Karol Łyskawiński 137335
Jakub Słota 137405

Inteligentne Wyszukiwanie Informacji – sprawozdanie z projektu

1. Temat projektu

Tematem realizowanego projektu jest **ekstrakcja fraz kluczowych z korpusów tekstowych**. Wydobywanie wyrazów bądź fraz o kluczowym znaczeniu dla danego korpusu tekstowego pozwala na zwięzłe opisanie jego zawartości. Frazy kluczowe znajdują wiele zastosowań w przetwarzaniu języka naturalnego, służą m. in. do kategoryzacji dokumentów, ich klasteryzacji oraz streszczania. Przyjęto założenie, że korpus tekstowy został napisany w języku angielskim.

2. Zastosowane algorytmy

2.1. Wyznaczanie fraz kluczowych

W celu realizacji projektu zaimplementowano algorytm **TextRank**. Jest to przykład algorytmu niewymagającego nadzoru, więc działającego bez zbioru uczącego. TextRank wyznacza ranking najlepszych fraz kluczowych poprzez zastosowanie algorytmu PageRank na odpowiednio przygotowanym grafie. Wierzchołkami grafu są pewne jednostki badanego korpusu tekstowego, a krawędzie stanowi miara podobieństwa pomiędzy tymi jednostkami. Za jednostki tekstowe uznaje się słowa, które można połączyć w wielowyrazowe frazy już po uzyskaniu rankingu. Natomiast miarę podobieństwa określa wspólne występowanie wyrazów w oknie mieszczącym N wyrazów.

W pierwszym kroku algorytmu tworzony jest zbiór kandydatów na wyrazy kluczowe. Wybierane są tokeny wyrazów wyszukane w tekście. Następnie odfiltrowywane są tokeny należące do zbioru stop-words w j. angielskim, będące znakami interpunkcyjnymi lub stanowiące inne części mowy, niż rzeczownik i przymiotnik.

W drugim kroku powstaje graf kandydatów na wyrazy kluczowe. Wierzchołki są tworzone ze zbioru unikatowych tokenów uzyskanych w pierwszym kroku. Krawędzie występujące między wierzchołkami są nieskierowane i nieważone. Krawędź jest tworzona, jeżeli w zbiorze kolejnych kandydatów wyrazy sąsiadowały bezpośrednio ze sobą, zatem okno sąsiedztwa mieści N=2 wyrazów.

Na powstałym w kroku drugim grafie uruchamiany jest algorytm PageRank celem utworzenia rankingu kandydatów na wyrazy kluczowe. Część najlepszych kandydatów w rankingu, których suma wag nie przekracza wartości 0.6 przechodzi dalej.

Na podstawie zbioru najlepszych kandydatów oraz zbioru wszystkich wyrazów w tekście tworzony jest zbiór ostatecznych fraz kluczowych. Występujące blisko siebie wyrazy kluczowe (do 4 wyrazów w przód) w zbiorze wszystkich wyrazów łączy się we

frazy kluczowe, których waga stanowi średnią arytmetyczną wag tworzących je wyrazów. Po tym kroku sumaryczna waga wszystkich fraz kluczowych może przekraczać wartość 1, jeśli nowa fraza składa się z dwóch innych fraz o długości jednego słowa. Aby nadal móc interpretować wagi jako wartości procentowe, należy je znormalizować.

Wszystkie wyrazy wchodzące w skład inicjalnego zbioru kandydatów na wyrazy kluczowe, jak i zbioru wszystkich wyrazów, są ujednolicane poprzez zamianę wielkich liter na małe oraz lematyzację. Dzięki temu powtórzenie wyrazu w liczbie mnogiej lub zapis wielką literą nie zaburza wyników uzyskiwanych przez algorytm.

Po wyznaczeniu wszystkich fraz kluczowych następuje selekcja najlepszych pozycji do wypisania na wyjściu programu. W tym kroku wybierane są takie najwyżej punktowane frazy, których suma wag nie przekracza określonego przez użytkownika progu.

2.2. Klasteryzacja fraz kluczowych

Kolejną z możliwości stworzonej aplikacji jest klasteryzacja fraz kluczowych. Operacja ta polega na zgrupowaniu fraz wokół powtarzających się w nich słów i wymaga wcześniejszego wyznaczenia fraz kluczowych.

Nie zawsze da się określić które słowo pełni najważniejszą rolę we frazie, żeby móc przypisać ją do takiego klastra, dlatego zdecydowano, że klastry będą miały bardziej rozmytą postać, czyli frazy będą mogły należeć jednocześnie do kilku klastrów. Kryterium przynależności frazy do klastra stanowi występowanie w niej danego wyrazu. W tej sytuacji frazę można rozumieć jako punkt w przestrzeni leżący w tej samej odległości od wszystkich punktów reprezentujących klastry, do których dana fraza należy.

Przykładowo dla poniższych fraz:

- python
- java
- language
- python scripting
- java application
- programming language
- java community
- java compiler
- java technology
- java platform
- scripting language
- popular language
- computer programming language
- implementation java compiler
- popular programming language
- application
- java virtual machine
- language processing task
- general-purpose computer programming language
- library

- use

Dokonano podziału na następujące klastry:

- java:
  - java
  - java application
  - java community
  - java compiler
  - java technology
  - java platform
  - implementation java compiler
  - java virtual machine
- language:
  - language
  - programming language
  - scripting language
  - popular language
  - computer programming language
  - popular programming language
  - language processing task
  - general-purpose computer programming language
- programming:
  - programming language
  - computer programming language
  - popular programming language
  - general-purpose computer programming language
- application:
  - java application
  - application
- python:
  - python
  - python scripting
- scripting:
  - python scripting
  - scripting language
- computer:
  - computer programming language
  - general-purpose computer programming language
- popular:
  - popular language
  - popular programming language
- compiler:
  - java compiler
  - implementation java compiler

2.3. Wyznaczanie podobieństwa dokumentów

Jedną z funkcji projektu jest wyznaczanie podobieństwa dokumentów. Dokumenty poboczne – pozostałe pliki w katalogu w przypadku wejścia w postaci wskazanego pliku lub linkowane artykuły na Wikipedii w przypadku wejścia w postaci tytułu artykułu Wikipedii – są porównywane do głównego dokumentu, określanego mianem „master". Aby wykorzystać gotową logikę wydobywania fraz kluczowych z dokumentów, wyznaczanie podobieństwa opiera się o frazy kluczowe. Każdy porównywany dokument zostaje poddany ekstrakcji fraz kluczowych. Frazy kluczowe uzyskane dla dokumentów są traktowane jako ich sygnatura, dobrze odzwierciedlająca zawartość tekstową.

Znaną metodą porównywania dokumentów jest obliczenie miary ich podobieństwa jako cosinusa kąta między dokumentami w przestrzeni wektorowej wyrazów. W celu wyjaśnienia algorytmu, należy wprowadzić poniższe pojęcia:

- Przestrzeń wektorowa wyrazów – *n*-wymiarowa przestrzeń, gdzie *n* jest liczbą różnych wyrazów występujących w zbiorze dokumentów.
- Wektor – dokument *i* jest reprezentowany przez wektor $d_i = (x_{i1}, x_{i2}, \dots, x_{1n})$, którego *j*-ty element wynosi w_{ij}, gdzie $w_{ij} > 0$, jeżeli wyraz *j* występuje w *i*-tym dokumencie oraz 0 w przeciwnym wypadku.
- w_{ij} – waga wyrazu *j* w dokumencie *i*, zakładamy, że $w_{ij} = 1$.
- Długość wektora – wektor $d = (x_1, x_2, \dots, x_n)$ ma długość $|d| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.
- Iloczyn skalarny wektorów – $d_1 * d_2 = x_{11} * x_{21} + x_{12} * x_{22} + \dots + x_{1n} * x_{2n}$
- Cosinus kąta między wektorami – $\cos(\theta) = \frac{d_1 * d_2}{|d_1| * |d_2|}$, wartość 1 oznacza maksymalne podobieństwo, a 0 brak podobieństwa.

Aby sprowadzić dokumenty do przestrzeni wektorowej, ich frazy kluczowe są zamieniane na zbiory unikatowych wyrazów. Wówczas przestrzeń wektorową tworzą wszystkie unikatowe wyrazy należące do zbiorów wydobytych z porównywanych dokumentów. W dalszej analizie porównywane są dokumenty – d_1 (master) na temat języka Python i Java oraz d_2 na temat języka Python. W tabeli 1 przedstawiono uzyskane frazy kluczowe obu dokumentów. Kontrastowym kolorem czcionki wskazano frazy kluczowe wspólne dla dokumentów.

Tab. 1. Frazy kluczowe wydobyte z dokumentów d_1 i d_2

Frazy kluczowe dokumentu d_1 (master)	Frazy kluczowe dokumentu d_2
python	python
java	language
language	python scripting
python scripting	programming language
java application	scripting language
programming language	popular language
java community	
java compiler	

W tabeli 2 zestawiono unikatowe wyrazy występujące we frazach kluczowych dokumentów d_1 i d_2. Kontrastowym kolorem czcionki wyróżniono wyrazy występujące we frazach kluczowych obu dokumentów.

Tab. 2. Unikatowe wyrazy wydobyte z fraz kluczowych dokumentów d_1 i d_2

Wyrazy dokumentu d_1	Wyrazy dokumentu d_2
python	python
language	language
scripting	scripting
programming	programming
application	popular
java	
community	
compiler	

Tabela 3 przedstawia wektory reprezentujące dokumenty d_1 i d_2 w przestrzeni wektorowej wyrazów. Kontrastowym kolorem czcionki wyróżniono kolumny odpowiadające wyrazom występującym we frazach kluczowych obu dokumentów.

Tab. 3. Wektory reprezentujące dokumenty d_1 i d_2

Wektor	python	java	language	scripting	programming	popular	application	community	compiler
d_1	1	1	1	1	1	0	1	1	1
d_2	1	0	1	1	1	1	0	0	0

W tej chwili są gotowe dane potrzebne do obliczeń:

$$d_1 * d_2 = 1*1 + 1*0 + 1*1 + 1*1 + 1*1 + 0*1 + 1*0 + 1*0 + 1*0 = 4$$

$$|d_1| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2 + 1^2} = \sqrt{8}$$

$$|d_2| = \sqrt{1^2 + 0^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 0^2 + 0^2} = \sqrt{5}$$

$$\boldsymbol{\cos(\theta)} = \frac{d_1 * d_2}{|d_1| * |d_2|} = \frac{4}{\sqrt{40}} = \frac{4}{2\sqrt{10}} = \frac{2}{\sqrt{10}} = \boldsymbol{0,63}$$

Otrzymano wynik wskazujący, że treść dokumentów d_1 i d_2 jest podobna w 63%. Jest to wysokie podobieństwo, przekraczające domyślny próg 45% wskazujący, czy porównywane dokument należy uznać za podobny do mastera.

Ponieważ algorytm wykorzystuje wagi $w_{ij} = 1$ można w łatwiejszy sposób obliczać cosinusowe podobieństwo dokumentów, niż z definicji. Można zauważyć, że iloczyn $d_1 * d_2$ ma wartość równą liczbie wspólnych wyrazów w porównywanych dokumentach, zaznaczonych kontrastującym kolorem czcionki w tabelach. Z kolei długości wektorów wynoszą pierwiastek z liczby unikatowych wyrazów uzyskanych z fraz kluczowych. Zatem można sprowadzić obliczenia do następujących operacji na zbiorach:

s_1 – zbiór unikatowych wyrazów dokumentu d_1

s_2 – zbiór unikatowych wyrazów dokumentu d_2

$$\boldsymbol{\cos(\theta)} = \frac{size(intersection(s_1\ ,\ s_2))}{\sqrt{size(s_1) * size(s_2)}} = \frac{4}{\sqrt{8} * \sqrt{5}} = \boldsymbol{0,63}$$

3. Środowisko projektu

Projekt został zrealizowany w języku programowania Python. Podjęto taką decyzję ze względu na ekspresyjność samego języka, jak i spektrum dostępnych bibliotek pomocnych w implementacji algorytmu TextRank. Oprócz standardowego API Pythona wykorzystano następujące biblioteki:

- networkx: biblioteka służąca do tworzenia oraz przeprowadzania operacji na sieciach i grafach, zawiera implementację algorytmu PageRank;
- wikipedia: biblioteka ułatwiająca korzystanie z API Wikipedii celem wyszukiwania artykułów oraz pobierania ich treści;
- nltk: biblioteka dostarczająca zestaw narzędzi związanych z przetwarzaniem języka naturalnego, m. in. do lematyzacji, tokenizacji, tagowania części mowy.

Przedstawiony w dalszej części punktu 3. proces instalacji jest zapisany w skróconej wersji w pliku *installation.txt* w głównym katalogu projektu. W celu przygotowania środowiska potrzebny jest interpreter Pythona dostępny na stronie Python Software Foundation. 64-bitową wersję 2.7.10 wykorzystaną w projekcie można znaleźć pod adresem: https://www.python.org/downloads/release/python-2710/. Należy upewnić się, że zostanie zainstalowane konsolowe narzędzie pip służące do wygodnej instalacji bibliotek Pythona, używane w dalszych krokach.

Teraz jest możliwa instalacja potrzebnych bibliotek. Każdą z nich można zainstalować w domyślnej wersji poleceniem: `pip install <nazwa_biblioteki>` lub w wersji pobranej na dysk twardy w formie archiwum poleceniem: `pip install <ścieżka_do_archiwum_z_biblioteką>`. Poniżej wymieniono wersje poszczególnych bibliotek używane w projekcie wraz z odnośnikami do ich stron w Python Package Index:

- networkx 1.10, https://pypi.python.org/pypi/networkx/;
- wikipedia 1.4.0, https://pypi.python.org/pypi/wikipedia/;
- nltk 3.1, https://pypi.python.org/pypi/nltk/.

Ostatnim krokiem potrzebnym do przygotowania środowiska programistycznego jest pobranie pakietów nltk, wykorzystywanych przez bibliotekę. Należy uruchomić interpreter Pythona, a następnie zaimportować bibliotekę do powłoki interpretera poleceniem: `import nltk` i uruchomić menadżer pakietów nltk: `nltk.download()`. Należy pozostawić wybrane domyślne opcje oraz pozwolić na instalację pakietów.

Projekt został napisany w darmowym zintegrowanym środowisku deweloperskim PyCharm w wersji Community Edition 5.0.1. To IDE dedykowane do pisania kodu w języku Python jest dostępne do pobrania pod adresem: https://www.jetbrains.com/pycharm/download/. Autorzy projektu polecają PyCharm.

4. Interfejs konsolowy

CLI (Command Line Interface) jest podstawowym sposobem korzystania z projektu ekstrakcji fraz kluczowych. Stanowi uproszczony wariant interfejsu graficznego. W początkowej fazie projektu był jedynym sposobem na korzystanie z programu,

a obecnie sprawdza się najlepiej w celach diagnostycznych, pomiarowych oraz wtedy, gdy użytkownik ma do dyspozycji tylko wiersz poleceń.

Interfejs konsolowy projektu pozwala na przeprowadzenie ekstrakcji fraz kluczowych z wskazanego w parametrach wejściowych źródła oraz opcjonalnie wyznaczenie podobieństwa z dokumentami pobocznymi. Klasteryzacja fraz kluczowych pozyskanych ze źródła, jako prosta i pożyteczna dla czytelności wyników operacja, jest wykonywana przy każdym uruchomieniu. CLI różni się od GUI przede wszystkim tym, że jedno uruchomienie CLI przeprowadzi jedną ekstrakcję fraz kluczowych, zdefiniowaną przez parametry wejściowe, po czym zakończy działanie.

Do uruchomienia interfejsu konsolowego służy skrypt *AKE.py*, zawierający logikę związaną z ekstrakcją fraz kluczowych i CLI, zlokalizowany w głównym katalogu projektu. Dostępne są następujące parametry uruchomienia:

- typ źródła danych – parametr obowiązkowy, wskazuje typ danych wejściowych do przeprowadzenia ekstrakcji fraz kluczowych, może to być:
  - `wiki`: źródłem będą artykuły na Wikipedii;
  - `file`: źródłem będzie pojedynczy plik tekstowy;
  - `dir`: źródłem będzie katalog zawierający wyłącznie pliki tekstowe;

- ścieżka do źródła – parametr obowiązkowy, wskazuje lokację źródła o typie podanym w parametrze źródła danych, jego oczekiwana wartość zależy od wyboru źródła:
  - dla źródła `wiki`: lista oddzielonych przecinkiem tytułów artykułów na Wikipedii, np. `„Linux, Java”`;
  - dla źródła `file`: ścieżka prowadząca do pliku tekstowego, np. `katalog/plik.txt`;
  - dla źródła `dir`: ścieżka prowadząca do katalogu zawierającego pliki tekstowe, np. `katalog`, pod uwagę zostaną wzięte pliki należące również do wszystkich katalogów zagnieżdżonych we wskazanym;

- wyznaczanie podobieństwa z dokumentem głównym – parametr opcjonalny `--master`, kompatybilny tylko z typami źródła `wiki` dla pojedynczego artykułu i `file`, decyduje o przeprowadzeniu porównania głównego źródła ze źródłami pobocznymi. W przypadku opcji `wiki` źródłami pobocznymi są linkowane do artykułu głównego artykuły na Wikipedii, a w przypadku opcji `file` wszystkie pozostałe pliki zlokalizowane w katalogu zawierającym bezpośrednio główny plik. Wybór opcji `wiki` może skutkować długimi obliczeniami, wynikającymi z typowej liczności (kilkaset) linkowanych artykułów.

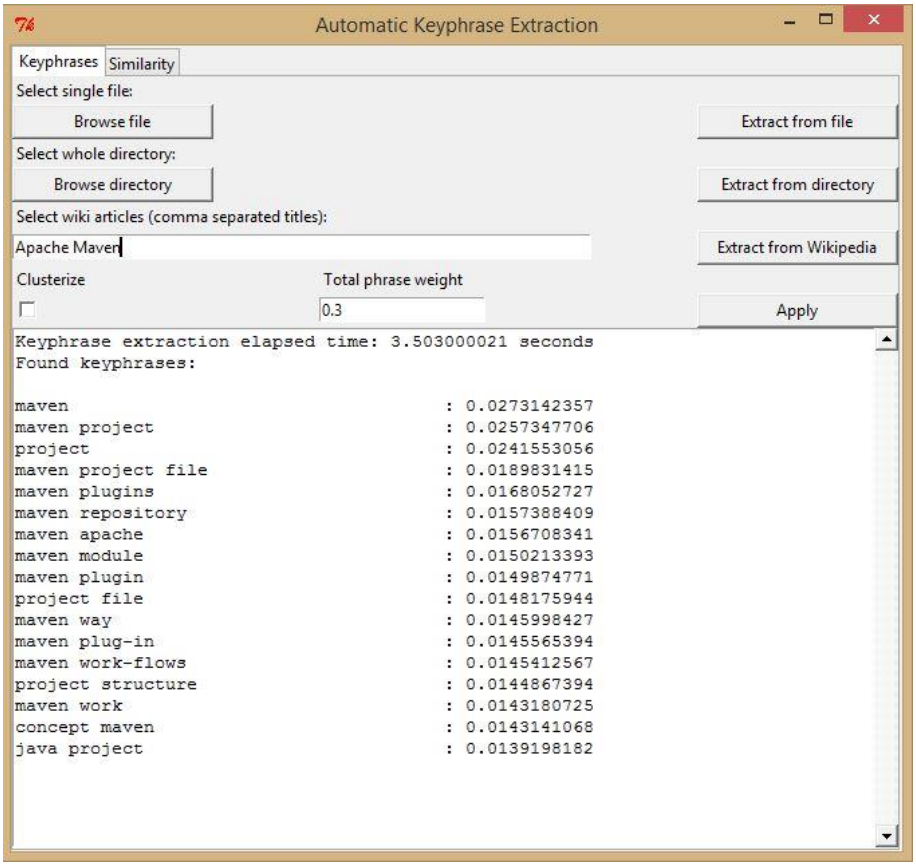
W komentarzu na początku skryptu *AKE.py* zostały wypisane możliwe sposoby użycia CLI. Wykorzystują umieszczony w katalogu głównym projektu katalog *res* z przykładowymi artykułami w plikach tekstowych. Poniżej opisano efekt uruchomienia CLI z różnymi wartościami parametrów wejściowych:

- `python AKE.py dir res` – przeprowadzenie ekstrakcji fraz kluczowych na wszystkich plikach zawartych w katalogu res;

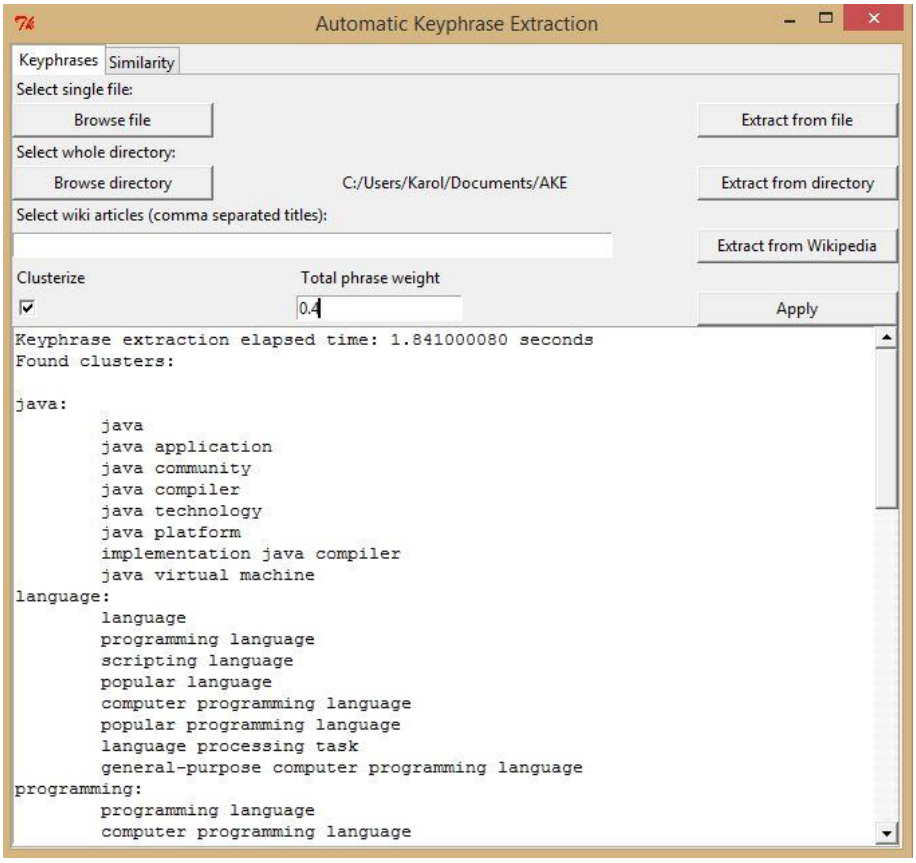
- `python AKE.py file res/pyton_usage.txt` – przeprowadzenie ekstrakcji fraz kluczowych na pliku pyton_usage.txt, zawartym w katalogu res;
- `python AKE.py wiki „Linux”` – przeprowadzenie ekstrakcji na artykule Wikipedii zatytułowanym „Linux”
- `python AKE.py file res/java_usage.txt --master` – przeprowadzenie ekstrakcji fraz kluczowych na pliku java_usage.txt, zamieszczonym w katalogu res wraz z wyznaczeniem podobieństwa do pozostałych plików w katalogu res;
- `pyton AKE.py wiki „Apache Maven”` – przeprowadzenie ekstrakcji fraz kluczowych na artykule Wikipedii zatytułowanym „Apache Maven” wraz z wyznaczeniem podobieństwa do linkowanych artykułów Wikipedii.

5. Graficzny interfejs użytkownika

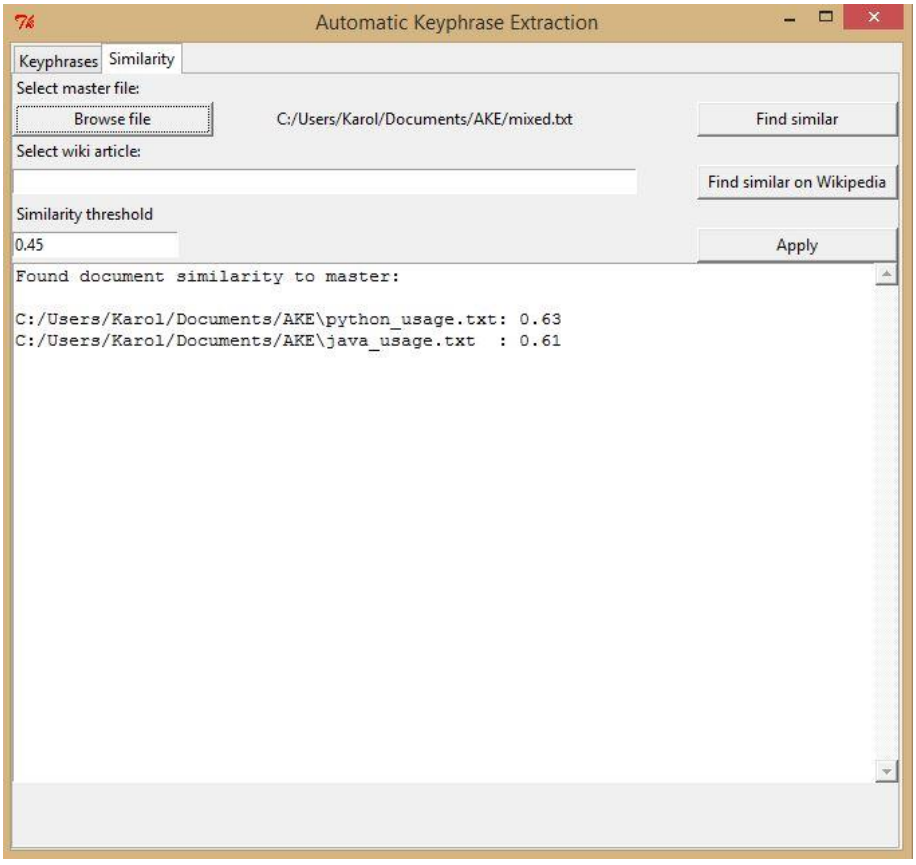
Oprócz interfejsu konsolowego, program wyposażony jest też w okienkowy interfejs graficzny. Ułatwia on korzystanie z programu, a jego kod demonstruje użycie modułu przetwarzania tekstu. Do uruchomienia interfejsu graficznego służy skrypt *AKE-gui.py*, który można uruchomić podwójnym kliknięciem lub poleceniem `python AKE-gui.py`. Poniżej przedstawiono zrzuty ekranu z aplikacji.



Rys. 1. Ekstrakcja fraz kluczowych z artykułu na wikipedii



Rys. 2. Klasteryzacja fraz kluczowych wydobytych z artykułów wewnątrz katalogu



Rys. 3. Wyszukiwanie artykułów podobnych do wskazanego pliku

Interfejs został napisany w bibliotece Tkinter dołączonej do dystrybucji Pythona. Składa się on z klasy Application wewnątrz pliku AKE-gui.py. Tworzenie kontrolek odbywa się w metodzie __init__, której pierwsza część dotyczy kontrolek związanych z kartą ekstrakcji fraz kluczowych, a druga tych związnych z wyszukiwaniem podobnych artykułów.

Rezultaty długo trwających operacji ekstrakcji fraz oraz wyszukiwania podobnych artykułów są zapisywane w polach klasy:

- `self.keyphrases` – lista par odnalezionych fraz i ich wag,
- `self.time_elapsed` – czas ekstrakcji wyrażony w sekundach,
- `self.similarity_top_keyphrases` – lista par fraz i ich wag odnalezionych w pliku „master”,
- `self.similarity_comparison_keyphrases_map` – mapa, w której kluczami są nazwy przeszukanych artykułów, a wartościami są listy par fraz i ich wag.

Dzięki temu możliwe jest odświeżenie pola tekstowego z wynikami, np. po zmianie wartości progowej wag artykułów do wyświetlenia bez potrzeby ponownego wykonywania obliczeń.

W akcjach reagujących na kliknięcie przycisków ekstrakcji fraz odbywa się tworzenie odpowiedniej instancji klas AbstractContentProvider oraz KeyphraseExtractor, opisanych w dalszej części. Następnie wywoływana jest metoda odpowiedzialna za samą ekstrakcję i kolejna metoda zwracająca frazy, których sumaryczna waga nie przekracza określonej przez użytkownika wartości. W dalszej kolejności wykonywana jest klasteryzacja, jeśli użytkownik zaznaczył tę opcję oraz wygenerowanie łańcucha znaków z rezultatem i wypisanie go w polu tekstowym.

W przypadku akcji wyszukującej podobnych artykułów procedura jest podobna, chociaż trochę bardziej rozbudowana. Oprócz utworzenia providera i exstractora dla głównego pliku lub artykułu, trzeba też stworzyć listę providerów dla pozostałych plików w tym samym katalogu lub dla artykułów, do których linkuje główny artykuł. Następnie dla wszystkich tych providerów tworzony jest MultipleProvidersKeyphraseExtractor wydobywający frazy kluczowe, których porównywaniem zajmuje się klasa DocumentKeyphrasesComparator. Po utworzeniu mapy z artykułami i ich frazami, na jej podstawie generowany jest łańcuch znaków z rezultatem, który zostaje wpisany w polu tekstowym.

6. Interfejs programistyczny

Moduł ekstrakcji fraz kluczowych znajduje się w pliku AKE.py. W tej części dokumentu opisano najważniejsze z klas i metod wchodzących w skład tego modułu.

Klasa `System` zawiera główną logikę konsolowej aplikacji oraz pomocnicze metody zwracające ciągi znaków na podstawie znalezionych fraz, klastrów oraz podobieństw dokumentów. Jej najważniejsze metody to:

- `run` – główna metoda aplikacji konsolowej, w której następuje stworzenie odpowienich obiektów, wykonanie ekstrakcji fraz kluczowych oraz wypisanie wyniku i czasu działania,
- `get_keyphrases_string` – pomocnicza metoda zwracająca gotowy do wypisania ciąg znaków na podstawie przekazanej w argumencie listy par frazy kluczowej i jej wagi,
- `get_clustered_keyphrases_string` – zwraca ciąg znaków na podstawie przekazanej mapy, w której kluczami są nazwy klastrów, a wartościami listy fraz,
- `get_document_similarity_string` – zwraca ciąg znaków na podstawie przekazanej listy par tytułów artykułów i ich stopnia podobieństwa

Klasa `KeyphraseExtractor` jest odpowiedzialna za logikę ekstrakcji fraz kluczowych z pojedynczego artykułu oraz klasteryzację, jej najważniejsze metody to:

- `extract_keyphrases_by_textrank` – zwraca listę par fraz kluczowych i ich wag, zawiera wywołania metod tej klasy, będących krokami algorytmu,
- `_tokenize_text` – dzieli cały tekst na słowa, które następnie są lematyzowane,
- `_extract_candidate_words` – zwraca listę słów, które potencjalnie mogą należeć do fraz kluczowych, czyli przymiotniki i rzeczowniki,
- `_build_graph_from_candidates` – zwraca obiekt typu Graph z biblioteki networkx, w którym wierzchołkami są potencjalne słowa kluczowe i są połączone krawędziami, jeśli te słowa sąsiadowały w tekście artykułu,
- `_build_word_pagerank_ranks_from_graph` – wykonuje algorytm PageRank i zwraca mapę, w której kluczami są najlepsze słowa, a wartościami ich wagi,
- `_merge_keywords_into_keyphrases` – łączy znalezione słowa we frazy kluczowe, jeśli te słowa sąsiadowały ze sobą w treści artykułu i zwraca mapę, w której kluczami są frazy, a wartościami uśredniona waga słów wchodzących w skład tej frazy,
- `_normalize_weights` – normalizuje wagi fraz tak, aby ich suma wynosiła 1,
- `clusterize` – klasteryzuje odnalezione frazy kluczowe i zwraca mapę, w której kluczami są nazwy klastrów, a wartościami listy fraz.

Klasa `MultipleProvidersKeyphraseExtractor` służy do ekstrakcji fraz kluczowych pochodzących z wielu źródeł poprzez kilkukrotne użycie klasy `KeyphraseExtractor`. Zawiera metodę:

- `extract_keyphrases_map_by_textrank` – zwraca mapę, w której kluczami są tytuły artykułów, a wartościami listy par jego fraz kluczowych i ich wag.

Abstrakcyjna klasa `AbstractContentProvider` definiuje standardowy sposób dostarczania treści artykułu niezależnie od jego źródła pochodzenia. Jej metody to:

- `get_content` – abstrakcyjna metoda, której implementacja powinna zwracać ciąg znaków reprezentujący treść artykułu,
- `get_title` – metoda zwracająca tytuł artykułu.

Klasa `WikipediaContentProvider` stanowi implementację klasy `AbstractContentProvider` reprezentującą treść artykułu z Wikipedii. Jej implementacja metody `get_content` pobiera artykuły z wikipedii i konkatenuje ich zawartość.

Klasa `FileContentProvider` jest implementacją klasy `AbstractContentProvider` reprezentująca treść artykułu z pojedynczego pliku. Jej implementacja metody `get_content` odczytuje dany plik i zwraca jego zawartość.

Klasa `DirectoryContentProvider` jest ostatnią implementacją `AbstractContentProvider` i reprezentuje treści artykułów ze wszystkich plików w pojedynczym katalogu. Jej metoda `get_content` przeszukuje wszystkie pliki wewnątrz danego katalogu i konkatenuje ich zawartość.

Klasa `WikipediaPageFinder` służy do pobierania stron z Wikipedii i zawiera jedną metodę:

- `get_wikipedia_page` – zwraca obiekt typu WikipediaPage pochodzący z biblioteki wikipedia.

Klasa DirectoryContentLister służy do przeszukiwania plików w danym katalogu i zawiera tylko jedną metodę:

- `get_content_list` – zwraca listę ścieżek do plików znajdujących się w danym katalogu.

Klasa `DocumentKeyphrasesComparator` odpowiedzialna za logikę porównywania podobieństwa pomiędzy frazami kluczowymi z różnych artykułów. Jej najważniejsza metoda to:

- `compare` – zwraca podobieństwo między dokumentami w postaci mapy, w której kluczami są tytuły artykułów, a wartościami liczbowy stopień podobieństwa

7. Rezultaty

Ocena, które wyrazy i frazy najlepiej podsumowują korpus tekstowy, jest trudna. Poniżej wypisano ranking fraz kluczowych uzyskanych dla artykułu na Wikipedii o tytule „Linux”. Wydają się być intuicyjnie zgodne ze spodziewanymi w artykule zagadnieniami.

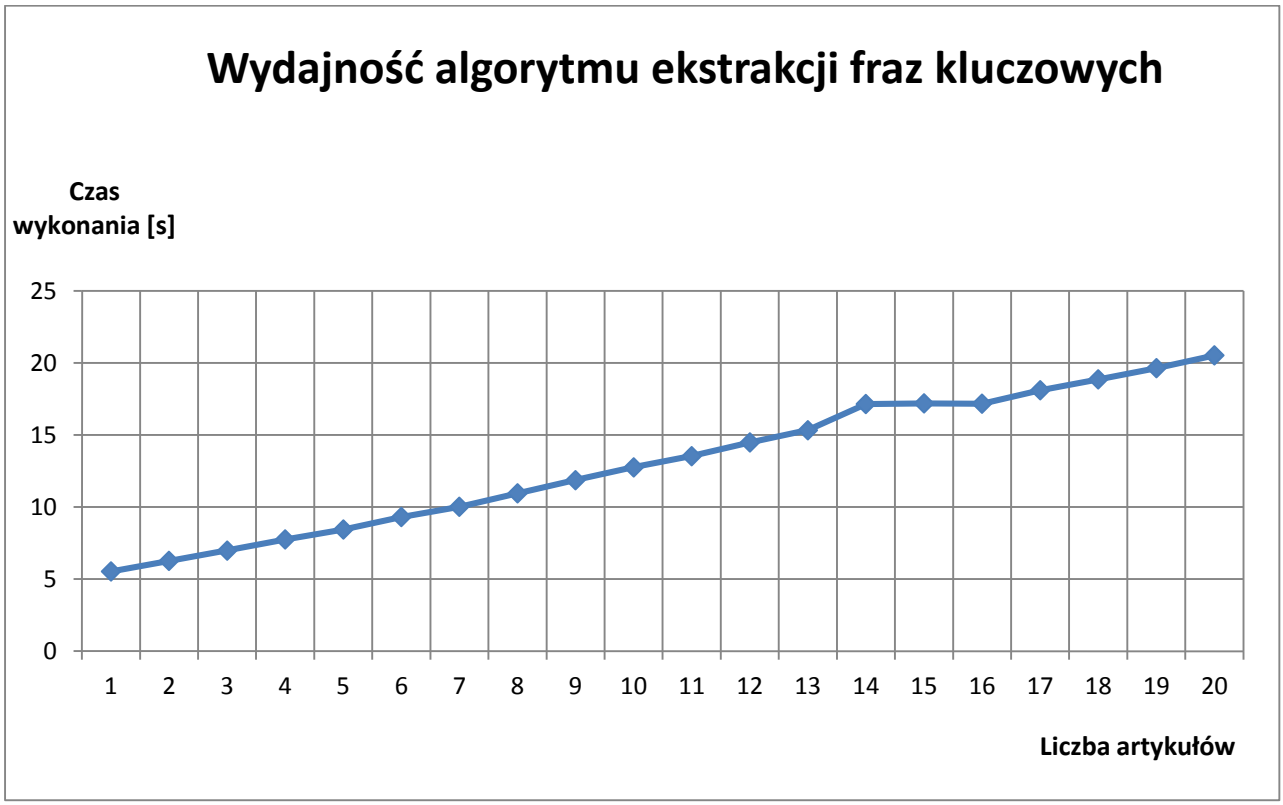
```
linux                                : 0.0094071668
linux system                         : 0.0076900915
linux distribution                   : 0.0067519241
system                               : 0.0059730162
desktop linux                        : 0.0056895815
linux desktop                        : 0.0056895815
linux kernel                         : 0.0056502344
linux server                         : 0.0055545326
support linux                        : 0.0054426424
linux support                        : 0.0054426424
linux application                    : 0.0054268801
linux user                           : 0.0053541777
linux component                      : 0.0053293873
linux version                        : 0.0053103274
ubuntu linux                         : 0.0052454574
linux market                         : 0.0052265695
use linux                            : 0.0052251001
linux community                      : 0.0050665312
many linux distribution              : 0.0050452610
linux name                           : 0.0050065084
name linux                           : 0.0050065084
linux distribution support           : 0.0049939887
linux foundation                     : 0.0049552020
free linux distribution              : 0.0049536272
trademark linux                      : 0.0049157827
fedora linux                         : 0.0048988209
hat linux                            : 0.0048946157
linux derivative                     : 0.0048837674
linux gaming                         : 0.0048799490
suse linux                           : 0.0048799055
linux installed                      : 0.0048753873
year linux                           : 0.0048736831
linux vendor                         : 0.0048682914
linux focus                          : 0.0048506184
enterprise linux                     : 0.0048383517
several linux distribution           : 0.0048372351
customized linux                     : 0.0048201513
```

8. Wydajność

Przeprowadzono testy wydajnościowe zaimplementowanego programu. Jako zbiór danych wejściowych posłużyły manualnie wybrane artykuły na Wikipedii dotyczące informatyki. Wybrane artykuły są rozbudowane. Charakteryzują się podobną długością i posiadaniem przynajmniej kilku wieloakapitowych sekcji.

Pomiary wykonano na laptopie z zainstalowanym systemem operacyjnym Windows 7 64-bit oraz 4-rdzeniowym procesorem Intel i7-3612 QM.

Czasy pracy algorytmu od 1 do 20 artykułów zmierzono 5 razy dla zmiennej kolejności artykułów oraz uśredniono. Wyniki pomiarów prezentuje wykres na rys. 4.



Rys. 4. Wykres wydajności algorytmu ekstrakcji fraz kluczowych.

9. Podsumowanie

W ramach projektu udało się stworzyć uniwersalne narzędzie, pozwalające na streszczanie dokumentów poprzez określanie ich fraz kluczowych, grupowanych w klastry, jak i stwierdzanie, czy inne dokumenty z nimi porównywane charakteryzują się podobną treścią. Program przyjmuje dokumenty w postaci plików tekstowych lub artykułów o zadanych tytułach pobieranych z Wikipedii. Oferuje funkcjonalność z poziomu interfejsu konsolowego oraz graficznego jako program standalone. Realizowana funkcjonalność jest dostarczona z wykorzystaniem stosunkowo prostych algorytmów.

Projekt może być kontynuowany z akcentem położonym na różne elementy funkcjonalności. Na zakończenie autorzy zamieszczają propozycje kierunków dalszego rozwoju:

- poprawa wydajności zaimplementowanych algorytmów, w szczególności ekstrakcji fraz kluczowych, np. optymalizując liczne operacje dokonywane na kolekcjach;
- przyspieszenie porównywania mastera z linkowanymi artykułami Wikipedii, obecnie mogącego trwać wiele minut ze względu na sekwencyjne pobieranie po jednym artykule;
- atrakcyjna wizualnie prezentacja wyników działania algorytmów w GUI;
- przerobienie programu na aplikację serwerową, udostępniającą funkcjonalność klientom mobilnym poprzez API lub w przeglądarce internetowej;
- projekt i implementacja modelu danych, pozwalającego na przechowywanie wyników przeprowadzanych ekstrakcji fraz kluczowych, w celu ich przeglądania np. w aplikacji webowej.