

SEMANTYCZNA WYSZUKIWARKA MediSemch

Sylwia Z., Jakub B., Kacper L.

Kraków, 2023

1. Abstrakt

Każdemu znane są wyszukiwarki internetowe będące obecnie podstawą zdobywania informacji w internecie. Ich zadaniem jest znalezienie najlepszych (oczekiwanych) wyników na podstawie zazwyczaj kilku słów – sformułowanego zapytania. Aby spełnić te wymagania, od lat 90-tych powstawały wyszukiwarki oparte na znajdowaniu i dopasowywaniu słów kluczowych, składników w danych [1]. Opisane rozwiązanie nie dawało jednak tak dobrych rezultatów jak wyszukiwarki semantyczne, które poza surowymi słowami biorą pod uwagę również „znaczenie” danego zapytania. Pozwala to na szukanie w znacznie szerszym obszarze bez ograniczeń do konkretnych zbiorów liter – brane pod uwagę są również inne dane z podobnym znaczeniem.

Aktualna praca omawia stworzenie wyszukiwarki opartej na semantyce. Podczas tego procesu znajdowane są najbardziej odpowiednie parametry takiego silnika w zależności do zadania jakie ma spełnić. Omówiono również sposoby pozyskania bazy danych (oraz jej odpowiednie przetworzenie), z której wyszukiwane będą wyniki zapytań. Wszystko zostało opracowane w interfejs graficzny, aby umożliwić proste korzystanie z aplikacji.

2. Wstęp

2.1 Cel

Zadaniem w projekcie będzie stworzenie wyszukiwarki semantycznej - *MedSemch*, która opiera swoją bazę danych na Wikipedii - darmowej encyklopedii [2] w języku angielskim. Tematem wspólnym pozyskiwanych artykułów będzie medycyna. Cel pracy to poprawne wyszukiwanie tytułów treści z Wikipedii na podstawie zapytania znajdującego się tematycznie w zakresie zebranej bazy danych. Wyszukiwarka ma choć w części być tak efektywna jak inne popularne tego typu narzędzia jak np.: *Google*, *DuckDuckGo* czy *Bing* – testy *MedSemch* opierać się będą na porównywaniu wyników z wymienionymi gigantami.

2.2 Zakres

Podstawą stworzenia wyszukiwarki będzie określenie sposobu pozyskania i przechowywania danych, na których będzie się ona opierać. Treści stron internetowych reprezentowane są za pomocą plików html lub xml. Dane pobrane z Wikipedii muszą zatem być odpowiednio przetworzone (wyodrębnione) aby uzyskać samą treść artykułu, na podstawie której oceniane będzie dopasowanie do zapytania. Przechowywanie zbioru danych może odbywać się zarówno lokalnie jak i w chmurze, stosując silnik bazodanowy lub zwykłe pliki tekstowe. Jednym z rozwiązań jest także wyszukiwanie artykułów z danej dziedziny w czasie rzeczywistym – jednak od początku zrezygnowano z tej opcji, przez wzgląd na wydajność wyszukiwarki.

Głównym celem jest jednak stworzenie odpowiedniego procesu, który będzie w stanie wyodrębnić znaczenie semantyczne dokumentów aby odszukać ten najbardziej oczekiwany na podstawie zapytania

użytkownika. Aby to uzyskać zbadano oraz przetestowano określone sposoby pozyskiwania znaczenia tekstu.

Aby ułatwić korzystanie i testowanie różnych opcji wyszukiwarki (rodzajów znajdowania podobieństw tekstów), narzędzie powinno zawierać prosty w obsłudze interfejs użytkownika.

2.3 Metodyka

Do napisania silnika *MediSemch* użyto języka *Python* w wersji 3.11. Dane do bazy danych zostały pozyskane za pomocą narzędzia *Wikipedii – Export pages* [3]. Pozwala ono na pobranie pliku xml zbioru artykułów z podanej kategorii. Oczyszczeniem artykułów z niepotrzebnych ciągów miały zająć się wyrażenia regularne regexp z modułu *re Pythona*. Baza danych *MongoDB* została użyta do przechowywania oraz późniejszego wykorzystania artykułów (oraz biblioteka *pymongo* do utworzenia klienta aplikacji).

Wyszukiwanie za pomocą semantyki opiera się na określonych aspektach przetwarzania tekstu. Od odpowiedniego podzielenia go na słowa/tokeny (tokenizacja). Przez utworzenie z nich statystyki słów (wektoryzacja – wykorzystywana w wyszukiwarkach opartych na słowach kluczowych). Po wyodrębnienie znaczenia zbioru słów (semantyzacja) – utworzenie tzw. topicków. Gdy z danych artykułów oraz zapytania użytkownika zostaną wyodrębnione odpowiednie cechy, można przejść do ich porównywania. Opiera się ono na obliczaniu podobieństwa/odległości wektorów semantycznych. Dzięki temu zostanie znaleziony najbliższy znaczeniowo odpowiadający tekst do danego zapytania.

Do procesu wyszukiwania semantycznego zastosowano takie narzędzia/biblioteki jak: *nlk* oraz *transformers* (tokenizacja), *sklearn* (wektoryzacja, semantyzacja oraz obliczanie podobieństwa wektorów). Użyto także innych dodatkowych bibliotek takich jak: *pandas* (przetwarzanie dokumentów), *numpy* (dodatkowe obliczenia), *matplotlib* (wykresy wyników analiz i testów).

W utworzeniu graficznego interfejsu użytkownika pomocna okazała się biblioteka *dearpygui*.

3. Wyszukiwarka semantyczna

3.1 Tokenizacja

Pierwszym krokiem analizy treści jest jej podział na mniejsze, łatwiejsze w interpretacji części. Zazwyczaj są to słowa (oddzielone znakiem spacją) lub zbiory słów. Takie jednostki nazywane są tokenami.

Omówione zostaną sposoby tokenizacji użyte w projekcie:

- *treebank* – wykorzystuje wyrażenia regularne regexp, posiada informacje o relacjach składniowych w zdaniu, znaki interpunkcyjne są oddzielane do osobnych tokenów, potrafi poprawnie oddzielać takie frazy jak „don’t” -> „do” „n’t”, oraz „they’ll” -> „they” „’ll”. [6]
- *casual tokenizer* – charakteryzuje się wykorzystaniem do tekstów z życia codziennego, np. dla treści z social mediów; potrafi poradzić sobie między innymi z emotikonami.

- biobert – został pre-trenowany na tekstach medycznych i biomedycznych przez co jest dostosowany do struktury tych dokumentów, stosuje tokenizację słów ale także i segmentów. [7]
- punkt - nie jest oparty na regułach lub wzorcach, został pre-trenowany uczeniem nienadzorowanym. [8]

3.2 Wektoryzacja

Po tokenizacji słów, należy nadać im znaczenie oraz reprezentację numeryczną. Taki proces nazywamy wektoryzacją. Polega on zazwyczaj na zliczeniu częstotliwości wystąpienia danego wyrazu w tekście. Najpierw utworzony zostaje korpus wszystkich słów występujących w bazie dokumentów. Następnie obliczamy dla każdego dokumentu ilość konkretnych tokenów. Wstawiamy te wartości na odpowiednie miejsca, które reprezentują dane słowo w wektorze.

Zastosowane metody wektoryzacji:

- bag of words – korzysta z podstawowej definicji wektoryzacji omówionej wyżej,
- tf-idf – jest bardziej użyteczną techniką wektoryzacji – mierzy stosunek liczby wystąpień w dokumencie do liczby wszystkich słów w dokumencie, dodatkowo uwzględniając przy tym całą resztę dokumentów zawierających dany termin (obniża wagę terminów powszechnie występujących we wszystkich dokumentach oraz pomaga wykryć tokeny które wyróżniają jedne dokumenty od innych).

3.3 Semantyzacja

Główną częścią wyróżniającą wyszukiwarkę semantyczną od zwykłej bazującej jedynie na wektoryzacji (występowaniu słów) – jest nadanie im znaczenia. Właśnie na tym polega semantyzacja. Na zamianę wektorów reprezentujących jedynie wartość występowania słów, na wektory (również reprezentowane wartościami liczbowymi) wnoszące „cechy” oraz ich znaczenie dla danego dokumentu [4]. Dodatkowo nie ma konieczności trzymania się sztywnych wymiarów wektorów (które w przypadku wektoryzacji oznaczały ilość różnorodnych tokenów w zbiorze dokumentów – wielkość słownika). Sposoby semantyzacji pozwalają na określenie/zredukowanie ilości podziałów „tematycznych” na jakkolwiek wymiar (nie raz zmniejszając tym jednak ilość niesionych przez nie informacji). Tak utworzone wektory nazywane są zazwyczaj topickami.

Przetestowane metody semantyzacji:

- PCA - technika statystyczna służąca do redukcji wymiarowości danych (przekształcenie współrzędnych na takie, których oś wyznaczają wektory własne macierzy kowariancji). [11]
- LDIA - tworzy semantyczny model przestrzeni wektorowej zakładając, że każdy dokument jest mieszanką różnych tematów, a każdy temat jest mieszanką różnych słów; najbardziej „ludzki” sposób „cechowania” dokumentów – wykorzystywany gdy ważne jest zrozumienie tematów korpusu; zazwyczaj trenowanie trwa dłużej porównując do innych metod [4] [9].
- Truncated SVD – ocenia jakie relacje mają sąsiadujące ze sobą słowa, oraz relacje pomiędzy dokumentami; wycina niepotrzebne informacje, zostawiając „esencję” znaczenia/semantyki dokumentu; ekstrakcja struktur w danych polega na dekompozycji na trzy macierze wektorów

osobliwych (lewa macierz U zawiera informacje o sąsiadach konkretnego słowa – korelacja pomiędzy współwystępowaniem ze sobą konkretnych zbiorów słów a tematami; diagonalna macierz S zawiera pojedyncze wartości topicków; a prawa macierz V zawiera informacje o wspólnych tematach pomiędzy dokumentami). [10]

3.4 Metryki dystansu/podobieństwa

Aby uzyskać odpowiedni artykuł, dopasowany do wprowadzonego zapytania potrzebna jest odpowiednia metoda porównująca wektory semantyczne. W tym celu wystarczy zastosować metrykę podobieństwa lub dystansu wektorów (dokumentu z zapytaniem).

Porównane metryki:

- Euclidean distance (euclidean) – odległość geometryczna między dwoma punktami w przestrzeni euklidesowej,
- Cosine similarity (cosine) – kąt między dwoma wektorami, co odzwierciedla kierunek i stopień zgodności między nimi,
- Chebyshev distance (chebyshev) – największa różnica między współrzędnymi dwóch punktów w przestrzeni,
- Manhattan distance (manhattan) – suma różnic bezwzględnych między współrzędnymi dwóch wektorów.

4. Działanie wyszukiwarki

4.1 Przygotowanie danych

Dane zostały zebrane za pomocą narzędzia Wikipedia – Export pages [2] – tematykę ograniczono do *choroby astmy*. Po ich oczyszczeniu pozostało 45 artykułów.

Dane surowych plików xml zapisane zostały w bazie danych MongoDB. Aby wyodrębnić potrzebny tekst artykułów pliki zostały oczyszczone za pomocą wyrażeń regularnych. Polegało to na usunięciu zbędnych elementów takich jak:

- zawartość tagów referencyjnych - `<ref> ... </ref>`,
- tagi HTML,
- niepotrzebne informacje zawarte w podwójnych nawiasach klamrowych - `{{ ... }}`,
- nagłówki znajdujące się na końcu artykułów (zdecydowano się zrezygnować z pochodzących od nich informacji – jako mniej istotnych w ocenie semantyki całości) takie jak: *References, See also, External links*,
- sekcje kategorii artykułu,
- niepotrzebne ciągi znaków związane z załączeniem grafiki, zachowując tytuł i opis zdjęcia,
- linkowania fraz do innych artykułów,
- znaki okalające tytuły, frazy (znaki równa się, nawiasy, apostrofy),

- inne zbędne wyrażenia takie jak '___TOC___'.

Oczyszczony tekst zapisany został w osobnej kolumnie tabeli w bazie danych.

articles
_id: ObjectId
title: String
content: String
clean_content: String

Rys. 1 Struktura tabeli artykułów.

4.2 Działanie silnika

Główna część aplikacji polega na wykonaniu algorytmów opisanych w rozdziale 3 *Wyszukiwarka semantyczna*. Baza danych artykułów zostaje wykorzystana do wytrenowania słownika słów (tokenizera) oraz semantyzatora. Następnie tworzone są wektory wystąpień słów, które wykorzystywane są do wyznaczenia wektorów semantycznych o wybranej długości (ten etap jest wykonywany na każdym dokumencie oraz zapytaniu). W większości przypadków ilość tematów jest mniejsza niż długość wektorów słów. Na koniec wystarczy porównać każdą tablicę topicków dokumentów z wektorem semantycznym wprowadzonego przez użytkownika tekstu – za pomocą jednej z metod podobieństwa/dystansu. Im większa wartość metryki podobieństwa (lub im mniejsza wartość metryki dystansu) tym bardziej dany dokument pasuje do zapytania.

4.3 Testy metod

Testy polegały na porównaniu wyników stworzonej wyszukiwarki semantycznej z testowanymi parametrami do wyników wyszukiwarki Google.

- 1) Wpisywane były szukane frazy (queries) do Google'a oraz zapisywano 10 pierwszych wyników. Następnie tworzono wektor poprawnych tytułów artykułów (`true_labels`) z całej bazy danych aplikacji do podanego zapytania – jeśli artykuł o podanym tytule pojawił się w wynikach Googla – przypisz 1, jeśli nie – przypisz 0.
- 2) Następnie brano 10 najbardziej pasujących artykułów według stworzonej wyszukiwarki do testowanego zapytania; Tworzono wektor adnotacji (`predicted_labels`) według testowanej wyszukiwarki (pierwsze 10 artykułów dostało wartość 1, pozostałe - 0).
- 3) Obliczano metrykę precyzji ($TP / (TP + FP)$) pomiędzy adnotacjami poprzez Google (`true_labels`) oraz testowaną wyszukiwarką (`predicted_labels`).

Z pierwszych testów zostawiono następujące parametry które, zazwyczaj uzyskiwały wyniki:

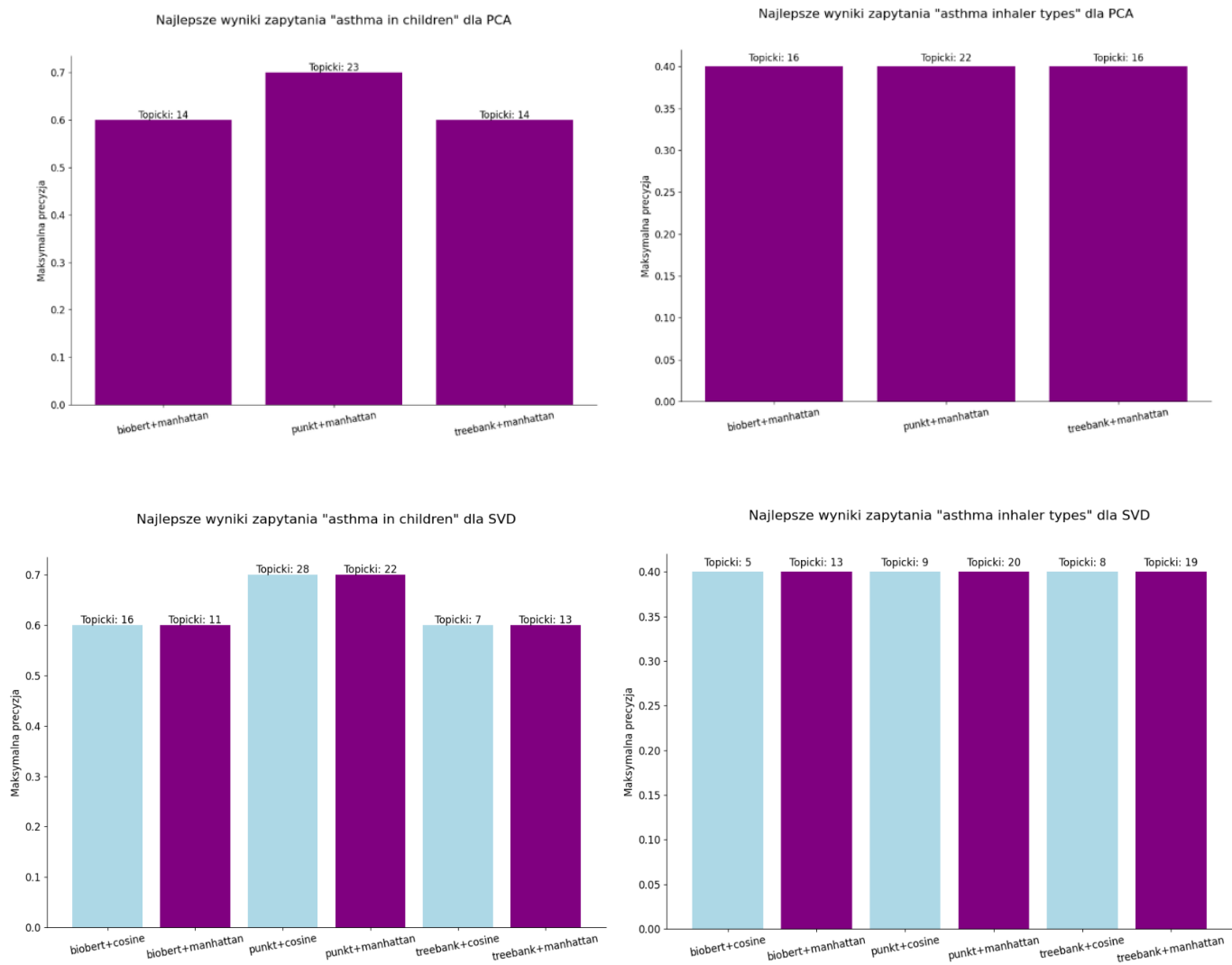
$$precision \geq 0.3$$

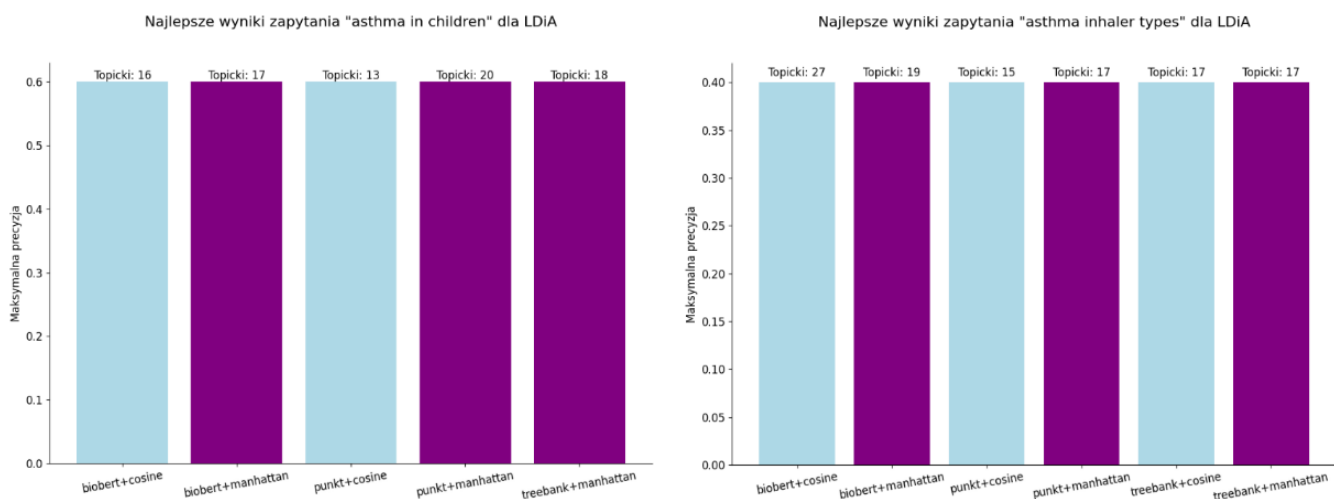
Parametry	Pozostające	Odrzucone
Tokenizacja	'treebank', 'biobert', 'punkt'	'casual'
Wektoryzacja	'tfidf'	'bow'
Semantyzacja	'pca', 'svd', 'ldia' – im więcej topicków tym lepsze wyniki	-
Metryka dystansu/podobieństwa	'cosine', 'manhattan'	'chebyshev', 'euclidean'

Tab. 1 Przedstawienie odrzuconych i zostawionych do dalszych testów metod.

W kolejnych testach poszerzono zakres testów dotyczących ilości topicków.

Wykresy prezentujące najlepsze wyniki poszczególnych połączeń metod dla dwóch testowanych zapytań „asthma in children” oraz „asthma inhaler types”:





Rys. 2 Przedstawienie precyzji poszczególnych połączeń metod zastosowanych w wyszukiwarce (metoda podobieństwa/dystansu: kolor błękitny – cosine; kolor purpurowy - manhattan).

Na podstawie wyników wybrane zostały podane metody jako domyślne w aplikacji:

Tokenizacja: **punkt**

Wektoryzacja: **tf-idf**

Semantyzacja: **SVD**

Metryka odległości/podobieństwa: **Manhattan/cosine** (ustawienia domyślne - metryka manhattan)

Liczba tematów semantycznych: **22**

4.4 Omówienie wyników

Metoda tokenizacji *punkt* jest specyficzna przez wzgląd na wykorzystanie metody uczenia nienadzorowanego do trenowania zamiast określonych zasad zdefiniowanych z góry czy wyrażen regularnych – co prawdopodobnie wpłynęło na jego przewagę nad innymi metodami. Jest on dodatkowo pre-trenowany. *Biobert* pozyskiwał dobre wyniki w zapytaniach, które pasowały do bardziej specjalistycznych/medycznych tekstów.

Tf-idf znacząco przewyższył 'worek słów' we wszystkich swoich wynikach. Opiera się on na bardziej skomplikowanych zasadach obliczania wektorów słów, niż samo zliczanie częstotliwości – co pokazało swoją przewagę.

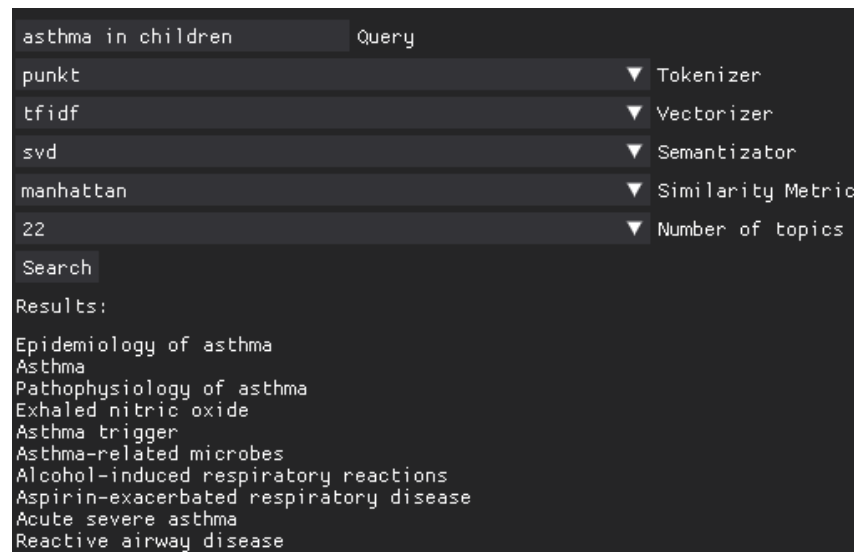
Metodzie *LDiA* nie udało się dorównać pozostałym pod względem precyzji w pierwszym pytaniu. *PCA* osiągało dobre wyniki jedynie przy wykorzystaniu metryki dystansu *Manhattan*. Metoda dorównywała jednak *SVD* jeśli chodzi o precyzję wyników. *SVD* połączony z metryką *cosine* potrzebuje zazwyczaj mniejszej liczby tematów do uzyskania odpowiednio wysokich wyników – możliwość powiązania może wynikać z tego, że metryka ta lepiej radzi sobie z danymi o mniejszej liczbie wymiarów - jednak w pierwszym pytaniu potrzebował aż 28 topicków w połączeniu z tokenizacją *punkt*.

Pomimo potrzeby użycia większej ilości wymiarów (mniejszej „kompresji” informacji) metoda *SVD* z tokenizacją *Manhattan* wydaje się dawać najbardziej precyzyjną niezależnie od rodzaju zapytania.

Metody PCA i SVD potrzebowały sporej ilości topicków aby osiągnąć lepsze wyniki – powodem może być zbyt jednolita (znajdująca się w wąskim zakresie jednego tematu) baza danych – przez co konieczne były dodatkowe wymiary do odpowiedniego podziału dokumentów. Z tego samego powodu LDiA mogło uzyskać gorsze wyniki – ten sam kontekst dokumentów mógł wpłynąć na gorsze porządzenie sobie z działającym jak „ludzka separacja” algorytmem.

4.5 Interfejs użytkownika

Wyświetlanie interfejsu jest zrealizowane przy pomocy biblioteki dearpygui. Przy użyciu funkcji `create_context()` oraz `create_viewport()` tworzone jest okienko aplikacji desktopowej. W polu tekstowym użytkownik wpisuje zapytanie. Następnie z rozwijalnych list może wybrać odpowiednie metody tokenizacji, wektorowania, semantyzacji oraz metryki podobieństw. Po kliknięciu przycisku Search wyświetlają się najbardziej odpowiednie wyniki.



Rys. 3 Przedstawienie interfejsu aplikacji.

5. Podsumowanie

Osiągnięto cel jakim było stworzenie wyszukiwarki semantycznej opartej na bazie danych z Wikipedii darmowej encyklopedii – z wystarczającą dokładnością – porównując wyniki do najbardziej popularnych narzędzi na rynku.

Mimo to wyniki powinny być zbadane w znacznie szerszym zakresie, uwzględniając takie aspekty jak:

- Zastosowanie obszerniejszej bazy danych (dodanie innych kategorii z domeny medycznej),
- W wynikach jakości poszczególnych metod należy wziąć pod uwagę inne metryki poza precyzją (obecnie oceniana jest tylko ilość pasujących artykułów – nie uwzględniając odpowiednio błędu jakim są niepasujące treści w najlepszych wynikach),

- Poszerzenie testów na większą ilość specyficznych zapytań.

Warto zaznaczyć, że macierze wykorzystywane do użycia metody semantyzacji PCA – muszą być gęste. Jednak po stworzeniu wektorów za pomocą tf-idf – zazwyczaj większość słów w słowniku stworzonym na podstawie dokumentów nie występuje osobno w każdym dokumencie. Z tego względu stworzona przez niego macierz jest rzadka (macierz zapisywana jest inaczej aby nie tracić miejsca na niepotrzebne wartości zer). Przez przekazaniem jej do PCA – należy dokonać niezbędnej konwersji.

Nie przetestowano metod stemmingu (proces usuwania sufiksów ze słów, aby uzyskać ich podstawową formę, czyli "rdzeń" - stem) oraz lematyzacji (proces przekształcania słów na ich lematy - czyli formy podstawowej, uwzględniając morfologiczną analizę słowa) ze względu na zazwyczaj gorsze wyniki w przypadku wykorzystania w wyszukiwarce semantycznej [5].

6. Bibliografia

- [1] Seymour, Dr. Tom & Frantsvog, Dean & Kumar, Satheesh. (2011). "History Of Search Engines". International Journal of Management & Information Systems (IJMIS). 15. 10.19030/ijmis.v15i4.5799.
- [2] Wikipedia The Free Encyclopedia
https://en.wikipedia.org/wiki/Main_Page
- [3] Wikipedia – Export pages
<https://en.wikipedia.org/wiki/Special%3aExport>
- [4] Hobson Lane, Cole Howard, Hannes Hapke, "Natural Language Processing in Action", Manning, 2019, chapter 4
- [5] Radosław Bujak Michał Borowski Michał Gątkowski "Wyszukiwarka semantyczna z bazą opartą na Wikipedii (wybrana tematyka)", 2022.
https://ii.pk.edu.pl/~rkycia/classes/2022/wiosna/files/NLP/PJN_Wyszukiwarka_Semantyczna_Raport.pdf
- [6] Robert McIntyre – NLTK Project, Penn Treebank Tokenizer, 2023.
<https://www.nltk.org/api/nltk.tokenize.treebank.html>
- [7] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang "BioBERT: a pre-trained biomedical language representation model for biomedical text mining", 2019.
<https://academic.oup.com/bioinformatics/article/36/4/1234/5566506>
- [8] NLTK Project, Punkt Sentence Tokenizer, 2023.
<https://www.nltk.org/api/nltk.tokenize.punkt.html>
- [9] LDiA - scikit-learn
<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html>
- [10] Truncated SVD - scikit-learn
<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html>

[11] PCA - scikit-learn

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>