработки текста.



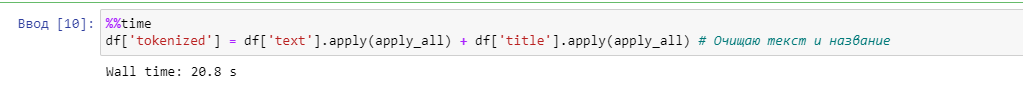
В начале я разработал функцию initial\_clean, которая очищала текст от ссылок, адресов почты и пунктуации. Кроме того, текст приводился к нижнему регистру. Это помогло убрать нежелательные элементы и обеспечить более структурированный вид текста.

Затем, я применил библиотеку NLTK для токенизации текста с использованием функции nltk.word\_tokenize. Токенизация разбивает текст на отдельные слова, что позволяет обрабатывать его на уровне отдельных элементов.

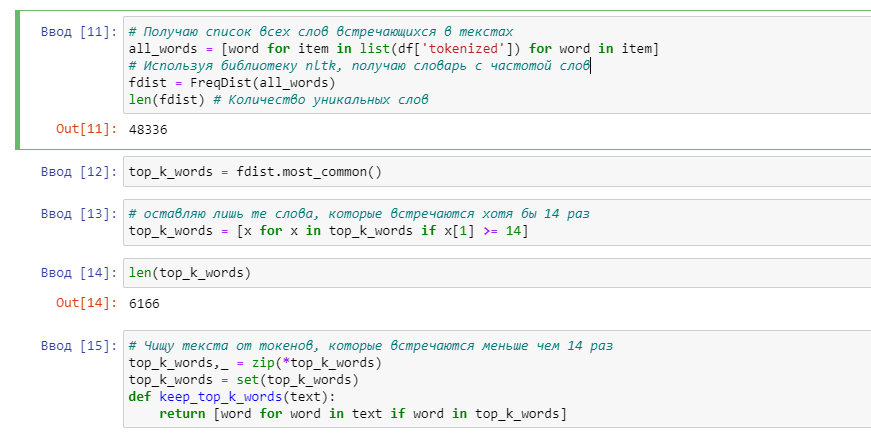
Далее, я использовал список стоп-слов stop\_words, предоставленный библиотекой NLTK, и создал функцию remove\_stop\_words. Эта функция удаляла стоп-слова из текста, такие как предлоги, союзы и местоимения, которые не несут смысловой нагрузки и могут искажать результаты анализа.

После этого я применил алгоритм стемминга, используя класс PorterStemmer из библиотеки NLTK. Стемминг сводит слова к их основной форме, что позволяет унифицировать разные формы одного и того же слова и сократить сложность анализа.

Наконец, я создал функцию apply\_all, которая объединила все вышеперечисленные шаги предобработки текста и позволила применить их ко всем статьям в моем датасете.



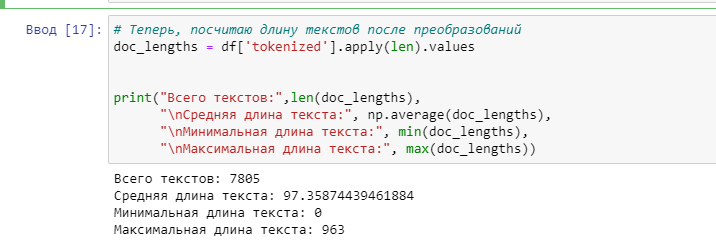
Этот процесс занял примерно 20 секунд, что является быстрым результатом.

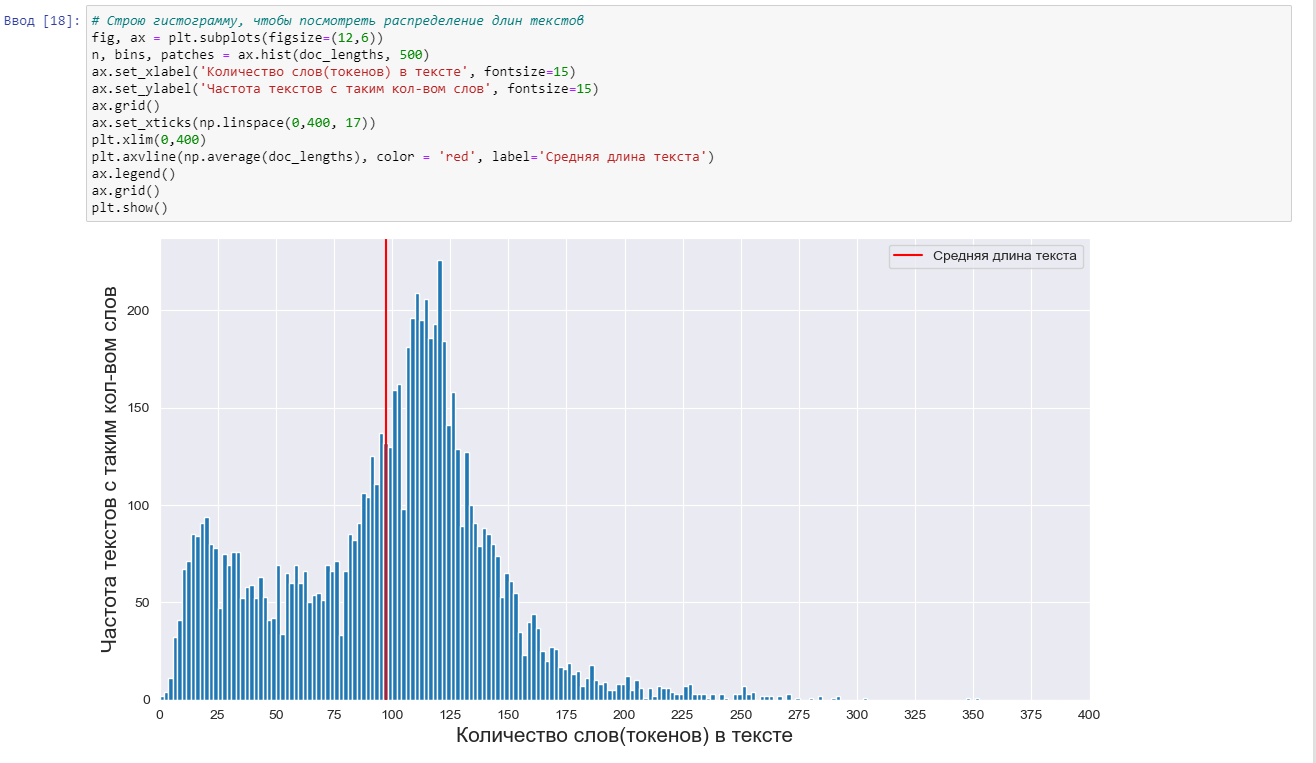
Разбив статьи на токены, я принялся за анализ частотности полученных токенов. 

В начале я создал список all\_words, включающий все токены, которые встречались в текстах.

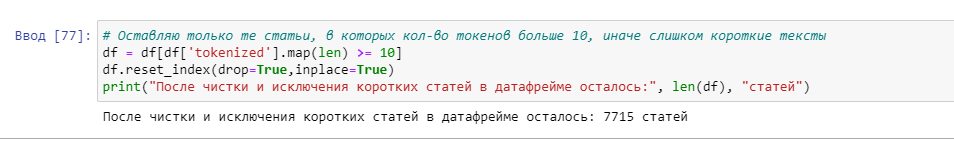
С использованием библиотеки NLTK, я создал объект fdist класса FreqDist, который предоставляет информацию о частотности слов в списке all\_words. Для каждого слова в all\_words было посчитано количество его вхождений. Это позволило мне определить, какие слова наиболее часто встречаются в текстах.

Количество уникальных токенов в статьях оказалось равно 48336.

Затем, я создал список top\_k\_words, который содержал наиболее часто встречающиеся слова и их частотность. Для этого, я использовал метод most\_common() объекта fdist, который возвращает отсортированный список кортежей (слово, частота) по убыванию частоты. Затем я отобрал только те слова, которые встречались хотя бы 14 раз, сократив размерность токенов примерно в 8 раз. Таким образом, список top\_k\_words содержал наиболее значимые слова. В нем осталось 6166 слов, что уже является довольно маленьким количеством.

На следующем этапе я подсчитал минимальную, среднюю и максимальную длину преобразованных текстов. И для более наглядного представления распределения длин текстов я построил гистограмму по длине текстов.

Решаю оставить только те текста, в которых токенов больше или равна 10, иначе просто получается совершенно неинформативный текст, который будет портить процесс обучения модели.



После финальной обработки текста в датафрейме осталось 7715 статей. Теперь можно переходить к выбору модели.

# **Выбор модели**

При выборе модели для кластеризации статей по темам я принял решение использовать модель LDA (Latent Dirichlet Allocation) по следующим причинам:

1. Вероятностная модель: LDA является вероятностной моделью, основанной на предположении, что каждый документ является смесью различных тем. Это предположение хорошо соответствует природе текстовых данных, где каждый документ может содержать несколько тем. Благодаря вероятностной природе модели LDA позволяет описать статистическую структуру коллекции документов и выявить скрытые темы.
2. Универсальность и применимость: LDA широко используется для анализа текстовых данных и кластеризации документов. Она позволяет выделить скрытые темы и определить вероятности принадлежности документов к каждой из тем. Благодаря этому LDA может быть применена к различным типам текстов и задачам кластеризации.
3. Интерпретируемость результатов: LDA предоставляет интерпретируемые результаты, так как каждая тема представляется распределением вероятности по словам. Это позволяет понять, какие слова связаны с каждой темой и дает возможность сопоставить полученные кластеры с конкретными темами или областями.
4. Распределение тем в документах: LDA также позволяет определить распределение тем в каждом документе. Это может быть полезно для анализа статей и определения их основной тематики. Кроме того, LDA может использоваться для поиска схожих документов, основываясь на их распределении по темам.

Учитывая эти преимущества, я решил использовать модель LDA для кластеризации статей по темам в моей курсовой работе. Она позволит мне выделить темы, определить вероятности принадлежности статей к каждой из тем, а также интерпретировать результаты, что поможет мне лучше понять структуру текстовых данных и их содержание.

Конечная цель модели LDA состоит в том, чтобы выделить скрытые темы в коллекции документов. Для этого модель проходит через четыре основных шага:

1. Инициализация:
   * В начале задаётся количество тем, которое мы хотим выделить из коллекции документов. Например, если у нас есть коллекция научных статей, мы можем выбрать число тем, соответствующее основным научным областям.
   * Также задаётся количество итераций, которое модель будет выполнять для обучения. Чем больше итераций, тем точнее может быть модель, но это требует больше времени для вычислений.
2. Предварительная обработка текста:
   * В этом шаге мы обрабатываем каждый документ в коллекции. Нам нужно удалить нерелевантные символы и пунктуацию, привести слова к их базовым формам (например, с помощью стемминга или лемматизации) и удалить общие слова, такие как "и", "в", "на" и т.д.
3. Инициализация распределений:
   * Здесь мы начинаем итерацию по всем словам в каждом документе. Каждому слову мы присваиваем случайную тему из всех возможных тем. Это начальное приближение для модели.
   * Также инициализируются два распределения: распределение тем в документах и распределение слов в темах. В начале они также задаются случайным образом.
4. Постепенное обновление:
   * Теперь модель начинает итерационный процесс обновления параметров для достижения лучшего представления данных.
   * В каждой итерации модель оценивает вероятность принадлежности каждого слова к каждой теме, исходя из текущих распределений тем в документах и слов в темах. Затем модель обновляет эти распределения на основе полученных оценок.
   * Шаги оценки вероятности (E-шаг) и обновления распределений (M-шаг) повторяются до тех пор, пока модель не достигнет сходимости или не выполнит заданное количество итераций.



Я выбрал количество тем и создал объект словаря dictionary, который будет преобразовывать слова в уникальные идентификаторы. Для этого я использую библиотеку corpora из gensim.

Затем, чтобы подготовить данные для модели LDA, я создаю список corpus. В этом списке каждый документ из df['tokenized'] преобразуется в вектор, состоящий из пар (слово, частота).

Далее, используя класс LdaModel из библиотеки gensim, я создаю объект модели LDA с заданными параметрами. В качестве аргументов передаются corpus (преобразованные векторы документов), num\_topics (количество тем), id2word (словарь), а также другие параметры, такие как random\_state, update\_every, chunksize, passes, alpha и per\_word\_topics. Эти параметры позволяют настроить модель LDA и определить ее поведение в процессе обучения.

Таким образом, после выполнения этого кода, я получил готовую модель LDA, которая обучена на моем корпусе документов. Эта модель будет использоваться для кластеризации статей по темам на основе их содержимого.

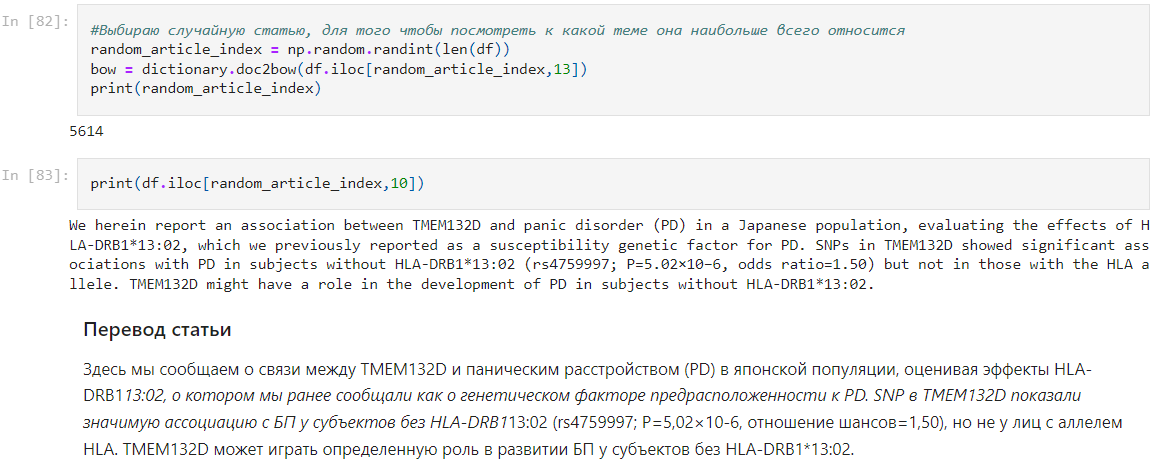
Далее я рассмотрел на какие темы модель разбила статьи.

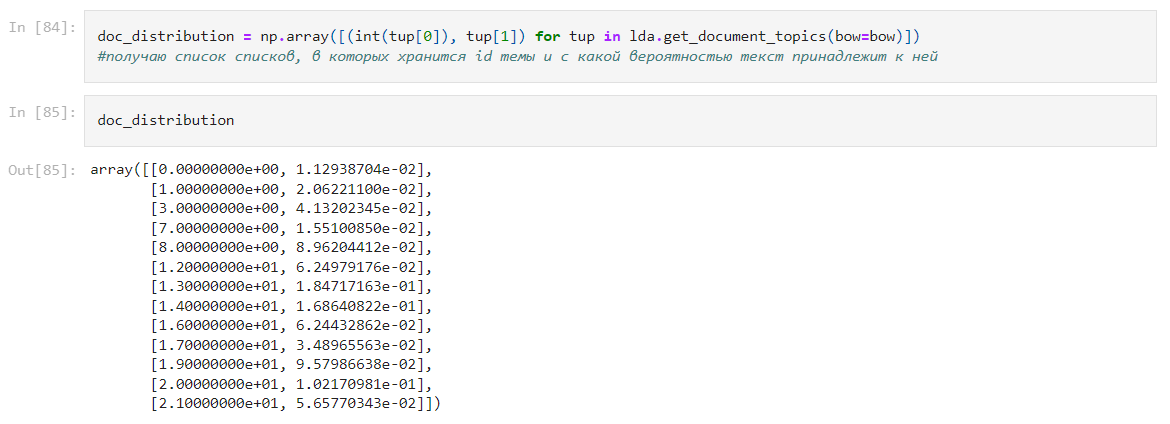
Изображение выглядит как текст, снимок экрана, Шрифт, число

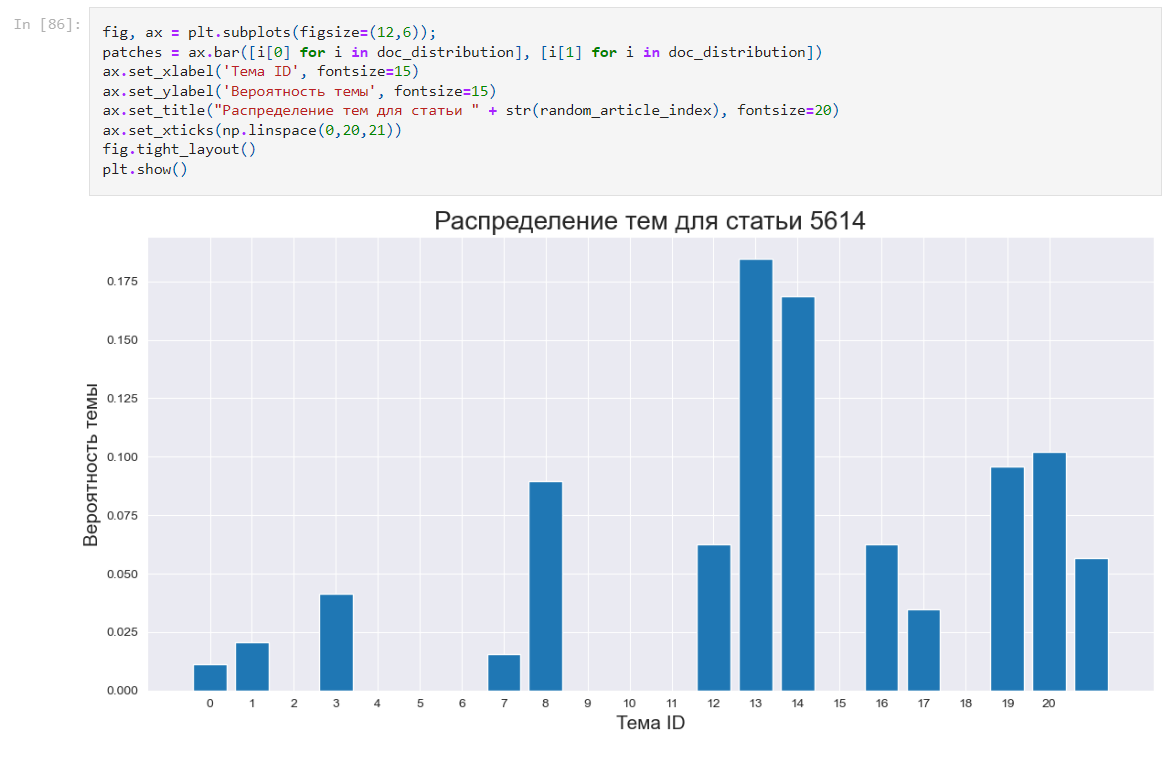
Автоматически созданное описание

Каждая тема охарактеризована списком токенов и их весом. Чем больше вес перед токеном, тем больше вклад он вносит в тему. Например, рассмотрим 18 тему. В данной теме основное внимание уделяется протеинам и различным процессам, связанным с ними.

Упоминаются понятия, такие как "protein" (протеин), "bind" (связываться), "membran" (мембрана), "nuclear" (ядерный), "degrad" (деградация), "recruit" (рекрутирование), "mitochondri" (митохондрии) и другие.

Разобравшись, как формируются темы кластеров, я решил посмотреть на случайную статью из датасета, чтобы узнать, к какой теме она относится. LDA модель способна предоставить для каждой статьи датасета список вероятностей отношения ее к какой-либо из тем.



Я получил список, состоящий из пар (id темы, вероятность), и решил визуализировать результат для наглядного анализа.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описаниеНа графике видно, что 13 и 14 тема самые вероятные для данной статьи. Рассмотрим их более детально.

Данная статья с вероятностью 0.185 относится к 13 теме - тема посвящена генам и генетическим процессам, с вероятностью 0.168 относится к 14 теме - исследования и анализ данных.

Читая перевод статьи, понимаю, что она действительно больше всего подходит к 13 теме, а значит модель, верно, классифицировала статью.

# **Оценка модели**

Я решил использовать метрику когерентности для оценки LDA модели, потому что она предоставляет количественную меру качества кластеризации текстов. Метрика когерентности помогает определить, насколько темы, выделенные моделью, являются интерпретируемыми и согласованными.

При выборе модели LDA с разными значениями параметров (например, различное количество тем), метрика когерентности позволяет сравнивать их результаты и выбрать самую оптимальную из них.

Метрика когерентности основана на анализе взаимосвязи слов внутри каждой темы. Она измеряет степень связи и согласованности между самыми значимыми словами, которые характеризуют одну и ту же тему. Если самые значимые слова внутри темы имеют сильную взаимосвязь и связаны с общей тематикой, то метрика когерентности будет высокой.

Использование метрики когерентности позволяет оценить качество кластеризации и понять, насколько модель LDA успешно выделяет интерпретируемые и согласованные темы.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Метрика когертности показала невысокий результат, а значит модель нуждается в доработке. Метрика должна быть приближена к единице. Перед тем как изменять параметры модели, я визуализировал результат кластеризации.

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Автоматически созданное описание

Кругами на левом графике представлены темы. Чем больше круг, тем больше токенов входит в него, соответсвенно тем больше распространена эта тема.

Хорошая тематическая модель будет иметь довольно большие непересекающиеся круги, разбросанные по всей диаграмме, а не сгруппированные в одной области. Модель со слишком большим количеством тем, как правило, имеет много пересекающихся кругов в одной области диаграммы.

Поскольку у меня вышло много маленьких пересекающихся кругов, то можно явно понять, что модель неидеально справилась с задачей и нуждается в доработке.

# **Подбор параметров**

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Мой подход к поиску оптимального количества тем состоит в том, чтобы построить множество моделей LDA с разными значениями количества тем и выбрать ту, которая даст наибольшее значение когерентности.

Изображение выглядит как текст, диаграмма, График, снимок экрана

Автоматически созданное описаниеВыбор ‘num\_topics’, обозначающий конец быстрого роста согласованности тем, обычно предлагает значимые и интерпретируемые темы. Выбор более высокого значения иногда может дать более детальные подтемы, что не всегда является хорошим результатом.

Как видно из графика, показатель согласованности быстро увеличивается, затем постепенно снижается. Наилучший показатель достигается при 6 темах, тогда метрика когертности равна 0.48.

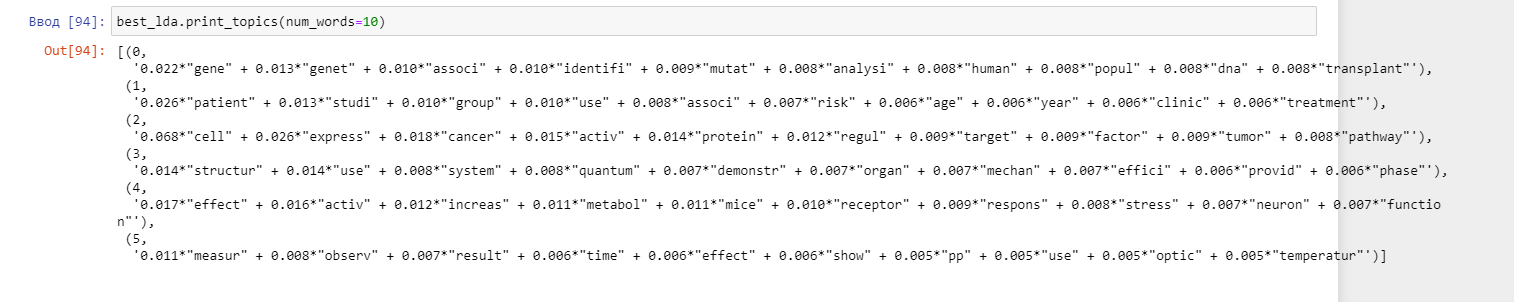
# **Интерпретация и визуализация лучшего результата**

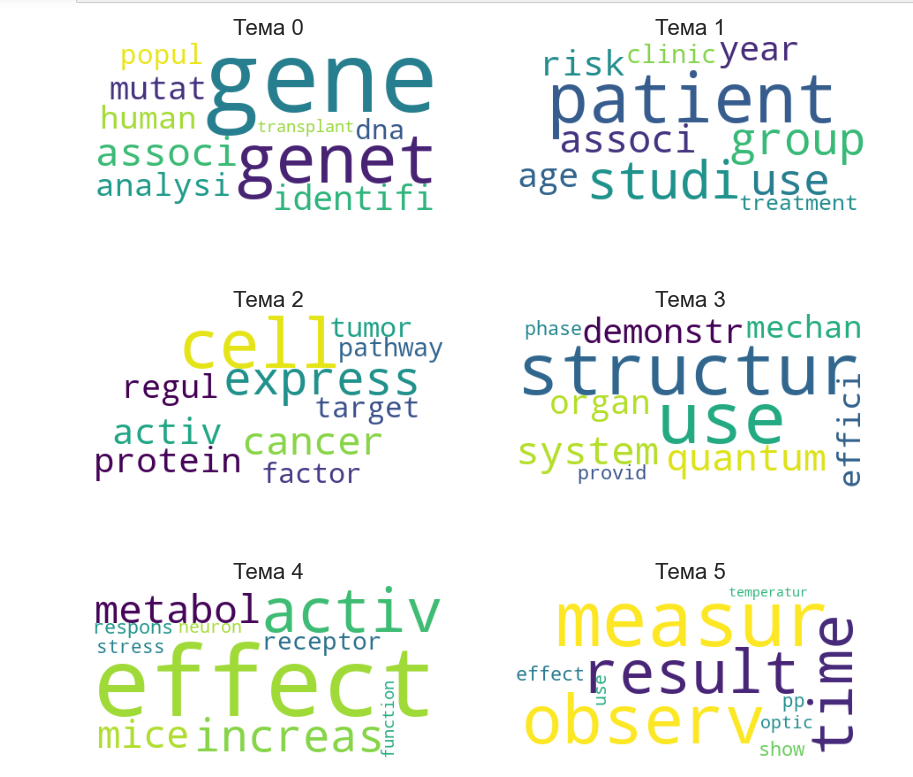
Изображение выглядит как текст, снимок экрана, программное обеспечение, диаграмма

Автоматически созданное описание

После оптимизации модели получилось, что на левом графике круги увеличились и перестали перекрывать друг друга. Что является явно лучше предыдущего результата. Наводя на каждую тему, можно посмотреть какие слова вносят наибольшее влияние в нее. Также наводя курсор на каждое слово, можно посмотреть его влияние на каждую из тем.

Рассмотрим более детально на какие темы разбила модель статьи.



Для более простого визуального восприятия я построил облако слов, в котором каждое слово пропорционально его весу.

Рассмотрев подробнее каждую из тем, я смог дать название и описание каждой.

Тема 0: Гены и генетика. Эта тема связана с идентификацией генов, мутациями, анализом генетических данных и ассоциациями с определенными популяциями. Вероятно, здесь идет речь о генетических исследованиях и генетической основе различных явлений.

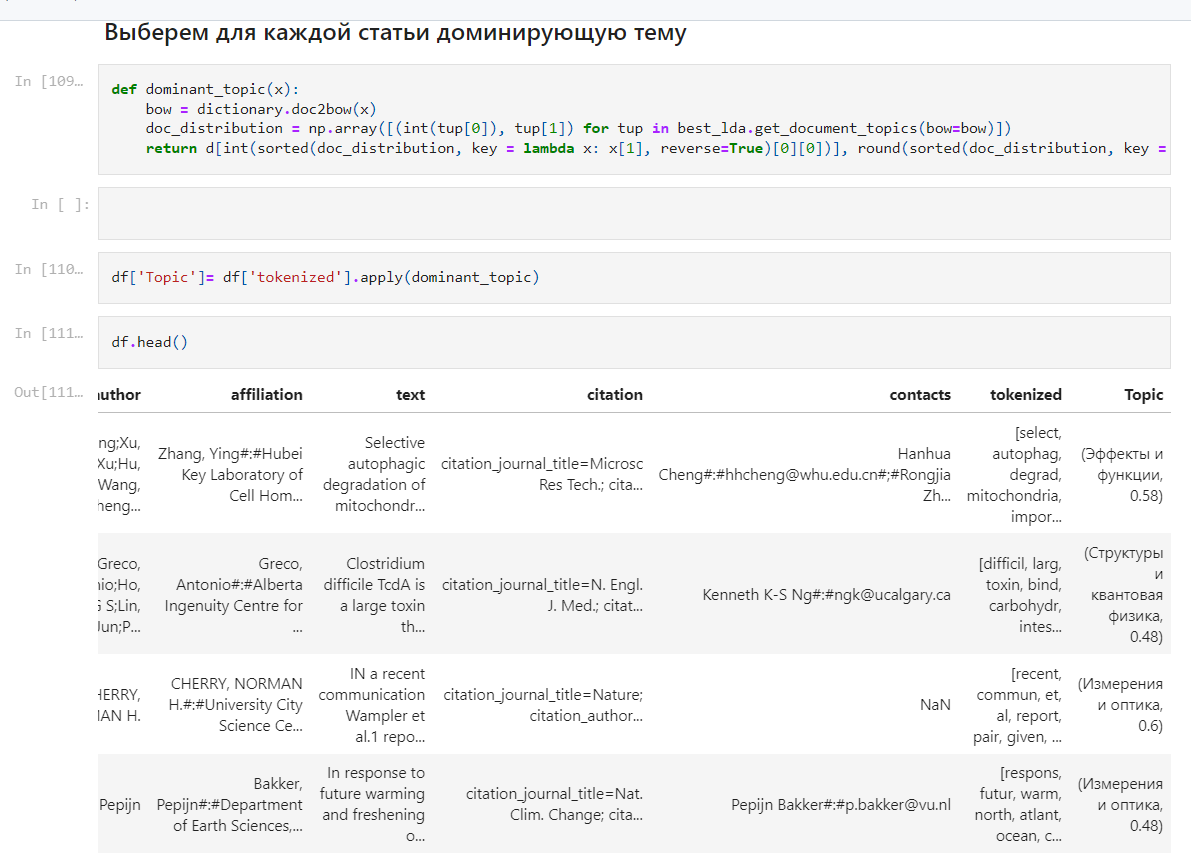
Тема 1: Пациенты и клинические исследования. Эта тема связана с исследованиями, проводимыми на пациентах, и анализом групп пациентов. Она включает использование различных методов исследования, связанных с клиническими данными, рисками и лечением различных заболеваний.

Тема 2: Клетки и рак. Эта тема связана с исследованиями клеток, особенно раковых клеток. Она включает экспрессию генов, активацию белков, регуляцию их функций, а также отслеживание путей, связанных с раком.

Тема 3: Структуры и квантовая физика. Эта тема связана с исследованием структур и систем на микроуровне, включая квантовую физику. Она может включать в себя исследования структуры материалов, механизмов и физических процессов, связанных с квантовыми явлениями.

Тема 4: Эффекты и функции. Эта тема связана с исследованиями различных эффектов и функций в организмах или системах. Включает изучение эффектов активации, метаболизма, рецепторов, ответов на стресс и функций нейронов.

Тема 5: Измерения и оптика. Эта тема связана с измерениями и оптикой. Включает изучение методов измерений, наблюдений, оптических свойств и их взаимосвязи с результатами исследований.

Теперь мы можем в явном виде понять, к какой теме относится каждая статья, выбрав доминирующую тему.

# **Заключение**

В ходе данной курсовой работы было проведено исследование по применению методов машинного обучения для кластеризации статей по темам в наукометрической деятельности. Цель работы заключалась в разработке модели, способной автоматически кластеризовать статьи на основе текстовых данных.

Для достижения поставленной цели был реализован ряд задач, включающих в себя сбор и предобработку данных. Была разработана модель машинного обучения, использующая выбранный метод представления данных, которая впоследствии была обучена на подготовленных наборах данных. Оценка модели была проведена с использованием соответствующих метрик, и результаты кластеризации были проанализированы и интерпретированы.

На основе полученных результатов можно сделать вывод, что предложенная модель позволяет эффективно кластеризовать статьи по темам в наукометрической деятельности, что позволяет автоматически выделять темы статьи и находить статьи с похожей тематикой.