Karol Łyskawiński 137335  
Jakub Słota 137405

Inteligentne Wyszukiwanie Informacji – sprawozdanie z projektu

## 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.

## Zastosowany algorytm

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 unigramy, które można połączyć w wielowyrazowe frazy już po uzyskaniu rankingu. Natomiast miarę podobieństwa określa wspólne występowanie wyrazów w oknie mieszczącym   
N wyrazów, typowo 2-10.

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, wyznaczona na podstawie empirycznego progu sumy wag, 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 jest normalizowana względem liczby tworzących   
je wyrazów.

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.

Na koniec możliwe jest wykonanie klasteryzacji znalezionych fraz, czyli zgrupowania ich wokół powtarzających się słów. Warunkiem zaklasyfikowania frazy do klastra jest występowanie w niej danego wyrazu. Pod uwagę nie jest brana długość frazy ani kolejność jej słów, a każda fraza może należeć jednocześnie do kilku klastrów.

Przewidziano możliwość porównywania dokumentów powiązanych z głównym dokumentem. Porównywanie zaczyna się od ekstrakcji fraz kluczowych z poszczególnych dokumentów pobocznych. Następnie frazy kluczowe dokumentu głównego i pobocznych są sprowadzane do zbiorów wyrazów. Podobieństwo dokumentu głównego i pobocznych określa się przez cosinus kąta pomiędzy wektorami złożonymi z uzyskanych zbiorów wyrazów.

## ~~Zastosowane narzędzia~~

Projekt został zrealizowany w języku programowania Python w wersji 2.7. 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.

## ~~Funkcjonalność~~

Zaimplementowany program pozwala na wydobywanie najlepszych fraz kluczowych z tekstu pochodzącego z wybranego przez użytkownika źródła. Do obsługiwanych rodzajów wejścia należą:

1. Wikipedia – korpus tekstu stanowi artykuł lub zbiór artykułów o tytułach podanych przez użytkownika, oddzielonych przecinkiem, np. „Linux, Python (programming language)”;
2. Pojedynczy plik – korpusem tekstu jest plik tekstowy znajdujący się pod wskazaną ścieżką;
3. Katalog – program tworzy korpus tekstu odczytując wszystkie pliki umieszczone w katalogu znajdującym się pod wskazaną ścieżką.

Na wyjściu programu podawane są najlepsze frazy kluczowe wraz z przypisaną do nich wagą. Mierzony jest czas wykonania algorytmu ekstrakcji fraz kluczowych.

Program jest też w stanie wyznaczyć słowa będące klastrami i przypisać do nich znalezione frazy. Zwrócone klastry posortowane są po liczbie należących do nich elementów.

Możliwe jest wybranie opcji szukania dokumentów podobnych do głównego, określanego mianem „master”. W przypadku wejścia będącego artykułem z Wikipedii analizowane są artykuły, do których prowadzą linki z artykułu głównego. Ze względu na liczność linków – typowo kilkaset – jest to długotrwały proces. W przypadku pojedynczego pliku wejściowego, program przeanalizuje pozostałe pliki zawarte w tym samym katalogu, co „master”. Zależnie od wybranego rodzaju wejścia, na wyjściu zostaną wypisane tytuły artykułów na Wikipedii lub ścieżki do plików na dysku, mających podobną zawartość do głównego dokumentu wraz z ich miarą podobieństwa.

## 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

## 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. 1.

Rys. 1. Wykres wydajności algorytmu ekstrakcji fraz kluczowych.

## 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 7. 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.

## 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 krypt *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.

## 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 , którego *j*-ty element wynosi , gdzie , jeżeli wyraz *j* występuje w *i*-tym dokumencie oraz 0 w przeciwnym wypadku.
* – waga wyrazu *j* w dokumencie *i*, zakładamy, że .
* Długość wektora – wektor ma długość   
  | .
* Iloczyn skalarny wektorów –
* Cosinus kąta między wektorami – , 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 – d1 (master) na temat języka Python i Java oraz d2 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 d1 i d2

|  |  |
| --- | --- |
| **Frazy kluczowe dokumentu d1 (master)** | **Frazy kluczowe dokumentu d2** |
| 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 d1 i d2. Kontrastowym kolorem czcionki wyróżniono wyrazy występujące we frazach kluczowych obu dokumentów.

Tab. 2. Unikatowe wyrazy wydobyte z fraz kluczowych dokumentów d1 i d2

|  |  |
| --- | --- |
| **Wyrazy dokumentu d1** | **Wyrazy dokumentu d2** |
| python | python |
| language | language |
| scripting | scripting |
| programming | programming |
| application | popular |
| java |  |
| community |  |
| compiler |  |

Tabela 3 przedstawia wektory reprezentujące dokumenty d1 i d2 w przestrzeni wektorowej wyrazów. Kontrastowym kolorem czcionki wyróżniono kolumny odpowiadające wyrazom występującym we frazach kluczowych obu dokumentów.

Tab. 3. Wektory reprezentujące dokumenty d1 i d2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Wektor** | **python** | **java** | **language** | **scripting** | **programming** | **popular** | **application** | **community** | **compiler** |
| d1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| d2 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

W tej chwili są gotowe dane potrzebne do obliczeń:

Otrzymano wynik wskazujący, że treść dokumentów d1 i d2 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 można w łatwiejszy sposób obliczać cosinusowe podobieństwo dokumentów, niż z definicji. Można zauważyć, że iloczyn 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:

– zbiór unikatowych wyrazów dokumentu d1

– zbiór unikatowych wyrazów dokumentu d2

## 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.