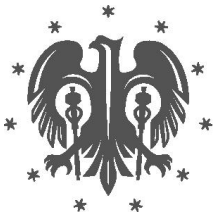

UMCS

UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE
Wydział Matematyki, Fizyki i Informatyki

Kierunek: WPISAĆ

Specjalność: WPISAĆ

Maksim Ryshko

nr albumu: NR 310030

Wykorzystanie metod klastrujących podczas analizy danych tekstowych

Using Clustering Methods in Text Data Analysis

Praca licencjacka

napisana w Zakładzie JAKIMŚ

pod kierunkiem dr. CZYIMŚ

Lublin rok WPISAĆ

Оглавление

Wstęp

Глава 1

Rozdzial 1

1.1. Klasterzyacja

1.1.1. Defenicja

Кластеризация представляет собой многомерную процедуру статистического анализа, направленную на сбор данных, содержащих информацию (признаки) о выборке объектов, с последующим упорядочиванием этих объектов в относительно однородные группы. Область применения кластеризации чрезвычайно широка: её используют в археологии, медицине, психологии, химии, биологии, государственном управлении, филологии, антропологии, маркетинге, социологии, геологии и других дисциплинах.

1.1.2. Цели кластеризации

Кластеризация чаще всего используется для следующих целей:

1. Понимание данных путем идентификации структуры кластеров. Разделение выборки на группы похожих объектов позволяет упростить дальнейшую обработку данных и принятие решений, применяя для каждого кластера свой метод анализа.

2. Компрессия данных. Если исходная выборка слишком велика, её можно сжать, сохраняя только по одному наиболее типичному представителю из каждого кластера. Заменяя все объекты в каждом кластере одним наиболее типичным представителем, можно значительно снизить объём данных, сохраняя при этом основные характеристики и структуру исходных данных.

3. Выявление новизны (выбросов или аномалий). Кластеризация позволяет выделить нетипичные объекты, которые нельзя отнести к какому-либо из кластеров. Эти объекты могут представлять интерес как потенциальные аномалии или выбросы, требующие дополнительного изучения, исследования или внимания.

1.1.3. Методы кластеризации

Общепринятой классификации методов кластеризации не существует, однако можно выделить ряд групп подходов:

1. Графовые алгоритмы кластеризации. Этот класс включает примитивные алгоритмы, основанные на построении графа сходства между объектами. В настоящее время они практически не применяются на практике.

2. Вероятностные алгоритмы кластеризации. Эти алгоритмы присваивают каждому объекту из обучающей выборки вероятность принадлежности к каждому из кластеров.

3. Иерархические алгоритмы кластеризации. Эти алгоритмы упорядочивают данные, создавая иерархию вложенных кластеров.

4. Алгоритмы, основанные на плотности данных:

    - K-средних. Итеративный алгоритм, основанный на минимизации суммарного квадратичного отклонения точек кластеров от их центров.
    - Распространение похожести. Алгоритм, который распространяет сообщения о сходстве между парами объектов для выбора типичных представителей каждого кластера.
    - Сдвиг среднего значения (mean shift). Метод, выбирающий центроиды кластеров в областях с наибольшей плотностью.
    - Спектральная кластеризация. Метод, использующий собственные значения матрицы расстояний для понижения размерности перед применением других методов кластеризации.
    - DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Алгоритм, группирующий точки в один кластер в областях с высокой плотностью и помечающий одиноко расположенные точки как шум.

1.1.4. Этапы кластеризации

Независимо от предмета исследования применение анализа кластеров включает следующие этапы:

1. Отбор выборки для кластеризации. Предполагается, что разумно классифицировать только количественные данные, поскольку числовые показатели могут быть количественно оценены.

2. Определение набора переменных, на основе которых объекты в выборке будут оцениваться, то есть пространства признаков. Признаки могут быть различными характеристиками объектов, которые имеют значение для исследования.

3. Вычисление значений той или иной меры сходства (или различия) между объектами. Цель - определить, насколько близки или различны объекты в выборке с точки зрения выбранных признаков.

4. Применение метода кластерного анализа для создания групп схожих объектов. Подруземевает использование одного из существуещего алгоритма кластеризации таким образом, чтобы объекты внутри одного кластера были максимально похожи друг на друга, а объекты из разных кластеров - максимально отличались.

5. Проверка достоверности результатов кластерного решения. Это может быть достигнуто с помощью различных методов, таких как визуализация кластеров, анализ стабильности кластеров и проверка качества разделения.

1.1.5. Применение

Применение метода кластерного анализа успешно используется во многих областях и дисциплинах. Также данный метод широко используется в информатике. Наиболее успешными примерами использования кластеризации в работе с текстовыми данными являются:

- кластеризация результатов поисковых запросов
- группировка пользователей со схожими интересами, которая помогает в персонализации контента, рекомендациях и таргетинге в рекламе
- группировка текстовых документов по их тематике, таких как новостные статьи, научные публикации или сообщения в социальных сетях

1.1.6. Оценка эффективности кластеризации

Проблема оценки качества в задаче кластеризации трудноразрешима, как минимум, по двум причинам. Во-первых, согласно теореме невозможности Клейнберга [4], не существует оптимального алгоритма кластеризации. Во-вторых, многие алгоритмы кластеризации не способны самостоятельно определить настоящее количество кластеров в данных; чаще всего количество кластеров задаётся на входе алгоритма и подбирается несколькими запусками. Тем не менее, принято выделять две группы методов оценки качества кластеризации:

1. Внешние методы сравнивают результат кластеризации с априори известным разделением на классы. Примером внешнего метода оценки можно рассмотреть индекс Рэнда. Этот метод оценивает, насколько много из тех пар элементов, которые находились в одном классе, и тех пар элементов, которые находились в разных классах, сохранили это состояние после применения алгоритма кластеризации.

2. Внутренние методы оценивают качество кластеризации, используя только информацию, содержащуюся в самих данных. Примером внутреннего метода является метод Cluster Cohesion. Идея данного метода заключается в том, что чем ближе друг к другу находятся объекты внутри кластеров, тем лучше разделение. Также можно отметить метод Cluster Separation, который оценивает качество кластеризации по тому, насколько далеко друг от друга находятся объекты разных кластеров.

1.2. Обработка естественного языка

1.2.1. Opis

Обработка естественного языка (Natural Language Processing, NLP) — это область исследований, которая объединяет методы машинного обучения и математической лингвистики с целью изучения и создания алгоритмов для анализа, понимания и генерации текста на естественных языках.

1.2.2. Предобработка текста

Предобработка текста является важным этапом NLP, которая переводит текст на естественном языке в формат удобный для распознования алгоритмами и дальнейшей работы. Предобработка состоит из различных этапов, которые могут отличаться в зависимости от задачи и реализации. Далее приведены одни из самых популярных основных подходов:

- Перевод всех букв в тексте в нижний или верхний регистры
- Удаление цифр (чисел) или замена на текстовый эквивалент (обычно используются регулярные выражения)
- Удаление пробельных символов (whitespaces)
- Токенизация (обычно реализуется на основе регулярных выражений)
- Удаление стоп слов
- Стемминг
- Лемматизация
- Векторизация

Далее подробнее будут описаны три последних метода.

1.2.3. Стемминг

Количество корректных словоформ, значения которых схожи, но написания отличаются суффиксами, приставками, окончаниями и прочим, очень велико, что усложняет создание словарей и дальнейшую обработку. Стемминг позволяет привести слово к его основной форме. Суть подхода в нахождении основы слова, для этого с конца и начала слова последовательно отрезаются его части. Правила отсекания для стеммера создаются заранее, и чаще всего представляют из себя регулярные выражения, что делает данный подход трудоемким, так как при подключении очередного языка нужны новые лингвистические исследования. Вторым недостатком подхода является возможная потеря информации при отрезании частей, например, мы можем потерять информацию о части речи.

1.2.4. Лемматизация

Данный подход является альтернативой стемминга. Основная идея в приведении слова к словарной форме — лемме. Пример лемматиации для польского языка:

- для существительных — именительный падеж, единственное число;
- для прилагательных — именительный падеж, единственное число, мужской род;
- для глаголов, причастий, деепричастий — глагол в инфинитиве несовершенного вида.

1.3. Векторизация

1.3.1. Opis

Большинство математических моделей работают в векторных пространствах больших размерностей, поэтому необходимо отобразить текст в векторном пространстве. Основной целью векторизации текста является создание представления текстовых данных, которое сохраняет семантическую и синтаксическую информацию о тексте и одновременно подходит для использования в математических моделях. Это позволяет эффективно анализировать и обрабатывать текстовые данные с помощью алгоритмов и алгоритмов машинного обучения.

1.3.2. Задачи векторизации

Основные задачи векторизации текста включают:

- Представление текста для анализа машинными алгоритмами.
- Поиск схожести между текстами или категоризация текстов по темам.
- Решение задач классификации, кластеризации или регрессии на основе текстовых данных.
- Создание моделей машинного обучения для автоматического обобщения текстовой информации

1.3.3. Методы векторизации

Первый шаг в векторизации текста - это определение пространства признаков, в котором текст будет представлен в виде векторов. Каждый признак может представлять собой слово, фразу, символ или иной элемент текста.
Существует несколько методов векторизации текста, включая:

- Мешок слов (Bag of Words, BoW): Каждый документ представляется в виде вектора, где каждый элемент соответствует отдельному слову, а значениями являются частоты встречаемости слов в документе.

- TF-IDF (Term Frequency-Inverse Document Frequency): Этот метод учитывает не только частоту встречаемости слов в документе (TF), но и инверсную частоту встречаемости слова во всех документах корпуса (IDF), что позволяет выделить ключевые слова.
- Word Embeddings: Это методы, основанные на обучении нейронных сетей, которые преобразуют слова в векторы непрерывного пространства с сохранением их семантических свойств.

1.3.4. Применение

Векторизация текста находит применение во многих областях, включая:

- Анализ тональности и сентиментов в социальных медиа и отзывах
- Классификация текстов на тематику или категории
- Рекомендательные системы, основанные на текстовых описаниях
- Автоматический перевод текста и реализация чат-ботов.

1.4. снижения разменрности

1.4.1. Дефениция

Снижение размерности - это процесс уменьшения количества признаков (измерений) в наборе данных, сохраняя при этом как можно больше информации. Часто используется в машином обучении где главная цель устранение в наборе признаков избыточных , неинформативных или слабо информативных которые могут понизить эффективность модели, а после такого преобразования она упрощается, и соответственно уменьшается размер набора данных в памяти и ускоряется работа алгоритмов на нем. Так же используется для упрощение анализа и визуализации данных, например путем редукции до низких размерностей таких как 2D или 3D.

1.4.2. Методы снижения размерности

Уменьшение размерности может быть осуществлено методами выбора признаков (англ. feature selection) или выделения признаков (англ. feature extraction).

Выбор признаков

Метод Выбор признаков оставляют некоторое подмножество исходного набора признаков, избавляясь от признаков избыточных и слабо информативных. Основные преимущества этого класса алгоритмов:

- Уменьшение вероятности переобучения
- Увеличение точности предсказания модели;
- Сокращение времени обучения;
- Увеличивается семантическое понимание модели.

Выделение признаков

Другим способом уменьшить размерность входных данных является выделение признаков. Эти методы каким-то образом составляют из уже исходных признаков новые, все также полностью описывающие пространство набора данных, но уменьшая его размерность и теряя в репрезентативности данных, т.к. становится непонятно, за что отвечают новые признаки. Все методы feature extraction можно разделить на линейные и нелинейные.

Методы выделения признаков

Линейные методы основываются на предположение, что зависимость между независимыми переменными и зависимой переменной линейна. Поэтому они с помощью линейых комбинацией, которые означают, что изменение в одной переменной приводит к пропорциональному изменению в другой. Одним из самых известных методов линейного выделения признаков является PCA . Основной идеей этого метода является поиск гиперплоскости, на которую при ортогональной проекции всех признаков максимизируется дисперсия. Нелинейные методы позволяют моделировать более сложные и нелинейные зависимости между переменными. Алгоритм t-sne стремится сохранить относительные расстояния между точками в исходном пространстве высокой размерности, а затем пытается воссоздать это распределение в низкоразмерном пространстве.

1.4.3. Применение

Метод снижение размерности часто и успешно используется в Машином обучение , Обработка изображений и компьютерное зрение, Биоинформатика и геномика , Финансовый анализ и торговля и многие другие области. Где снижение количества признаков позволяет оптимизировать обработку данных.

Глава 2

Rozdzial 2

2.1. Приложение

2.1.1. Opis

Данный проект направлен на создание инструмента для анализа текстовых данных, включающего предобработку текста, векторизацию и кластеризацию. Основная цель приложения — предоставить пользователю возможность гибкой настройки и использования различных методов обработки данных для решения конкретных задач.

2.1.2. Технологии

Для реализации проекта был выбран язык программирования Python. Он обладает рядом преимуществ:

- Богатый набор библиотек для реализации различных методов, таких как предобработка текста, векторизация и кластеризация.
- Высокая эффективность для анализа данных благодаря библиотекам, таким как NumPy, scikit-learn.
- Сообщество и документация, которые облегчают разработку и поддержку проектов.

2.1.3. GUI

Основная идея приложения заключается в создании удобного инструмента для анализа текстовых данных, где пользователь может самостоятельно выбирать и настраивать методы обработки под конкретные задачи с помощью графического интерфейса. Для разработки графического интерфейса были использованы библиотеки PyQt и Matplotlib:

- PyQt: Обеспечивает создание интуитивно понятных пользовательских интерфейсов.
- Matplotlib: Используется для визуализации данных, что позволяет пользователю легко интерпретировать результаты анализа.

Графический интерфейс позволяет пользователю загружать данные, настраивать параметры обработки текста, векторизации и кластеризации, а также визуализировать результаты.

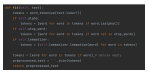

Рис. 2.1: Przyklad rys

2.1.4. OOP

Главным компонентом моего приложения является класс **Workspace**, который представляет собой контейнер, хранящий параметры для трех основных процессов:

- Обработка текста: Включает методы предобработки, такие как токенизация, удаление стоп-слов и лемматизация.
- Векторизация: Преобразует текстовые данные в числовые векторы с использованием методов, таких как TF-IDF или Word2Vec.
- Кластеризация: Реализует алгоритмы, такие как K-means или DBSCAN, для группировки текстовых данных.

Класс **Workspace** обеспечивает структурированное хранение и управление параметрами каждого этапа, что упрощает процесс настройки и выполнения анализа.

2.2. Текстовые данные

2.2.1. Opis

Для реализации данного проекта потребуются текстовые данные, которые в данном случае будут комментариями с YouTube. Мы будем использовать YouTube Data API v3 для получения информации о видео и связанных с ним комментариев. Данное API является бесплатным и простым в использовании. Для начала работы необходимо зарегистрироваться и создать проект в Google Cloud, а затем активировать YouTube Data API. После этого будет сгенерирован API-ключ, который мы будем использовать для аутентификации.

2.2.2. API запросы

В данном проекте будут использованы два HTTP-запроса:

- "GET https://www.googleapis.com/youtube/v3/commentThreads"
- "GET https://www.googleapis.com/youtube/v3/videos"

Первый запрос помогает получить комментарии к конкретному видео, а второй — получить информацию о видео. Подрбобно ознакомиться с полной документацией можно в [3]. Этапы получения комментариев:

1. Пользователь с помощью графического интерфейса (GUI) вводит ссылку на интересующее его видео.
2. Программа извлекает video_id из предоставленной ссылки.
3. Используя video_id, выполняется запрос GET https://www.googleapis.com/youtube/v3/videos для получения краткой информации о видео. Это позволяет пользователю убедиться, что выбранное видео соответствует его ожиданиям.
4. После подтверждения пользователем, что видео корректное, выполняется запрос GET https://www.googleapis.com/youtube/v3/commentThreads, чтобы получить комментарии к этому видео.
5. Полученные комментарии сохраняются в текстовый файл, где каждый комментарий записан в отдельной строке. Название файла соответствует уникальному идентификатору видео (video_id).

2.2.3. Обработка полученных данных

Результаты наших запросов будут предоставлены в формате JSON. Подробную информацию о структуре JSON-ответов можно найти в документации [3]. Для получения информации о видео будут использоваться только следующие поля:

- title: название видеоролика
- channelTitle: имя автора канала
- thumbnails: URL-адрес изображения

При получении комментариев необходимы только следующие поля:

- textDisplay: содержание комментария
- replies: контейнер комментариев-ответов

Поле textDisplay содержит текст комментария, а поле replies является контейнером для комментариев-ответов. Каждый ответ в поле replies.comments также имеет поле textDisplay, которое мы можем отдельно обработать. В конечном итоге мы получим список всех комментариев, где каждый элемент представляет из себя отдельный комментарий.

2.2.4. Хранение

После получения все комментарии будут сохраняться в отдельном каталоге comments, который будет создаваться в случае его отсутсвии, где каждый файл с коменатриями будет иметь название **video_id.txt**, для возможности индефекации. Для упрощения процесса хранения и доступа к данным, а также для обеспечения их совместимости с различными инструментами анализа, принято решение использовать обычный текстовый файл (.txt) для хранения комментариев. Каждый комментарий будет записан в отдельную строку файла, что упрощает структурирование данных и позволяет легко считывать их для анализа. Такой способ хранения данных был выбран по нескольким причинам:

- **Простота реализации**: Текстовые файлы легко создаются, читаются и редактируются с помощью множества программных средств и языков программирования.
- **Совместимость**: Практически все аналитические инструменты и библиотеки могут работать с текстовыми файлами, что облегчает интеграцию данных в различные этапы анализа.
- **Малое влияние на анализ**: Формат хранения не оказывает значительного влияния на сам процесс анализа данных. Основное внимание уделяется содержимому комментариев, а не способу их хранения.

2.3. Обработка текста

2.3.1. Opis

Следующим процессом является обработка текста. Важно понимать, как выглядят комментарии после их получения и сохранения с YouTube. Реализация обработки будет выполнена с помощью библиотеки NLTK (Natural Language Toolkit) и встроенных функций Python. Ознакомиться с документацией NLTK можно здесь [**?**]. Обработка текста разделена на несколько этапов:

- Обязательная обработка выполняющеся независимо от выбораных параметров пользователем.
- Примитивная обрботка удаление символ и цифр
- Удаление stop-words

- Лематизация
- Анализ

Все методы кроме обязательной обработки и анализа, являются необязательным, то есть будут выполняться только в случае если пользователь укажет что хочет использовать данные методы.

2.3.2. Обязательная обработка

Обязательными этапами обработки текста являются приведение слов к нижнему регистру и процесс токенизации, который устраняет необходимость удаления пробелов. С помощью функции **nltk.word tokenize** каждое слово будет разделено на токены. Эти этапы обязательны, так как они упрощают дальнейшую работу с текстом и гарантируют правильную работу многих последующих функций, которые могут быть чувствительны к регистру.

2.3.3. Примитивная обработка

Примитивная обработка подразумевает удаление неалфавитных символов или слов. Этот процесс не является обязательным, однако его использование помогает избавиться от различных языков разметки, которые YouTube использует для визуализации смайликов или инструментов изменения текста. Пример таких комментариев:

```
"Great video! (here should be emoji"
"Check this out: <a href='https://example.com'>link</a>"
```

В данном проекте это реализовано с помощью встроенной функции Python **isalpha()**, которая проверяет символы или слова на наличие неалфавитных символов.

```
1        def remove_non_alpha(tokens):
2        return [word for word in tokens if word.isalpha()]
```

2.3.4. Удаление stop-words

Этот этап реализует удаление слов, которые не несут значимой информации. Типичными примерами таких слов являются:

```
"the", "is", "in", "and"
```

Удаление стоп-слов уменьшает размер текста и делает информативные слова более заметными. Для этого используется словарь стоп-слов из библиотеки NLTK, который в случае необходимости можно дополнить своими словами.

```
1    from nltk.corpus import stopwords
2
3    def remove_stop_words(tokens):
4      stop_words = set(stopwords.words('english'))
5      return [word for word in tokens if word not in stop_words]
```

2.3.5. Лематизация

Последним необязательным процессом является лемматизация, которая приводит слова к их базовой форме. Это полезно в случаях, когда нам нужно сгруппировать разные формы одного слова (например, "running "ran "runs"к "run").

```
1    from nltk.stem import WordNetLemmatizer
2
3    def lemmatize_tokens(tokens):
4      lemmatizer = WordNetLemmatizer()
5    return [lemmatizer.lemmatize(word) for word in tokens]
```

2.3.6. Анализ

В конце, независимо от выбранных пользователем этапов, все токены проверяются на непустые значения, чтобы избежать полностью пустых слов и строк. Далее все токены объединяются в одну строку, разделённую пробелами. На этом этапе результат обработки возвращается и анализируется функциями частотного анализа (**freq_analysis**) и анализа длины (**length_analysis**). Эти функции будут использоваться для построения графиков, которые помогут пользователю визуально оценить частоту встречаемых слов и длины комментариев для последующего принятия решений.

freq analysis

Эта функция использует обработанные строки, создавая единый массив со всеми словами из всех комментариев, и применяет класс **collections.Counter** в Python. **Counter** используется для подсчёта хэшируемых объектов, таких как строки или числа, и предоставляет удобные методы для работы с частотными данными. Таким образом, функция генерирует два массива: массив слов и соответствующий ему массив количества каждого слова в тексте. Эти массивы будут использоваться для построения столбчатой диаграммы (bar chart), показывающей частоту встречаемости слов.

Пример реализации алгоритма в проекте:

```
        from collections import Counter
        import matplotlib.pyplot as plt

        def freq_analysis(tokens):
          data = ' '.join(self.nlp_data)
          words = nltk.word_tokenize(data)
          # words = [word.lower() for word in words]
          # words = [word for word in words if word.isalnum()]
          word_freq = Counter(words)
          words_counts_sorted = word_freq.most_common()
          self.words_sorted = [word_count[0] for word_count in words_counts_sorted]
          self.counts_sorted = [word_count[1] for word_count in words_counts_sorted]
```

length analysis

Эта функция также использует обработанные строки комментариев, определяя количество слов для каждого комментария. В конечном итоге генерируется массив количеств слов, который будет использован для построения гистограммы (histogram), показывающей распределение длины комментариев.

Пример реализации алгоритма в проекте:

```
        from collections import Counter
        import matplotlib.pyplot as plt

        def freq_analysis(tokens):
          word_counts = [len(nltk.word_tokenize(el)) for el in self.nlp_data]
          self.max_size = max(word_counts) + 1
          self.lengths = [0] * (self.max_size + 1)
          for count in word_counts:
          self.lengths[count] += 1
          endTime = time.time()
```

2.3.7. Итог

Все этапы обработки текста, от обязательных до необязательных, играют важную роль в подготовке данных для дальнейшего анализа. Их правильное выполнение гарантирует структурированный и удобный вид для обработки. В визуализация графиков анализруещих содержания комментариев позволит пользователям лучше понимать структуру и содержание комментариев, а также принимать обоснованные решения на основе представленных данных.

2.4. Wektoryzacja

2.4.1. Opis

На данном этапе текст готов к следующему этапу обработки. Для последующего анализа необходимо преобразовать комментарии в векторную форму (векторизировать). Будт использовано два метода: TF-IDF и Word2Vec. Эти методы позволят представить текстовые данные в виде числовых векторов, что облегчит дальнейший анализ и обработку. Векторизация будет реализована с использованием библиотек **scikit-learn** для метода TF-IDF и **gensim** для метода Word2Vec.

2.4.2. TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) — это статистическая мера, используемая для оценки важности термина в документе относительно коллекции документов (корпуса). Она широко применяется в задачах обработки естественного языка и информационного поиска. В библиотеке Scikit-learn существует удобный инструмент для расчета TF-IDF: класс TfidfVectorizer.

Параметры

- max_df: Число или доля. Термины, которые встречаются в большем числе документов, чем указано, игнорируются. Полезно для удаления общеупотребительных слов.
- min_df: Число или доля. Термины, которые встречаются в меньшем числе документов, чем указано, игнорируются. Полезно для удаления редких слов.
- ngram range: Тупл (min_n, max_n). Определяет диапазон для извлечения n-грамм.
- stop_words: Список слов или строка, определяющая слова, которые следует игнорировать.

Пояснение алгоритма

- Term Frequency (TF)
- Inverse Document Frequency (IDF)
- TF-IDF

Пример использования

```
from sklearn.feature_extraction.text import TfidfVectorizer

def vectorize_tfidf(corpus):
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(corpus)
return tfidf_matrix
```

Пример преобразования текстовых данных:

- До: ["This is a great video "I love this content"]
- После: [[0.5, 0.5, 0, 0.5, 0, 0.5], [0, 0, 0.7, 0, 0.7, 0]]

Для работы с TF-IDF необходимо иметь корпус текстов, который можно представить в виде списка строк, где каждая строка — это отдельный документ.

2.4.3. Word2Vec

Word2Vec — это алгоритм, разработанный для обучения векторных представлений слов (эмбеддингов), которые захватывают семантические отношения между словами. В библиотеке Gensim предоставляется удобный интерфейс для работы с Word2Vec.

Важные параметры

- sentences: Корпус текстов, представленный в виде списка списков слов.
- vector_size: Размерность векторного представления слов.
- window: Максимальное расстояние между текущим и предсказанным словом в предложении.
- min_count: Минимальная частота слова в корпусе для его учета.
- workers: Количество потоков для параллельного обучения.

Более глабальные настройки

- SG (Skip-Gram) и CBOW (Continuous Bag of Words): В Word2Vec есть два основных архитектурных подхода — Skip-Gram и CBOW. Параметр sg в Gensim позволяет выбрать между ними:
  - sg=0 использует CBOW (по умолчанию).
  - sg=1 использует Skip-Gram.
- Negative Sampling: В Word2Vec используется негативное сэмплирование для повышения эффективности обучения. Параметр negative задает количество "негативных"примеров для каждого положительного примера.
- Learning Rate: Параметр alpha задает начальную скорость обучения, а min_alpha — минимальную скорость обучения, которая уменьшается до этого значения по мере обучения.

Описание алгоритма

1. Инициализация весов: Веса нейронной сети инициализируются случайными значениями. Это веса между входным слоем и скрытым слоем, а также между скрытым слоем и выходным слоем.
2. Прямой проход (Forward Pass): Для каждого слова в предложении создается входной вектор. В случае Skip-Gram этот входной вектор используется для предсказания контекстных слов. В случае CBOW контекстные слова используются для предсказания текущего слова.
   - Skip-Gram:
     - Входное слово кодируется в виде одноразового вектора.
     - Одноразовый вектор умножается на веса между входным и скрытым слоями для получения скрытого представления.
     - Скрытое представление умножается на веса между скрытым и выходным слоями для предсказания контекстных слов.
   - CBOW:
     - Контекстные слова кодируются в виде одноразовых векторов.
     - Одноразовые векторы суммируются и умножаются на веса между входным и скрытым слоями для получения скрытого представления.
     - Скрытое представление умножается на веса между скрытым и выходным слоями для предсказания текущего слова.

3. Обратное распространение (Backpropagation): Ошибка предсказания рассчитывается путем сравнения предсказанного слова с фактическим. Затем эта ошибка распространяется обратно через сеть для обновления весов. Используется метод стохастического градиентного спуска (SGD) и, в некоторых случаях, негативное сэмплирование для ускорения обучения.

    - Обновление весов: Веса обновляются в направлении, противоположном градиенту ошибки. Это помогает минимизировать ошибку предсказания.

4. Повторение процесса: Этот процесс повторяется для всех слов в предложении и для всех предложений в корпусе. Модель проходит через корпус несколько раз, как определено параметром epochs.

Пример использования

```
from gensim.models import Word2Vec

def vectorize_word2vec(tokens):
model = Word2Vec(sentences=tokens, vector_size=100, window=5, min_count=1, workers=4)
word_vectors = model.wv
return word_vectors
```

Пример преобразования текстовых данных:

- До: ["This is a great video "I love this content"]
- После: Векторы слов: "this": [...], "is": [...], "a": [...], "great": [...], "video": [...], "I": [...], "love": [...], "content": [...]

2.4.4. Ocena

После векторизации текста с использованием методов TF-IDF и Word2Vec важно провести оценку их эффективности и понять, насколько хорошо они справляются с задачей представления текстовых данных. В данной секции будут рассмотрены критерии оценки и результаты применения каждого из методов на наших данных.

Критерии

Для оценки работы методов TF-IDF и Word2Vec используются следующие критерии:

- **Семантическая значимость:** Насколько хорошо метод улавливает семантические связи между словами и контекст их использования.
- **Скорость вычислений:** Время, необходимое для векторизации текстов.
- **Простота интерпретации:** Насколько легко интерпретировать результаты векторизации и использовать их в дальнейших анализах.
- **Применимость к задачам машинного обучения:** Насколько хорошо векторы подходят для задач классификации, кластеризации и других методов машинного обучения.

Оценка TF-IDF

Оценка Word2Vec

2.5. Кластеризация

2.5.1. Opis

Следующим этапом обработки данных является кластеризация, которая использует векторезированные комментарии и разбивает их на кластеры. Реализовано это будет с помощь дву методов K-means и DBSCAN используя библиотеку sklearn.

2.5.2. K-means

Алгоритм K-means разбивает набор данных на непересекающиеся кластеры, каждый из которых описывается средним значением (центроидом) образцов в кластере. K-means стремится выбрать центроиды, минимизируя инерцию, или критерий суммы квадратов внутри кластеров.

Описание алгоритма

Алгоритм K-means можно понять через три основных шага. Первый шаг заключается в выборе начальных центроидов. Обычно используется простой метод выбора случайных образцов из набора данных. После инициализации K-means выполняется последовательное выполнение двух других шагов. Первый шаг заключается в присвоении каждого образца ближайшему центроиду. Второй шаг создает новые центроиды, вычисляя среднее значение всех образцов, принадлежащих каждому предыдущему центроиду. Разница между старыми и новыми центроидами вычисляется, и алгоритм повторяет эти два последних шага до тех пор, пока эта разница не станет менее определенного порога. Другими словами, алгоритм повторяется, пока центроиды не перестанут существенно перемещаться.

Как и многие алгоритмы, K-means подвержен проблеме локальных минимумов, что сильно зависит от начальной инициализации центроидов. Обычно алгоритм запускается несколько раз с различными инициализациями центроидов.

Параметры

Для настройки алгоритма K-means используются следующие параметры:

- n_clusters: Количество кластеров (K), на которые нужно разделить данные.
- init: Метод инициализации центроидов.
- max_iter: Максимальное количество итераций для одного запуска K-means.
- tol: Порог для остановки алгоритма. Если разница между старыми и новыми центроидами меньше этого значения, алгоритм останавливается.
- n_init: Количество запусков алгоритма с различными инициализациями. Результат с наименьшей инерцией будет выбран как окончательный.
- random_state: Начальное значение генератора случайных чисел для воспроизводимости результатов.
- algorithm: Алгоритм для вычисления K-means. Возможные значения включают 'auto', 'full', и 'elkan'.

Пример использования

```
kmeans = KMeans(n_clusters=2, init='k-means++', max_iter=300, n_init=10, random_state=0)

# Обучение модели
kmeans.fit(X)

# Предсказание кластеров для данных
y_kmeans = kmeans.predict(X)
```

Пример преобразования данных: были точки, а теперь число которое показывет в каком кластере.

2.5.3. DBSCAN

Алгоритм DBSCAN (Density-Based Spatial Clustering of Applications with Noise) рассматривает кластеры как области высокой плотности, отделенные от областей низкой плотности. Это позволяет DBSCAN обнаруживать кластеры любой формы, в отличие от K-means, который предполагает, что кластеры имеют выпуклую форму. Основной компонент DBSCAN - это понятие ядерных образцов, которые представляют собой образцы в областях высокой плотности.

Параметры

Парметры алгоритам DBSCAN:

- eps: Максимальное расстояние между двумя образцами, при котором один считается соседом другого. Это параметр радиуса для определения плотности области.
- min_samples: Минимальное число образцов (включая сам образец), необходимое для того, чтобы область считалась плотной, т.е. чтобы образец стал ядерным.
- metric: Метрика для вычисления расстояний между точками (по умолчанию 'euclidean'). Это может быть любая метрика, поддерживаемая функцией scipy.spatial.distance.pdist.
- algorithm: Алгоритм, используемый для вычисления ближайших соседей ('auto', 'ball_tree', 'kd_tree', 'brute').
- leaf_size: Размер листа, передаваемый алгоритмам 'ball_tree', 'kd_tree'. Это может повлиять на скорость построения и запроса ближайших соседей.
- p: Степень Минковского расстояния, когда используется метрика 'minkowski' (по умолчанию p=2, что соответствует евклидову расстоянию).

Описание алгоритма

Алгоритм DBSCAN реализуется следующим образом:

1. Определяются ядерные образцы: Образцы, для которых количество соседей в пределах eps не меньше min_samples.
2. Строятся кластеры: Для каждого ядерного образца строится кластер путем нахождения всех его соседей, которые также являются ядерными образцами, и их собственных соседей и так далее. Каждый кластер также включает не-ядерные образцы, которые являются соседями ядерных образцов в кластере.
3. Выделяются выбросы: Все образцы, которые не являются ядерными и не находятся на достаточном расстоянии от ядерных образцов, считаются выбросами.

Алгоритм DBSCAN детерминирован и всегда генерирует одинаковые кластеры при одинаковых входных данных в одном и том же порядке. Однако результаты могут отличаться при предоставлении данных в другом порядке. Это происходит из-за того, что порядок обработки данных влияет на то, каким образом назначаются образцы кластерам.

Пример использования

```
X = np.array([[0, 1], [1, 1], [2, 1], [3, 1], [4, 1],
              [0, 0], [1, 0], [2, 0], [3, 0], [4, 0]])

# Создание модели t-SNE
tsne = TSNE(n_components=2, perplexity=5, n_iter=300, random_state=0)

# Преобразование данных
X_embedded = tsne.fit_transform(X)
```

Пример преобразования данных: были точки, а теперь число которое показывет в каком кластере.

2.5.4. Выбор параметров

2.6. Визуализация

2.6.1. Opis

Заключителый процесс анализа это визуализация данных. Итоговый результат представляет из себя **2D Plot**, где каждая точка будет представлять некоторые текстовые данные (комментарии) с опрелённым цветом, соотвутсвующим цвету кластера, которому она принадлежит, или же цвету отсутсвия кластера как в случае DBSCAN. Так же на графике будут визуализироваться центра каждого кластера в виде красного крестика. Для удобства навигации и проверки результатов при наведении на точку(коментарий) или крустик(центр кластера) будет высчевиваться информация с содержанием элемента, которая репрезентуется с помощью библиотеки **mplcursor**. В случае точки это содержание обработоного комментария. В случае крестика репрезентуется центр кластера, которая с помощью метода обратного векторизации будет пытаться представить данную точку как слово.

2.6.2. Снижение размерности

Для возможности визуализации векторизированых данных в качестве 2D точек необходимо воспользоваться методом снижения размерности (T-SNE). Данный метод является нелейным методом выделения признаков. Изначально трудно определить за что отвечают признаки, то выбрать какие-то наиболее подходящие признаки невозможно. Нелейный метод поможет нам сохранить неленейные связи.

2.6.3. T-SNE

Параметры

- n_components: Количество измерений, в которое нужно проецировать данные (обычно 2 или 3 для визуализации).
- perplexity: Параметр, связанный с количеством ближайших соседей. Он влияет на баланс между локальной и глобальной структурой данных.
- early_exaggeration: Параметр, который контролирует расстояния между точками в начальной фазе оптимизации.
- learning_rate: Скорость обучения для градиентного спуска.
- n_iter: Количество итераций для оптимизации.
- metric: Метрика для вычисления расстояний между точками (по умолчанию 'euclidean').
- init: Метод инициализации ('random' или 'pca').

- random_state: Начальное значение генератора случайных чисел для воспроизводимости результатов.

Описание алгоритма

Алгоритм t-SNE выполняется следующим образом:

1. Преобразование расстояний в вероятности: Для каждой точки рассчитываются вероятности Пирсона, которые определяют вероятность того, что соседняя точка будет выбрана в качестве соседа.

2. Определение целевых вероятностей: На низкоразмерной проекции рассчитываются вероятности Q, аналогичные вероятностям Пирсона, но в другой размерности.

3. Минимизация расхождения (Kullback-Leibler divergence): Градиентный спуск используется для минимизации расхождения между распределениями P и Q, что приводит к сохранению близости точек.

Пример использования

```
X = np.array([[1, 2], [2, 2], [2, 3],
              [8, 7], [8, 8], [25, 80]])

# Создание модели DBSCAN
dbscan = DBSCAN(eps=3, min_samples=2)

# Обучение модели
dbscan.fit(X)

# Предсказание кластеров для данных
labels = dbscan.labels_
```

Пример преобразования данных: **тут фотку**

Глава 3

Rozdzial 3

3.1. Описание приложения

3.1.1. Рабочее окно

Графическое приложение представляет собой окно с меню-баром, содержащим два поля ("File"и "Comments"), и основной зоной, где располагается таб-меню. Каждый таб представляет собой отдельное рабочее пространство (workspace). Рассмотрим эти элементы детальнее.

3.1.2. Меню

Меню-бар позволяет быстро навигировать по рабочему окну в любой момент. В меню-баре реализованы следующие функции:

- Открытие диалогового окна для создания нового рабочего пространства ("File> "New Project").
- Открытие диалогового окна для получения комментариев с YouTube ("Comments> "YouTube").

Рассмотрим более подробно использование этих функций:

Создавние workspace

Для создания нового рабочего пространства необходимо указать имя рабочей области (Workspace name) и файл-источник (Source file) с текстовыми данными. Вот как выглядит пустое окно **CreateWorkspaceWindowEmpty**. Пользователю необходимо ввести любое имя для рабочего пространства, а также указать путь к текстовому файлу с данными. Это можно сделать вручную или выбрать файл с помощью диалогового окна FileDialog из библиотеки PyQt, что позволяет найти необходимый файл на компьютере. Пример правильно заполненного окна **CreateWorkspaceWindowFill**.

Получение комментариев

Для получения комментариев используется диалоговое окно, в котором необходимо указать ссылку на интересующее видео, как показано на скриншоте **CommentsWindowStep1**. Если видео найдено, то в поле Video Info появится информация о нем, позволяющая пользователю убедиться, что это именно то видео, комментарии которого необходимо получить. Далее пользователю остается только нажать на кнопку Download comments, которая сохранит все комментарии.

3.1.3. Workspace

После создания хотя бы одного рабочего пространства появляется меню навигации по вкладкам, где каждая вкладка соответствует отдельному рабочему пространству. По этим вкладкам можно свободно переключаться и закрывать их. Содержимое каждой вкладки включает:

- Поле для графиков.
- Кнопки Previous и Next для переключения графиков в поле для графиков.
- Кнопку Text для открытия диалогового окна управления обработкой текста.
- Кнопку Vectorization для открытия диалогового окна управления векторизацией.
- Кнопку Cluster для открытия диалогового окна управления кластеризацией.
- Кнопку Apply для применения всех выбранных параметров и открытия окна с результатами кластеризации.

Поле для графиков

Поле для графиков в каждый момент времени может отображать один из трех предложенных графиков:

- График частот слов (freq). Описан ранее в
- График длин текста(length). Описан ранее в
- График ближайших соседей (k-nearest). Описан ранее в

С помощью кнопок Previous и Next графики можно переключать соответственно вперед и назад.

3.1.4. Text

Кнопка Text открывает диалоговое окно, в котором пользователь может указать параметры для обработки текста. Вот как выглядит окно по умолчанию **TextProcessingWindowStandard**. По умолчанию все параметры установлены на значение true, то есть используются. Пользователь может выбрать один из предложеных опций обработки текста:

- Alpha - примитиваная обработка описанная ранее
- Stop-words - удаленее стоп слов описаная ранее
- Lematize - лемматизациия описаная ранее

Vectorization

Кнопка Vectorization открывает диалоговое окно, в котором пользователь может выбрать один из предложенных методов векторизации и указать параметры для выбранного метода. Вот как выглядит окно по умолчанию **VectorizationWindowStandard**. По умолчанию выбран метод TF-IDF с параметрами по умолчанию. Далее описан список доступных методов и параметров для каждого метода:

- Метод TF-IDF. Доступные параметры для настройки:
  - max_df: Диапозон допустимых значений ()
  - min_df: Диапозон допустимых значений ()
- Методы Word2vec. Доступные парметры для настройки:

  - vector_size: Диапозон допустимых значений ()
  - window: Диапозон допустимых значений ()
  - min_count: Диапозон допустимых значений ()
  - sg: Диапозон допустимых значений ()

Cluster

Кнопка Cluster открывает диалоговое окно, в котором пользователь может выбрать один из предложенных методов кластеризации и указать параметры для выбранного метода. Вот как выглядит окно по умолчанию **ClusterWindowStandard**. По умолчанию выбран метод K-means с параметрами по умолчанию. Далее описан список доступных методов и параметров для каждого метода:

- Метод K-means. Доступные параметры для настройки:
  - n_cluster: Диапозон допустимых значений ()
  - max_iter: Диапозон допустимых значений ()
  - n_iter: Диапозон допустимых значений ()
- Методы DBSCAN. Доступные парметры для настройки:
  - min_samples: Диапозон допустимых значений ()
  - eps: Диапозон допустимых значений ()

Result Window

Кнопка Apply открывает новое окно с результатами проделанной работы. Данное окно включает поле с графиком (2D plot) и панель инструментов (toolbar), где пользователь может воспользоваться готовыми функциями, предложенными библиотекой Matplotlib, для навигации по графику и других операций. Окно сохраняется, и пользователь может анализировать результаты, закрывать окно или сворачивать его и продолжать работу. Для упрощения навигации по всем окнам с результатами (ResultWindows) они будут иметь уникальные названия, чтобы можно было легко идентифицировать нужное окно.

3.2. Пример использования

3.2.1. Получение текстовых данных

3.2.2. Создание Workspace

3.2.3. Выбор методов и параметров

Возможных камбинаций методов и парметров может быть большое количество количество и достаточно сложно предугадать итоговый результат работы. Подробно рассмотрим как каждый элемент влияет влияет на тестовых данных:

3.2.4. Текстовая обработка

Далее будет расмотренно использование всех трех методов, то есть активен при обаботке будет только один. Оцениновать эфиктивность этих методов можно с помощью графкиков (freq, lenght). Сравнивая графики исходных данных и графики после конкретного метода обработки.

Alpha

Исходные графики, графики после. Визуальны видны изменения в длинее общий длине комментариев, все они стали немного меньше, более это заметно на графике freq где мы можем увидеть теперь более встречаемые слова.

stop-words

Исходные графики, графики после. Визуально графики недолжны были сильно измениться ведь шум от alpha не дает это сделать. Однако в комбинации с alpha обработкой можено заметить что слова которые остовались только после alpha обработки исчезли так как ("the "in "a") считаются наиболее встрчеающимися словами в анлглийском языке. Однако stop-words включает в себя эти слова.

lemmatize

При использовании только лемматизации сложно заметить какое-то весомое влияние этой функции. Более заметно оно станет только при использовании всех трех методоов. Когда особо встречаемыае смогут обьединиться так можно заметить на примере (human , humans) обьединились в одну группу.

3.2.5. Векторизация

При дальнейшем расмотрении работы этих методов, мы будем расматривать их отдельно. Рассмотрим как работает каждый из этих метдоов на необработаных текстовых данны и обработаных, а так же рассмотрим влияние изменений параметров. Для анализа наших результатов нам потребуются полностью выполнить весь процес анализа данных, чтобы визуализировать наши векторезированные данные, так как необходмо выбрать метод кластеризации будет использоваться метод K-means с 1 кластером.

TF-IDF

При исполльзование на необработаных данных При исполльзование на обработаных данных Изменяем параметры min , max Замечаем типичные результаты с центром и то что рядом стоящие точки никак не связаны.

Word2Vec

При исполльзование на необработаных данных При исполльзование на обработаных данных Изменяем параметры vector_size , window, min_count, sg Замечаем типичные результаты (при увелечения вектора как буд-то все растягивается и становиться дальше).

3.2.6. Кластеризация

Далее расмотрим методы кластеризации, расмотрим их одтельно. Расмотрим их на обработаных данных с зафиксироваными параметрами векторизации word2vec.

k-means

Рассмотрим как работает увеличение количество кластеров. И остальных

DBSCAN

Рассмотрим как работает увелечение eps, min_pts и попробуем подобрать значения основываясь на графике k-nears

Заметим что сложно подобрать эти значения.

3.2.7. Интерпритиация результатов

Глава 4

Tytuł 4

Tekst.

4.1. Sekcja

Na Rysunku 4.1.

Рис. 4.1: Podpis (źródło: http://jakis.adres)

Podsumowanie

Podsumujac analiza danych tekstowych jest bardzo chujna, jest bardza prydatna dla analizy malych komentarzej.

Литература

[1] Autor, *Tytuł*, Wydawnictwo, Rok wydania.

[2] Tytuł strony, *Tytuł artykułu*, *http://adres.strony* (dostęp:20.10.2023).

[3] YouTube Data API *YouTube Data API*, *https://developers.google.com/youtube/v3/docs?hl=pl*

[4] Impossibility Theorem for Clustering *Theorem for Clustering*, *https://www.cs.cornell.edu/home/kleinber/nips15.pdf*

Листинги

Список таблиц

Список иллюстраций