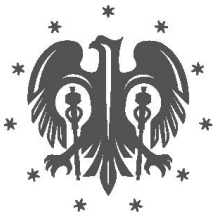

UMCS

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

0.1. Причина появления проекта

В современном мире количество текстовых данных растет с невероятной скоростью. Текстовые данные становятся неотъемлемой частью многих областей, включая социальные сети, форумы, блоги и другие платформы, где пользователи активно делятся своими мнениями и информацией. Одной из наиболее популярных платформ является YouTube, на которой миллионы пользователей ежедневно оставляют комментарии под видео. Анализ этих текстовых данных может предоставить ценную информацию для маркетинговых исследований, социальных исследований, а также для улучшения взаимодействия с аудиторией.

0.2. Идея проекта

Данный проект проект представляет собой GUI-приложение, предназначенное для анализа текстовых данных. В роли текстовых данных может выступать любые текстовые документы. Проект ориентирован на обработку комментариев с YouTube, что позволяет пользователям легко и быстро получать и анализировать эти данные. После выбора определенных методов обработки комментариев, которые пользователь может настраивать пользователь может оценить результат работы в качетсве итогово 2D графика на котором будут визуально разделенеы текстовые данные на групы(кластеры). Для помощи в качественном анализе данных пользователю будут доступны для настройки методы NLP, векторизации и кластеризации.

0.3. Для кого это

Таким образом, проект предоставляет мощный инструмент для качественного анализа текстовых данных, который может быть полезен как исследователям, так и специалистам в области маркетинга и социальных медиа.

Глава 1

Rozdzial 1

1.1. Klasterzyacja

1.1.1. Defenicja

Klasteryzacja (ang. clustering) to wielowymiarowa procedura analizy statystycznej, mająca na celu zbieranie danych zawierających informacje (cechy) o próbie obiektów, a następnie porządkowanie tych obiektów w stosunkowo jednorodne grupy. Zakres zastosowania klasteryzacji jest niezwykle szeroki: używa się jej w archeologii, medycynie, psychologii, chemii, biologii, administracji państwowej, filologii, antropologii, marketingu, socjologii, geologii i innych dyscyplinach. [4]

1.1.2. Cele klasteryzacji

Klasteryzacja jest najczęściej wykorzystywana do następujących celów:

1. Zrozumienie danych poprzez identyfikację struktury klastrów. Podział próby na grupy podobnych obiektów pozwala uprościć dalsze przetwarzanie danych i podejmowanie decyzji, stosując dla każdego klastra odpowiednią metodę analizy.

2. Kompresja danych. Jeśli pierwotna próba jest zbyt duża, można ją skompresować, zachowując tylko jednego najbardziej typowego przedstawiciela z każdego klastra. Zastępując wszystkie obiekty w każdym klastrze jednym najbardziej typowym przedstawicielem, można znacznie zmniejszyć objętość danych, zachowując jednocześnie główne cechy i strukturę pierwotnych danych.

3. Wykrywanie nowości (odchyleń lub anomalii). Klasteryzacja pozwala na wyróżnienie nietypowych obiektów, które nie mogą być przypisane do żadnego z klastrów. Te obiekty mogą być interesujące jako potencjalne anomalie lub odchylenia, wymagające dodatkowych badań, analiz lub uwagi.

1.1.3. Metody klasteryzacji

Nie ma powszechnie przyjętej klasyfikacji metod klasteryzacji, jednak można wyróżnić kilka grup podejść:

1. Algorytmy klasteryzacji oparte na grafach. Klasa ta obejmuje prymitywne algorytmy oparte na budowaniu grafu podobieństwa między obiektami. Obecnie są one praktycznie nie stosowane w praktyce.

2. Algorytmy klasteryzacji probabilistycznej. Algorytmy te przypisują każdemu obiektowi z próby treningowej prawdopodobieństwo przynależności do każdego z klastrów.

3. Hierarchiczne algorytmy klasteryzacji. Algorytmy te porządkują dane, tworząc hierarchię zagnieżdżonych klastrów.

4. Algorytmy oparte na gęstości danych:
    - K-średnich (K-means). Iteracyjny algorytm oparty na minimalizacji sumy kwadratowych odchyleń punktów klastrów od ich centrów.
    - Rozpowszechnianie podobieństwa (Affinity Propagation). Algorytm, który rozprzestrzenia wiadomości o podobieństwie między parami obiektów w celu wyboru typowych przedstawicieli każdego klastra.
    - Przesunięcie średniej (mean shift). Metoda wybierająca centroidy klastrów w obszarach o największej gęstości.
    - Klasteryzacja spektralna (spectral clustering). Metoda wykorzystująca wartości własne macierzy odległości do zmniejszenia wymiarowości przed zastosowaniem innych metod klasteryzacji.
    - DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Algorytm grupujący punkty w jeden klaster w obszarach o wysokiej gęstości i oznaczający odosobnione punkty jako szum.

1.1.4. Etapy klasteryzacji

Niezależnie od przedmiotu badań, zastosowanie analizy klastrów obejmuje następujące etapy:

1. Wybór próby do klasteryzacji. Zakłada się, że klasyfikować można tylko dane ilościowe, ponieważ wskaźniki liczbowe mogą być ilościowo oceniane.
2. Określenie zestawu zmiennych, na podstawie których obiekty w próbie będą oceniane, tj. przestrzeni cech. Cechy mogą być różnymi charakterystykami obiektów, które mają znaczenie dla badań.
3. Obliczanie wartości wybranej miary podobieństwa (lub różnicy) między obiektami. Celem jest określenie, jak blisko lub różnią się obiekty w próbie pod względem wybranych cech.
4. Zastosowanie metody analizy klastrów do stworzenia grup podobnych obiektów. Zakłada się wykorzystanie jednego z istniejących algorytmów klasteryzacji w taki sposób, aby obiekty w jednym klastrze były maksymalnie podobne do siebie, a obiekty z różnych klastrów maksymalnie się różniły.
5. Sprawdzenie wiarygodności wyników klasteryzacji. Może to być osiągnięte za pomocą różnych metod, takich jak wizualizacja klastrów, analiza stabilności klastrów i ocena jakości podziału.

1.1.5. Zastosowanie

Zastosowanie metody analizy klastrów jest z powodzeniem wykorzystywane w wielu dziedzinach i dyscyplinach. Metoda ta jest również szeroko stosowana w informatyce. Najbardziej udanymi przykładami wykorzystania klasteryzacji w pracy z danymi tekstowymi są:

- klasteryzacja wyników wyszukiwania
- grupowanie użytkowników o podobnych zainteresowaniach, co pomaga w personalizacji treści, rekomendacjach i targetowaniu w reklamie
- grupowanie tekstowych dokumentów według ich tematyki, takich jak artykuły prasowe, publikacje naukowe lub wiadomości w mediach społecznościowych

1.1.6. Ocena efektywności klasteryzacji

Problem oceny jakości w zadaniu klasteryzacji jest trudny do rozwiązania z co najmniej dwóch powodów. Po pierwsze, zgodnie z twierdzeniem o niemożności Kleinberga [3], nie istnieje optymalny algorytm klasteryzacji. Po drugie, wiele algorytmów klasteryzacji nie jest w stanie samodzielnie określić rzeczywistej liczby klastrów w danych; najczęściej liczba klastrów jest zadawana na wejściu algorytmu i dobierana kilkoma uruchomieniami. Niemniej jednak, wyróżnia się dwie grupy metod oceny jakości klasteryzacji:

1. Zewnętrzne metody porównują wynik klasteryzacji z apriorycznie znanym podziałem na klasy. Przykładem zewnętrznej metody oceny może być indeks Randa. Metoda ta ocenia, jak wiele spośród par elementów, które znajdowały się w jednej klasie, oraz tych, które znajdowały się w różnych klasach, zachowało ten stan po zastosowaniu algorytmu klasteryzacji.

2. Wewnętrzne metody oceniają jakość klasteryzacji, korzystając jedynie z informacji zawartych w samych danych. Przykładem wewnętrznej metody jest metoda Spójności Klastra (Cluster Cohesion). Idea tej metody polega na tym, że im bliżej siebie znajdują się obiekty w klastrach, tym lepsze jest podzielenie. Można również wyróżnić metodę Separacji Klastra (Cluster Separation), która ocenia jakość klasteryzacji na podstawie odległości między obiektami różnych klastrów.

1.2. Przetwarzanie języka naturalnego

1.2.1. Defenicja

Przetwarzanie języka naturalnego (ang. Natural Language Processing, NLP) to dziedzina badań, która łączy metody uczenia maszynowego i lingwistyki matematycznej w celu opracowania algorytmów do analizy, rozumienia i generowania tekstu w językach naturalnych. [5]

1.2.2. Przedprzetwarzanie tekstu

Przedprzetwarzanie tekstu jest ważnym etapem NLP, który przekształca tekst w języku naturalnym w format wygodny do rozpoznawania przez algorytmy i dalszej pracy. Przedprzetwarzanie składa się z różnych etapów, które mogą się różnić w zależności od zadania i implementacji. Oto niektóre z najpopularniejszych podstawowych podejść:

- Konwersja wszystkich liter w tekście na małe lub wielkie litery
- Usuwanie cyfr (liczb) lub zamiana na tekstowy ekwiwalent (zwykle używane są wyrażenia regularne)
- Usuwanie znaków białych (whitespaces)
- Tokenizacja (zwykle realizowana na podstawie wyrażeń regularnych)
- Usuwanie słów stop
- Stemming
- Lematyzacja
- Wektoryzacja

Poniżej bardziej szczegółowo opisane są trzy ostatnie metody.

1.2.3. Stemming

Liczba poprawnych form wyrazowych, których znaczenia są podobne, ale pisownia różni się sufiksami, przedrostkami, końcówkami itp., jest bardzo duża, co utrudnia tworzenie słowników i dalsze przetwarzanie. Stemming pozwala sprowadzić słowo do jego podstawowej formy. Istota podejścia polega na znalezieniu rdzenia słowa, w tym celu z końca i początku słowa kolejno odcinane są jego części. Reguły odcinania dla stemmera tworzone są z góry i najczęściej stanowią wyrażenia regularne, co sprawia, że podejście to jest pracochłonne, ponieważ przy dodawaniu kolejnego języka potrzebne są nowe badania lingwistyczne. Drugą wadą podejścia jest możliwa utrata informacji podczas odcinania części, na przykład możemy stracić informację o części mowy.

1.2.4. Lematyzacja

Podejście to stanowi alternatywę dla stemmingu. Główna idea polega na sprowadzeniu słowa do formy słownikowej — lematu. Przykład lematyzacji dla języka polskiego:

- dla rzeczowników — mianownik liczby pojedynczej;
- dla przymiotników — mianownik liczby pojedynczej, rodzaj męski;
- dla czasowników, imiesłowów, rzeczowników odczasownikowych — czasownik w bezokoliczniku aspektu niedokonanego.

1.3. Wektoryzacja

1.3.1. Opis

Większość modeli matematycznych działa w przestrzeniach wektorowych o dużych wymiarach, dlatego konieczne jest przekształcenie tekstu w przestrzeń wektorową. Głównym celem wektoryzacji tekstu jest stworzenie reprezentacji danych tekstowych, która zachowuje informacje semantyczne i syntaktyczne o tekście, a jednocześnie nadaje się do wykorzystania w modelach matematycznych. Pozwala to na efektywną analizę i przetwarzanie danych tekstowych za pomocą algorytmów i metod uczenia maszynowego. [6]

1.3.2. Celi wektoryzacji

Główne celi wektoryzacji tekstu obejmują:

- Reprezentowanie tekstu do analizy przez algorytmy maszynowe.
- Znajdowanie podobieństw między tekstami lub kategoryzowanie tekstów według tematyki.
- Rozwiązywanie zadań klasyfikacji, klasteryzacji lub regresji na podstawie danych tekstowych.
- Tworzenie modeli uczenia maszynowego do automatycznego uogólniania informacji tekstowych.

1.3.3. Metody wektoryzacji

Pierwszym krokiem w wektoryzacji tekstu jest określenie przestrzeni cech, w której tekst będzie reprezentowany jako wektory. Każda cecha może reprezentować słowo, frazę, symbol lub inny element tekstu. Istnieje kilka metod wektoryzacji tekstu, w tym:

- Bag of Words (BoW): Każdy dokument jest reprezentowany jako wektor, gdzie każdy element odpowiada poszczególnemu słowu, a wartościami są częstotliwości występowania słów w dokumencie.
- TF-IDF (Term Frequency-Inverse Document Frequency): Metoda ta uwzględnia nie tylko częstotliwość występowania słów w dokumencie (TF), ale także odwrotną częstotliwość występowania słowa we wszystkich dokumentach korpusu (IDF), co pozwala wyróżnić słowa kluczowe.
- Word Embeddings: Są to metody oparte na trenowaniu sieci neuronowych, które przekształcają słowa w wektory w przestrzeni ciągłej, zachowując ich właściwości semantyczne.

1.3.4. Zastosowanie

Wektoryzacja tekstu znajduje zastosowanie w wielu dziedzinach, w tym:

- Analiza nastrojów i sentymentów w mediach społecznościowych i recenzjach.
- Klasyfikacja tekstów według tematyki lub kategorii.
- Systemy rekomendacyjne oparte na opisach tekstowych.
- Automatyczne tłumaczenie tekstu i implementacja chatbotów.

1.4. Redukcja wymiarowości

1.4.1. Definicja

Redukcja wymiarowości to proces zmniejszania liczby cech (wymiarów) w zbiorze danych, przy jednoczesnym zachowaniu jak największej ilości informacji. Często stosowana jest w uczeniu maszynowym, gdzie głównym celem jest eliminacja nadmiarowych, nieinformatywnych lub mało istotnych cech, które mogą obniżać skuteczność modelu. Po takim przekształceniu model staje się prostszy, co zmniejsza rozmiar zbioru danych w pamięci i przyspiesza działanie algorytmów. Redukcja wymiarowości jest również używana do uproszczenia analizy i wizualizacji danych, na przykład poprzez redukcję do niskich wymiarów takich jak 2D lub 3D. [7]

1.4.2. Metody redukcji wymiarowości

Redukcja wymiarowości może być realizowana metodami wyboru cech (ang. feature selection) lub wydobywania cech (ang. feature extraction).

Wybór cech

Metody wyboru cech pozostawiają pewne podzbiory oryginalnego zestawu cech, eliminując cechy nadmiarowe i mało informatywne. Główne zalety tego rodzaju algorytmów to:

- Zmniejszenie prawdopodobieństwa przeuczenia
- Zwiększenie dokładności predykcji modelu
- Skrócenie czasu uczenia
- Zwiększenie semantycznego zrozumienia modelu.

Wydobywanie cech

Innym sposobem zmniejszenia wymiarowości danych wejściowych jest wydobywanie cech. Te metody tworzą z oryginalnych cech nowe, które nadal w pełni opisują przestrzeń zbioru danych, ale zmniejszają jego wymiarowość, tracąc na reprezentatywności danych, ponieważ staje się niejasne, za co odpowiadają nowe cechy. Metody wydobywania cech można podzielić na liniowe i nieliniowe.

Metody wydobywania cech

Liniowe metody opierają się na założeniu, że zależność między zmiennymi niezależnymi a zmienną zależną jest liniowa. Oznacza to, że zmiana jednej zmiennej prowadzi do proporcjonalnej zmiany innej. Jedną z najbardziej znanych metod liniowego wydobywania cech jest PCA (Principal Component Analysis). Główną ideą tej metody jest znalezienie hiperpłaszczyzny, na którą przy ortogonalnej projekcji wszystkich cech maksymalizowana jest wariancja.

Nieliniowe metody pozwalają modelować bardziej złożone i nieliniowe zależności między zmiennymi. Algorytm t-SNE (t-distributed Stochastic Neighbor Embedding) stara się zachować względne odległości między punktami w oryginalnej przestrzeni wysokowymiarowej, a następnie próbuje odtworzyć to rozkład w przestrzeni niskowymiarowej.

1.4.3. Zastosowanie

Metoda redukcji wymiarowości jest często i skutecznie stosowana w uczeniu maszynowym, przetwarzaniu obrazów i widzeniu komputerowym, bioinformatyce i genomice, analizie finansowej i handlu oraz wielu innych dziedzinach. Zmniejszenie liczby cech pozwala na optymalizację przetwarzania danych.

Глава 2

Rozdzial 2

2.1. Aplikacja

2.1.1. Opis aplikacji

Ten projekt ma na celu stworzenie narzędzia do analizy danych tekstowych, obejmującego wstępne przetwarzanie tekstu, wektoryzację i klasteryzację. Głównym celem aplikacji jest umożliwienie użytkownikowi elastycznego dostosowania i wykorzystania różnych metod przetwarzania danych do rozwiązywania konkretnych zadań.

2.1.2. Technologie

Do realizacji projektu wybrano język programowania Python. Posiada on szereg zalet: [**?**]

- Bogaty zestaw bibliotek do implementacji różnych metod, takich jak wstępne przetwarzanie tekstu, wektoryzacja i klasteryzacja.
- Wysoka efektywność w analizie danych dzięki bibliotekom takim jak NumPy, scikit-learn.
- Społeczność i dokumentacja, które ułatwiają rozwój i utrzymanie projektów.

2.1.3. GUI

Główną ideą aplikacji jest stworzenie wygodnego narzędzia do analizy danych tekstowych, w którym użytkownik może samodzielnie wybierać i dostosowywać metody przetwarzania do konkretnych zadań za pomocą interfejsu graficznego. Do opracowania interfejsu graficznego wykorzystano biblioteki PyQt i Matplotlib:

- PyQt: Umożliwia tworzenie intuicyjnych interfejsów użytkownika.
- Matplotlib: Wykorzystywana do wizualizacji danych, co pozwala użytkownikowi łatwo interpretować wyniki analizy.

Interfejs graficzny umożliwia użytkownikowi ładowanie danych, konfigurowanie parametrów przetwarzania tekstu, wektoryzacji i klasteryzacji oraz wizualizację wyników.

2.1.4. OOP

Głównym komponentem mojej aplikacji jest klasa **Workspace**, która stanowi kontener przechowujący parametry dla trzech głównych procesów:

- Przetwarzanie tekstu: Obejmuje metody wstępnego przetwarzania, takie jak tokenizacja, usuwanie stop-słów i lematyzacja.
- Wektoryzacja: Przekształca dane tekstowe w wektory liczbowe przy użyciu metod takich jak TF-IDF lub Word2Vec.

- Klasteryzacja: Implementuje algorytmy, takie jak K-means lub DBSCAN, do grupowania danych tekstowych.

Klasa **Workspace** zapewnia strukturalne przechowywanie i zarządzanie parametrami każdego etapu, co upraszcza proces konfiguracji i wykonania analizy.

2.2. Dane tekstowe

2.2.1. Opis

Aby zrealizować ten projekt, potrzebne będą dane tekstowe, które w tym przypadku będą komentarzami z YouTube. Do pozyskiwania informacji o filmach i powiązanych komentarzach będziemy używać YouTube Data API v3. API to jest darmowe i łatwe w użyciu. Aby rozpocząć pracę, należy zarejestrować się i utworzyć projekt w Google Cloud, a następnie aktywować YouTube Data API. Po tym wygenerowany zostanie klucz API, który będziemy używać do uwierzytelniania.

2.2.2. Zapytania API

W projekcie wykorzystane będą dwa zapytania HTTP:

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

Pierwsze zapytanie służy do pobierania komentarzy do konkretnego filmu, a drugie do uzyskiwania informacji o filmie. Szczegółowe informacje można znaleźć w dokumentacji [8]. Etapy pobierania komentarzy:

1. Użytkownik wprowadza w interfejsie graficznym (GUI) link do interesującego go filmu.
2. Program wyodrębnia **video_id** z podanego linku.
3. Za pomocą **video_id** wykonywane jest zapytanie GET https://www.googleapis.com/youtube/v3/videos, aby uzyskać podstawowe informacje o filmie. To pozwala użytkownikowi upewnić się, że wybrany film odpowiada jego oczekiwaniom.
4. Po potwierdzeniu przez użytkownika, że film jest prawidłowy, wykonywane jest zapytanie GET https://www.googleapis.com/youtube/v3/commentThreads, aby pobrać komentarze do tego filmu.
5. Uzyskane komentarze są zapisywane w pliku tekstowym, gdzie każdy komentarz jest zapisany w oddzielnej linii. Nazwa pliku odpowiada unikalnemu identyfikatorowi filmu (**video_id**).

2.2.3. Przetwarzanie uzyskanych danych

Wyniki naszych zapytań będą dostarczane w formacie JSON. Szczegółowe informacje o strukturze odpowiedzi JSON można znaleźć w dokumentacji [8]. Do uzyskiwania informacji o filmie będą używane tylko następujące pola:

- title: tytuł filmu
- channelTitle: nazwa autora kanału
- thumbnails: URL miniaturki

Przy pobieraniu komentarzy potrzebne są tylko następujące pola:

- textDisplay: treść komentarza
- replies: kontener z odpowiedziami na komentarze

Pole **textDisplay** zawiera tekst komentarza, a pole **replies** jest kontenerem na odpowiedzi na komentarze. Każda odpowiedź w **replies.comments** również ma pole **textDisplay**, które możemy osobno przetwarzać. W końcowym wyniku otrzymamy listę wszystkich komentarzy, gdzie każdy element reprezentuje oddzielny komentarz.

2.2.4. Przechowywanie

Po uzyskaniu wszystkie komentarze będą przechowywane w osobnym katalogu **comments**, który zostanie utworzony w przypadku jego braku. Każdy plik z komentarzami będzie miał nazwę **video_id.txt** dla łatwej identyfikacji. Dla uproszczenia procesu przechowywania i dostępu do danych oraz zapewnienia ich kompatybilności z różnymi narzędziami analitycznymi, zdecydowano się na użycie zwykłego pliku tekstowego (.txt) do przechowywania komentarzy. Każdy komentarz będzie zapisany w oddzielnej linii pliku, co upraszcza strukturę danych i umożliwia łatwe ich odczytanie do analizy. Taki sposób przechowywania danych został wybrany z kilku powodów:

- **Prostota implementacji**: Pliki tekstowe łatwo tworzyć, czytać i edytować za pomocą wielu narzędzi programistycznych i języków programowania.
- **Kompatybilność**: Praktycznie wszystkie narzędzia analityczne i biblioteki mogą pracować z plikami tekstowymi, co ułatwia integrację danych w różnych etapach analizy.
- **Mały wpływ na analizę**: Format przechowywania nie ma znaczącego wpływu na sam proces analizy danych. Główne znaczenie ma zawartość komentarzy, a nie sposób ich przechowywania.

2.3. Przetwarzania tekstu

2.3.1. Defenicja

Następnym procesem jest przetwarzanie tekstu. Ważne jest, aby zrozumieć, jak wyglądają komentarze po ich pobraniu i zapisaniu z YouTube. Implementacja przetwarzania zostanie wykonana za pomocą biblioteki NLTK (Natural Language Toolkit) oraz wbudowanych funkcji Pythona. Dokumentację NLTK można znaleźć tutaj [9]. Przetwarzanie tekstu jest podzielone na kilka etapów:

- Przetwarzanie obowiązkowe, które jest wykonywane niezależnie od wybranych parametrów przez użytkownika.
- Przetwarzanie prymitywne, czyli usuwanie symboli i cyfr.
- Usuwanie stop-words.
- Lematyzacja.
- Analiza.

Wszystkie metody, oprócz przetwarzania obowiązkowego i analizy, są opcjonalne, co oznacza, że będą wykonywane tylko wtedy, gdy użytkownik wskaże, że chce używać tych metod.

2.3.2. Przetwarzanie obowiązkowe

Obowiązkowe etapy przetwarzania tekstu to zmiana wszystkich słów na małe litery oraz proces tokenizacji, który eliminuje potrzebę usuwania spacji. Za pomocą funkcji **nltk.word_tokenize** każde słowo zostanie podzielone na tokeny. Te etapy są obowiązkowe, ponieważ upraszczają dalszą pracę z tekstem i gwarantują poprawne działanie wielu późniejszych funkcji, które mogą być czułe na wielkość liter.

2.3.3. Przetwarzanie prymitywne

Przetwarzanie prymitywne polega na usuwaniu niealfabetycznych symboli lub słów. Ten proces nie jest obowiązkowy, jednak jego zastosowanie pomaga pozbyć się różnych języków znaczników, które YouTube używa do wizualizacji emotikonów lub narzędzi do zmiany tekstu.

Przykłady takich komentarzy:

```
"Great video! (tu musi byc emoji, ale nie wiem jak tu wkleic)"
"Check this out: <a href='https://example.com'>link</a>"
```

W tym projekcie jest to realizowane za pomocą wbudowanej funkcji Pythona **isalpha()**, która sprawdza, czy symbole lub słowa zawierają niealfabetyczne znaki.

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

2.3.4. Usuwanie stop-words

Ten etap polega na usuwaniu słów, które nie niosą znaczącej informacji. Typowymi przykładami takich słów są:

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

Usuwanie stop-words zmniejsza rozmiar tekstu i sprawia, że znaczące słowa są bardziej widoczne. Do tego celu używa się słownika stop-words z biblioteki NLTK, który w razie potrzeby można uzupełnić własnymi słowami.

```
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. Lematyzacja

Ostatnim opcjonalnym procesem jest lematyzacja, która sprowadza słowa do ich podstawowej formy. Jest to przydatne w sytuacjach, gdy trzeba zgrupować różne formy jednego słowa (np. "running "ran "runs"do "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. Analiza

Na końcu, niezależnie od wybranych przez użytkownika etapów, wszystkie tokeny są sprawdzane pod kątem niepustych wartości, aby uniknąć całkowicie pustych słów i wierszy. Następnie wszystkie tokeny są łączone w jeden ciąg znaków, oddzielony spacjami. Na tym etapie wynik przetwarzania jest zwracany i analizowany za pomocą funkcji analizy częstotliwości (**freq_analysis**) i analizy długości (**length_analysis**). Te funkcje będą używane do tworzenia wykresów, które pomogą użytkownikowi wizualnie ocenić częstotliwość występowania słów i długość komentarzy w celu podejmowania dalszych decyzji.

Analiza częstotliwości (freq analysis)

Ta funkcja używa przetworzonych ciągów znaków, tworząc pojedynczą tablicę ze wszystkimi słowami ze wszystkich komentarzy, i stosuje klasę **collections.Counter** w Pythonie. **Counter** jest używany do zliczania hashowalnych obiektów, takich jak ciągi znaków lub liczby, i dostarcza wygodne metody do pracy z danymi częstotliwości. W ten sposób funkcja generuje dwie tablice: tablicę słów i odpowiadającą jej tablicę liczby wystąpień każdego słowa w tekście. Te tablice będą

używane do tworzenia wykresu słupkowego (bar chart), pokazującego częstotliwość występowania słów.

Przykład implementacji algorytmu w projekcie:

```
        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]
```

Analiza długości (length analysis)

Ta funkcja również używa przetworzonych ciągów znaków komentarzy, określając liczbę słów w każdym komentarzu. Ostatecznie generowana jest tablica z liczbą słów, która będzie używana do tworzenia histogramu (histogram), pokazującego rozkład długości komentarzy.

Przykład implementacji algorytmu w projekcie:

```
        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. Podsumowanie

Wszystkie etapy przetwarzania tekstu, od obowiązkowych do opcjonalnych, odgrywają ważną rolę w przygotowaniu danych do dalszej analizy. Ich poprawne wykonanie gwarantuje strukturalizowany i wygodny do przetwarzania wygląd danych. Wizualizacja wykresów analizujących zawartość komentarzy pozwoli użytkownikom lepiej zrozumieć strukturę i treść komentarzy, a także podejmować uzasadnione decyzje na podstawie przedstawionych danych.

2.4. Wektoryzacja

2.4.1. Defenicja

Na tym etapie tekst jest gotowy do dalszego przetwarzania. Aby przeprowadzić analizę, konieczne jest przekształcenie komentarzy w formę wektorową (wektoryzacja). Zostaną użyte dwie metody: TF-IDF i Word2Vec. Te metody umożliwiają przedstawienie danych tekstowych w postaci wektorów liczbowych, co ułatwia dalszą analizę i przetwarzanie. Wektoryzacja będzie realizowana przy użyciu bibliotek **scikit-learn** [10] dla metody TF-IDF i **gensim** [11] dla metody Word2Vec.

Moje pytanie

Tu jak i w nastepnych sekcjach mam pytnasnie, bo nie wiem jak opisac dlaczego wybralem dokladnie te metody. Wybralem tak naprawde 2 najpopularniyjszy metody, i wybralem 2 bo chcial to porownywac w wyniku

2.4.2. TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) to miara statystyczna używana do oceny ważności terminu w dokumencie w odniesieniu do kolekcji dokumentów (korpusu). Jest szeroko stosowana w zadaniach przetwarzania języka naturalnego i wyszukiwania informacji. W bibliotece Scikit-learn istnieje wygodne narzędzie do obliczania TF-IDF: klasa TfidfVectorizer.

Parametry

- max_df: Liczba lub udział. Terminy, które występują w większej liczbie dokumentów niż podano, są ignorowane. Przydatne do usuwania powszechnie używanych słów.
- min_df: Liczba lub udział. Terminy, które występują w mniejszej liczbie dokumentów niż podano, są ignorowane. Przydatne do usuwania rzadkich słów.
- ngra_range: Krotka (min_n, max_n). Określa zakres dla ekstrakcji n-gramów.
- stop_words: Lista słów lub ciąg znaków określający słowa, które należy ignorować.

Wyjaśnienie algorytmu

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

Przykład użycia

```
from sklearn.feature_extraction.text import TfidfVectorizer

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

Przykład przekształcenia danych tekstowych:

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

Aby korzystać z TF-IDF, potrzebny jest korpus tekstów, który można przedstawić jako listę ciągów znaków, gdzie każdy ciąg znaków to osobny dokument.

2.4.3. Word2Vec

Word2Vec to algorytm opracowany do nauki wektorowych reprezentacji słów (embeddingów), które oddają semantyczne relacje między słowami. W bibliotece Gensim dostępny jest wygodny interfejs do pracy z Word2Vec.

Parametry

- sentences: Korpus tekstów, przedstawiony jako lista list słów.
- vector_size: Wymiarowość wektorowej reprezentacji słów.
- window: Maksymalna odległość między bieżącym a przewidywanym słowem w zdaniu.
- min_count: Minimalna częstotliwość słowa w korpusie, aby je uwzględnić.
- workers: Liczba wątków do równoległego uczenia.

Dodatkowe ustawienia:

- SG (Skip-Gram) i CBOW (Continuous Bag of Words): W Word2Vec są dwa główne podejścia architektoniczne — Skip-Gram i CBOW. Parametr sg w Gensim pozwala wybrać między nimi:
  - sg=0 używa CBOW (domyślnie).
  - sg=1 używa Skip-Gram.

- Negative Sampling: W Word2Vec używa się negatywnego próbkowania dla zwiększenia efektywności uczenia. Parametr negative określa liczbę "negatywnych"przykładów dla każdego pozytywnego przykładu.
- Learning Rate: Parametr alpha określa początkową szybkość uczenia, a min_alpha — minimalną szybkość uczenia, która zmniejsza się do tej wartości w trakcie uczenia..

Opis algorytmu

1. Inicjalizacja wag: Wagi sieci neuronowej są inicjalizowane losowymi wartościami. Są to wagi między warstwą wejściową a ukrytą, oraz między warstwą ukrytą a wyjściową.
2. Przejście do przodu (Forward Pass): Dla każdego słowa w zdaniu tworzony jest wektor wejściowy. W przypadku Skip-Gram ten wektor wejściowy jest używany do przewidywania słów kontekstowych. W przypadku CBOW słowa kontekstowe są używane do przewidywania bieżącego słowa.
    - Skip-Gram:
        - Słowo wejściowe jest kodowane jako jednowymiarowy wektor.
        - Jednowymiarowy wektor jest mnożony przez wagi między warstwą wejściową a ukrytą, aby uzyskać ukrytą reprezentację.
        - Ukryta reprezentacja jest mnożona przez wagi między warstwą ukrytą a wyjściową, aby przewidzieć słowa kontekstowe.
    - CBOW:
        - Słowa kontekstowe są kodowane jako jednowymiarowe wektory.
        - Jednowymiarowe wektory są sumowane i mnożone przez wagi między warstwą wejściową a ukrytą, aby uzyskać ukrytą reprezentację.
        - Ukryta reprezentacja jest mnożona przez wagi między warstwą ukrytą a wyjściową, aby przewidzieć bieżące słowo.
3. Propagacja wsteczna (Backpropagation): Błąd przewidywania jest obliczany przez porównanie przewidywanego słowa z faktycznym. Następnie ten błąd jest propagowany wstecz przez sieć, aby zaktualizować wagi. Używa się metody stochastycznego spadku gradientu (SGD) i, w niektórych przypadkach, negatywnego próbkowania, aby przyspieszyć uczenie.
    - Aktualizacja wag: Wagi są aktualizowane w kierunku przeciwnym do gradientu błędu. To pomaga zminimalizować błąd przewidywania.
4. Powtarzanie procesu: Proces ten powtarza się dla wszystkich słów w zdaniu i dla wszystkich zdań w korpusie. Model przechodzi przez korpus kilka razy, zgodnie z wartością parametru epochs.

Przykład użycia

```
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
```

Przykład przekształcenia danych tekstowych:

- Przed: ["This is a great video "I love this content"]
- Po: Wektory słów: "this": [...], "is": [...], "a": [...], "great": [...], "video": [...], "I": [...], "love": [...], "content": [...]

2.4.4. Ocena

Po wektoryzacji tekstu za pomocą metod TF-IDF i Word2Vec ważne jest przeprowadzenie oceny ich skuteczności i zrozumienie, jak dobrze radzą sobie z zadaniem przedstawiania danych tekstowych. W tej sekcji zostaną omówione kryteria oceny i wyniki zastosowania każdej z metod na naszych danych.

Kryteria

Do oceny pracy metod TF-IDF i Word2Vec używane są następujące kryteria:

- **Znaczenie semantyczne:** Jak dobrze metoda uchwytuje semantyczne związki między słowami i kontekstem ich użycia.
- **Szybkość obliczeń:** Czas potrzebny na wektoryzację tekstów.
- **Łatwość interpretacji:** Jak łatwo interpretować wyniki wektoryzacji i używać ich w dalszych analizach.
- **Zastosowalność do zadań uczenia maszynowego:** Jak dobrze wektory nadają się do zadań klasyfikacji, klasteryzacji i innych metod uczenia maszynowego.

Ocena TF-IDF

Tego nie mam w kodzie, moze to i nie jest potrzebne

Ocena Word2Vec

Jest nie wielka czesc, ale napisano to bylo przez chatGPT, zeby ja mog sprawdzic

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:

Рис. 3.1: Przyklad rys

- 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 = Итог

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

[4] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Inc., 1988.

[5] Daniel Jurafsky, James H. Martin, *Speech and Language Processing*, *https://web.stanford.edu/ jurafsky/slp3/ed3book.pdf*, Stanford University., 2024.

[6] C. C. Aggarwal, C. Zhai (Eds.), *Mining Text Data*, Springer, 2012.

[7] A. Y. Ng, M. I. Jordan, Y. Weiss, *On Spectral Clustering: Analysis and an Algorithm*, Advances in Neural Information Processing Systems, 14, 2001.

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

[9] Steven Bird, Ewan Klein, Edward Loper, *Natural Language Processing with Python*, O'Reilly Media, 2009.

[10] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, pp. 2825-2830, 2011.

[11] Radim Řehůřek, Petr Sojka, *Software Framework for Topic Modelling with Large Corpora*, In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45-50, 2010.

[12] G. Salton, A. Wong, C. S. Yang, *A Vector Space Model for Automatic Indexing*, Communications of the ACM, 18(11), pp. 613-620, 1975.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv preprint arXiv:1301.3781, 2013.

[14] J. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*, In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability (Vol. 1, No. 14, pp. 281-297), 1967.

[15] M. Ester, H. P. Kriegel, J. Sander, X. Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, In KDD (Vol. 96, No. 34, pp. 226-231), 1996.

[16] L. van der Maaten, G. Hinton, *Visualizing Data using t-SNE*, Journal of Machine Learning Research, 9(Nov), pp. 2579-2605, 2008.

[17] Mark Summerfield, *Rapid GUI Programming with Python and Qt*, Prentice Hall, 2007.

[18] John D. Hunter, *Matplotlib: A 2D Graphics Environment*, Computing in Science And Engineering, 9(3), pp. 90-95, 2007.

[19] Matplotlib Development Team, *mplcursors*, *https://mplcursors.readthedocs.io/*, (доступ: 20.10.2023).

Листинги

Список таблиц

Список иллюстраций