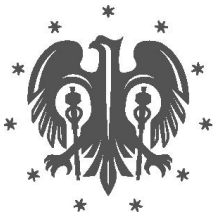

UMCS

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

Kierunek: Informatyka
Specjalność: Informatyka

Maksim Ryshko
nr albumu: 310030

Wykorzystanie metod klastrujących podczas analizy danych tekstowych

Using Clustering Methods in Text Data Analysis

Praca licencjacka
napisana w Wydziale Matematyki, Fizyki i Informatyki
pod kierunkiem dr. Anna Gajos-Balińska

Lublin rok 2024

Spis treści

Wstęp

0.1. Powód powstania projektu

W dzisiejszym świecie ilość danych tekstowych rośnie z niesamowitą prędkością. Dane tekstowe stają się nieodłączną częścią wielu obszarów, w tym mediów społecznościowych, forów, blogów i innych platform, gdzie użytkownicy aktywnie dzielą się swoimi opiniami i informacjami. Jedną z najpopularniejszych platform jest YouTube, na której miliony użytkowników codziennie zostawiają komentarze pod filmami. Analiza tych danych tekstowych może dostarczyć cenne informacje do badań marketingowych, społecznych oraz do poprawy interakcji z publicznością.

0.2. Idea projektu

W ramach niniejszej pracy opracowana będzie aplikacja GUI przeznaczona do analizy danych tekstowych. Dane tekstowe mogą pochodzić z dowolnych dokumentów tekstowych. Projekt skupia się na przetwarzaniu komentarzy z YouTube, co umożliwia użytkownikom łatwe i szybkie pobieranie oraz analizowanie tych danych. Po wybraniu dostępnych metod przetwarzania komentarzy, możliwa będzie ocena wyniku za pomocą dwuwymiarowego wykresu, na którym dane tekstowe zostaną wizualnie podzielone na grupy (klastry). Aby pomóc w jakościowej analizie danych, użytkownik będzie mógł dostosować metody przetwarzania języka naturalnego (NLP), wektoryzacji i klasteryzacji.

0.3. Dla kogo to

W ten sposób aplikacja stanie się potężnym narzędziem do jakościowej analizy danych tekstowych, które może być przydatne zarówno badaczom, jak i specjalistom z dziedziny marketingu i mediów społecznościowych.

Rozdział 1

Techniki analizy danych

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, w jakim stopniu obiekty w próbie są do siebie podobne lub różnią się 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 lepszy jest podział. 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. Wstępne przetwarzanie tekstu

Wstępne przetwarzanie tekstu jest ważnym etapem NLP, który przekształca tekst w języku naturalnym w format wygodny do rozpoznawania przez algorytmy i dalszej pracy. Przetwarzanie tekstu 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

Stemming to proces redukcji słowa do jego podstawowej formy, zwanej rdzeniem lub stemem. 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.

Rozdział 2

Wykorzystanie metody i technologii w aplikacji

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. Zastosowane technologie

Do realizacji projektu wybrano język programowania Python versji 3.10.2 . 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. Technologie tworzenia interfejsów graficznych

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 versji 5.15.10 i Matplotlib versji 3.9.0:

- **PyQt** umożliwia tworzenie intuicyjnych interfejsów użytkownika.
- **Matplotlib** jest wykorzystywany do wizualizacji danych.

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

2.1.4. Struktura aplikacji

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

- **Przetwarzanie tekstu** obejmuje metody wstępnego przetwarzania, takie jak tokenizacja, usuwanie stop-words 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 będzie używany 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 odpowiedzi wszystkie komentarze będą przechowywane w osobnym katalogu **comments**. Katalog zostanie utworzony, jeśli nie istnieje. 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) versji 3.8.1 oraz wbudowanych funkcji Pythona. Szczegóły biblioteki NLTK można znaleźć w documentacji [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 emotikon lub narzędzi do zmiany tekstu.

Przykłady takich komentarzy:

```
"Great video! :) :("
"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, jak widać na Listingu 2.1.

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

Listing 2.1: Funkcja usuwająca niealfabetyczne symbole

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. Przykład implementacji widać na Listingu 2.2.

```
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]
```

Listing 2.2: Funkcja usuwająca stop słowa

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"). Przykład implementacji widać na Listingu 2.3.

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

Listing 2.3: Funkcja lematyzująca tokeny

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 jest zaprezentowany na Listingu **??**.

```
1         from collections import Counter
2
3         def freq_analysis(tokens):
4           data = ' '.join(self.nlp_data)
5           words = nltk.word_tokenize(data)
6           word_freq = Counter(words)
7           words_counts_sorted = word_freq.most_common()
8           self.words_sorted = [word_count[0] for word_count in words_counts_sorted]
9           self.counts_sorted = [word_count[1] for word_count in words_counts_sorted]
```

Listing 2.4: Funkcja przeprowadzająca analizę częstotliwości tokenów

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 jest zaprezentowany na Listingu **??**.

```
1
2         def length_analysis(tokens):
3           word_counts = [len(nltk.word_tokenize(el)) for el in self.nlp_data]
4           self.max_size = max(word_counts) + 1
5           self.lengths = [0] * (self.max_size + 1)
6           for count in word_counts:
7             self.lengths[count] += 1
8           endTime = time.time()
```

Listing 2.5: Funkcja przeprowadzająca analizę długości tokenów

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 następnych sekcjach mam pytania, bo nie wiem, jak opisać dlaczego wybrałem dokładnie te metody. Wybrałem tak naprawdę dwie najpopularniejsze metody, i wybrałem dwie, bo chciałem je porównać pod względem wyników.

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. [12]

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

Przykład użycia metody TF-IDF w Listingu 2.6.

```
1
2       from sklearn.feature_extraction.text import TfidfVectorizer
3
4       def vectorize_tfidf(corpus):
5         vectorizer = TfidfVectorizer()
6         tfidf_matrix = vectorizer.fit_transform(corpus)
7       return tfidf_matrix
```

Listing 2.6: Funkcja wektoryzująca korpus za pomocą TF-IDF

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. [13]

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

Przykład użycia metody Word2Vec w Listingu 2.7.

```
1        from gensim.models import Word2Vec
2
3        def vectorize_word2vec(tokens):
4        model = Word2Vec(sentences=tokens, vector_size=100, window=5, min_count=1, workers=4)
5        word_vectors = model.wv
6        return word_vectors
```

Listing 2.7: Funkcja wektoryzująca za pomocą Word2Vec

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. Klasteryzacja

2.5.1. Defenicja

Kolejnym etapem przetwarzania danych jest klasteryzacja, która wykorzystuje zwektoryzowane komentarze i dzieli je na klastry. Zostanie to zrealizowane za pomocą dwóch metod: K-means i DBSCAN, z wykorzystaniem biblioteki sklearn [10, 15].

2.5.2. K-means

Algorytm K-means dzieli zbiór danych na nieprzekrywające się klastry, z których każdy opisany jest średnią wartością (centroidem) próbek w klastrze. K-means dąży do wyboru centroidów, minimalizując bezwładność, czyli kryterium sumy kwadratów wewnątrz klastrów [14].

Opis algorytmu

Algorytm K-means można zrozumieć przez trzy główne kroki. Pierwszy krok polega na wyborze początkowych centroidów. Zwykle używa się prostego metody losowego wyboru próbek z zestawu danych. Po inicjalizacji K-means wykonuje dwa kolejne kroki. Pierwszy krok polega na przypisaniu każdej próbki do najbliższego centroidu. Drugi krok tworzy nowe centroidy, obliczając średnią wartość wszystkich próbek przypisanych do każdego z poprzednich centroidów. Różnica między starymi a nowymi centroidami jest obliczana i algorytm powtarza te dwa ostatnie kroki, aż ta różnica nie będzie mniejsza od określonego progu. Innymi słowy, algorytm powtarza się, dopóki centroidy nie przestaną się znacząco przesuwać.

Jak wiele algorytmów, K-means jest podatny na problem lokalnych minimów, co silnie zależy od początkowej inicjalizacji centroidów. Zazwyczaj algorytm uruchamia się kilkakrotnie z różnymi inicjalizacjami centroidów.

Parametry

Do konfiguracji algorytmu K-means używa się następujących parametrów:

- n_clusters: Liczba klastrów (K), na które należy podzielić dane.
- init: Metoda inicjalizacji centroidów.
- max_iter: Maksymalna liczba iteracji dla jednego uruchomienia K-means.
- tol: Próg dla zatrzymania algorytmu. Jeśli różnica między starymi a nowymi centroidami jest mniejsza od tej wartości, algorytm zatrzymuje się.
- n_init: Liczba uruchomień algorytmu z różnymi inicjalizacjami. Wynik z najmniejszą bezwładnością zostanie wybrany jako ostateczny.
- random_state: Wartość początkowa generatora liczb losowych dla powtarzalności wyników.
- algorithm: Algorytm do obliczania K-means. Możliwe wartości to 'auto', 'full' i 'elkan'.

Przykład użycia

Przykład użycia metody K-means w Listingu 2.8.

```
1         kmeans = KMeans(n_clusters=2, init='k-means++', max_iter=300, n_init=10, random_state=0)
2         kmeans.fit(X)
3         return kmeans.labels_
```

Listing 2.8: Przykład użycia algorytmu K-Means

Przykład przekształcenia danych: tu przyklad

2.5.3. DBSCAN

Algorytm DBSCAN (Density-Based Spatial Clustering of Applications with Noise) traktuje klastry jako obszary o wysokiej gęstości, oddzielone od obszarów o niskiej gęstości. Dzięki temu DBSCAN może wykrywać klastry o dowolnym kształcie, w przeciwieństwie do K-means, który zakłada, że klastry mają kształt wypukły. Głównym elementem DBSCAN jest pojęcie punktów rdzeniowych, które reprezentują próbki w obszarach o wysokiej gęstości [15].

Parametry

Parametry algorytmu DBSCAN:

- eps: Maksymalna odległość między dwiema próbkami, przy której jedna jest uważana za sąsiada drugiej. Jest to parametr promienia do określania gęstości obszaru.
- min_samples: Minimalna liczba próbek (w tym sama próbka) potrzebna do uznania obszaru za gęsty, tj. do uznania próbki za rdzeniową.
- metric: Metryka do obliczania odległości między punktami (domyślnie 'euclidean'). Może to być dowolna metryka obsługiwana przez funkcję scipy.spatial.distance.pdist.
- algorithm: Algorytm używany do obliczania najbliższych sąsiadów ('auto', 'ball_tree', 'kd_tree', 'brute').
- leaf_size: Rozmiar liścia przekazywany algorytmom 'ball_tree' i 'kd_tree'. Może to wpłynąć na szybkość budowy i zapytania najbliższych sąsiadów.
- p: Wykładnik odległości Minkowskiego, gdy używana jest metryka 'minkowski' (domyślnie p=2, co odpowiada odległości euklidesowej).

Opis algorytmu

Algorytm DBSCAN działa w następujący sposób:

1. Określenie punktów rdzeniowych: Próbki, dla których liczba sąsiadów w obrębie eps nie jest mniejsza niż min_samples.

2. Tworzenie klastrów: Dla każdej próbki rdzeniowej tworzony jest klaster poprzez znalezienie wszystkich jej sąsiadów, którzy również są próbkami rdzeniowymi, oraz ich własnych sąsiadów itd. Każdy klaster obejmuje także próbki nie będące rdzeniowymi, które są sąsiadami próbek rdzeniowych w klastrze.

3. Wykrywanie szumów: Wszystkie próbki, które nie są rdzeniowe i nie znajdują się w wystarczającej odległości od próbek rdzeniowych, są uznawane za szumy.

Algorytm DBSCAN jest deterministyczny i zawsze generuje takie same klastry dla tych samych danych wejściowych w tej samej kolejności. Jednak wyniki mogą się różnić przy podaniu danych w innej kolejności. Wynika to z faktu, że kolejność przetwarzania danych wpływa na sposób, w jaki próbki są przypisywane do klastrów.

Przykład użycia

Przykład użycia metody DBSCAN w Listingu 2.9.

```
1
2       dbscan = DBSCAN(eps=0.1, min_samples=4)
3       dbscan.fit(X)
4       return dbscan.labels_
```

Listing 2.9: Przykład użycia t-SNE do osadzania danych

Przykład przekształcenia danych: tu przyklad

2.5.4. Dobór parametrów

Dobór odpowiednich parametrów dla algorytmów klasteryzacji, takich jak K-means i DBSCAN, jest kluczowy dla uzyskania satysfakcjonujących wyników. Poniżej przedstawione są niektóre techniki i wskazówki dotyczące doboru parametrów:

- K-means: Ważnym parametrem jest liczba klastrów (n_clusters). Można użyć metody "łokcia"(elbow method) lub średniego współczynnika sylwetki (silhouette score) do określenia optymalnej liczby klastrów. Parametry takie jak max_iter, tol oraz n_init mogą być dostosowane w celu poprawy zbieżności i stabilności algorytmu.

- DBSCAN: Kluczowymi parametrami są eps i min_samples. Wybór eps zależy od gęstości danych i można go oszacować na podstawie wykresu odległości K-najbliższych sąsiadów (K-distance plot). Parametr min_samples można ustawić w zależności od oczekiwanej minimalnej liczby punktów w klastrze.

this is new

2.6. Wizualizacja

2.6.1. Defenicja

Ostatnim etapem analizy jest wizualizacja danych. Końcowy wynik to wykres 2D, gdzie każdy punkt będzie reprezentował pewne dane tekstowe (komentarze) z określonym kolorem odpowiadającym klastrowi, do którego należy, lub kolorem braku klastra, jak w przypadku DBSCAN. Na wykresie będą również wizualizowane centroidy każdego klastra w postaci czerwonego krzyżyka.

2.6.2. Redukcja wymiarowości

Aby możliwa była wizualizacja zwektoryzowanych danych jako punktów 2D, konieczne jest skorzystanie z metody redukcji wymiarowości (T-SNE). Ta metoda jest nieliniową metodą ekstrakcji cech. Na początku trudno jest określić, za co odpowiadają cechy, więc wybór odpowiednich cech jest niemożliwy. Metoda nieliniowa pomoże nam zachować nieliniowe zależności [16].

2.6.3. T-SNE

Parametry

- n_components: Liczba wymiarów, na które należy zrzutować dane (zwykle 2 lub 3 dla wizualizacji).
- perplexity: Parametr związany z liczbą najbliższych sąsiadów. Wpływa na równowagę między lokalną a globalną strukturą danych.
- early_exaggeration: Parametr kontrolujący odległości między punktami na początkowym etapie optymalizacji.
- learning_rate: Szybkość uczenia się dla spadku gradientowego.
- n_iter: Liczba iteracji do optymalizacji.
- metric: Metryka do obliczania odległości między punktami (domyślnie 'euclidean').
- init: Metoda inicjalizacji ('random' lub 'pca').
- random_state: Początkowa wartość generatora liczb losowych dla reprodukowalności wyników.

Opis algorytmu

Algorytm t-SNE działa w następujący sposób:

1. Przekształcenie odległości na prawdopodobieństwa: Dla każdego punktu obliczane są prawdopodobieństwa Pearsona, które określają prawdopodobieństwo wybrania sąsiedniego punktu jako sąsiada.
2. Określenie docelowych prawdopodobieństw: Na niskowymiarowej projekcji obliczane są prawdopodobieństwa Q, podobne do prawdopodobieństw Pearsona, ale w innym wymiarze.
3. Minimalizacja dywergencji (Kullback-Leibler divergence): Używany jest spadek gradientowy do minimalizacji dywergencji między rozkładami P i Q, co prowadzi do zachowania bliskości punktów.

Przykład użycia

Przykład użycia metody T-SNE w Listingu 2.10.

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

    tsne = TSNE(n_components=2, perplexity=5, n_iter=300, random_state=0)

    X_embedded = tsne.fit_transform(X)
```

Listing 2.10: Przykład użycia t-SNE do osadzania danych

Przykład przekształcenia danych:
tu będzie zdjęcie

Rozdział 3

Rozdzial 3

3.1. Opis aplikacji

3.1.1. Okno pracy

Graficzna aplikacja składa się z okna z paskiem menu zawierającym dwa pola (”File”i ”Comments”), oraz główną strefą, w której znajduje się menu zakładek. Każda zakładka odpowiada osobnej przestrzeni roboczej (workspace). Przyjrzyjmy się tym elementom bliżej.

3.1.2. Menu

Pasek menu umożliwia szybką nawigację po oknie pracy w dowolnym momencie. W menu zaimplementowane są następujące funkcje:

- Otwieranie okna dialogowego w celu utworzenia nowej przestrzeni roboczej (”File¿ ”New Project”).
- Otwieranie okna dialogowego w celu pobrania komentarzy z YouTube (”Comments¿ ”YouTube”).

Podglądamy bardziej szczegółowo sposób korzystania z tych funkcji:

Tworzenie przestrzeni roboczej

Aby utworzyć nową przestrzeń roboczą, należy podać nazwę przestrzeni roboczej (Workspace name) i plik źródłowy (Source file) zawierający dane tekstowe. Początkowe okno **CreateWorkspaceWindowEmpty**. Użytkownik musi podać dowolną nazwę dla przestrzeni roboczej i wybrać ścieżkę do pliku tekstowego z danymi. Można to zrobić ręcznie lub wybrać plik za pomocą okna dialogowego FileDialog z biblioteki PyQt, co pozwala odnaleźć potrzebny plik na komputerze. Przykład wypełnionego poprawnie okna **CreateWorkspaceWindowFill**.

Pobieranie komentarzy

Do pobrania komentarzy służy okno dialogowe, w którym należy podać link do interesującego nas filmu, jak pokazano na zrzucie ekranu **CommentsWindowStep1**. Jeśli film zostanie znaleziony, w polu Video Info pojawią się informacje o nim, pozwalając użytkownikowi upewnić się, że to właśnie ten film, komentarze z którego chcemy pobrać. Następnie użytkownik musi kliknąć przycisk Download comments, który zapisze wszystkie komentarze.

3.1.3. Workspace

Po utworzeniu przynajmniej jednej przestrzeni roboczej pojawia się menu nawigacyjne zakładek, gdzie każda zakładka odpowiada osobnej przestrzeni roboczej. Można swobodnie przełączać się między tymi zakładkami i zamykać je. Zawartość każdej zakładki obejmuje:

- Pole na wykresy.
- Przyciski Previous i Next do przełączania wykresów w polu na wykresy.
- Przycisk Text do otwierania okna dialogowego zarządzania przetwarzaniem tekstu.
- Przycisk Vectorization do otwierania okna dialogowego zarządzania wektoryzacją.
- Przycisk Cluster do otwierania okna dialogowego zarządzania klasteryzacją.
- Przycisk Apply do zastosowania wszystkich wybranych parametrów i otwarcia okna z wynikami klasteryzacji.

Pole na wykresy

Pole na wykresy w każdej chwili może wyświetlać jeden z trzech proponowanych wykresów:

- Wykres częstości słów (freq).
- Wykres długości tekstu (length).
- Wykres najbliższych sąsiadów (k-nearest).

Za pomocą przycisków Previous i Next można przełączać wykresy odpowiednio do przodu i do tyłu.

3.1.4. Text

Przycisk Text otwiera okno dialogowe, w którym użytkownik może określić parametry przetwarzania tekstu. Oto jak domyślne okno wygląda **TextProcessingWindowStandard**. Domyślnie wszystkie parametry są ustawione na wartość true, co oznacza, że są używane.

Użytkownik może wybrać jedną z proponowanych opcji przetwarzania tekstu:

- Alpha – podstawowa obróbka opisana wcześniej
- Stop-words – usuwanie stop słów opisane wcześniej
- Lematize – lematyzacja opisana wcześniej

3.1.5. Wektoryzacja

Przycisk Vectorization otwiera okno dialogowe, w którym użytkownik może wybrać jeden z proponowanych metod wektoryzacji i określić parametry dla wybranej metody. Oto jak domyślne okno wygląda **VectorizationWindowStandard**. Domyślnie wybrana jest metoda TF-IDF z domyślnymi parametrami. Następnie wymieniono listę dostępnych metod i parametrów dla każdej metody:

- Metoda TF-IDF. Dostępne parametry do konfiguracji:
  - max_df: Zakres dozwolonych wartości ()
  - min_df: Zakres dozwolonych wartości ()
- Metody Word2vec. Dostępne parametry do konfiguracji:
  - vector_size: Zakres dozwolonych wartości ()
  - window: Zakres dozwolonych wartości ()
  - min_count: Zakres dozwolonych wartości ()
  - sg: Zakres dozwolonych wartości ()

Rysunek 3.1: Przyklad rys

3.1.6. Cluster

Przycisk Cluster otwiera okno dialogowe, w którym użytkownik może wybrać jedną z proponowanych metod klasteryzacji i określić parametry dla wybranej metody. Oto jak domyślne okno wygląda **ClusterWindowStandard**. Domyślnie wybrana jest metoda K-means z domyślnymi parametrami. Następnie wymieniono listę dostępnych metod i parametrów dla każdej metody:

- Metoda K-means. Dostępne parametry do konfiguracji:
  - n_cluster: Zakres dozwolonych wartości ()
  - max_iter: Zakres dozwolonych wartości ()
  - n_iter: Zakres dozwolonych wartości ()
- Metody DBSCAN. Dostępne parametry do konfiguracji:
  - min_samples: Zakres dozwolonych wartości ()
  - eps: Zakres dozwolonych wartości ()

3.1.7. Result Window

Przycisk Apply otwiera nowe okno z wynikami wykonanej pracy. Okno to zawiera pole z wykresem (2D plot) i pasek narzędziowy (toolbar), gdzie użytkownik może skorzystać z gotowych funkcji zaproponowanych przez bibliotekę Matplotlib do nawigacji po wykresie i innych operacji.

3.2. Przykład użycia

how use image

3.2.1. Pobieranie danych tekstowych

3.2.2. Tworzenie Workspace

3.2.3. Wybór metod i parametrów

3.2.4. Przetwarzanie tekstu

Alpha

Stop-words

Lemmatize

3.2.5. Wektoryzacja

TF-IDF

Word2Vec

3.2.6. Klasteryzacja

K-means

DBSCAN

3.2.7. Analiza wynkiow pracy

Rozdział 4

Tytuł 4

Tekst.

4.1. Sekcja

Na Rysunku 4.1.

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

Podsumowanie

Podsumujac analiza danych tekstowych jest...

Bibliografia

[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/*, (20.10.2023).

Spis listingów

Spis tabel

Spis rysunków