Rok akademicki 2014/2015

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej

Praca Dyplomowa Inżynierska

Wiktor Ślęczka

Analizator pojęć zawartych w nazwach DNS

Opiekun pracy:
dr inż. Mariusz Kamola

Ocena pracy: .....

.....
Data i podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Systemy informacyjno-decyzyjne

Data urodzenia: 26.05.1992

Data rozpoczęcia studiów: 01.10.2011

Życiorys

Urodziłem się 26 maja 1992 roku. W 2011 roku ukończyłem XIV LO im. Stanisława Staszica w Warszawie. Następnie w październiku tego samego roku rozpocząłem studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Informatyka. Zawodowo zajmuję się budowaniem stron internetowych oraz projektowaniem architektury serwerów aplikacji webowych.

Streszczenie

Tytuł: *Analizator pojęć w nazwach DNS*

Celem pracy było stworzenie narzędzia do znajdywania grafu kolokacji słów w nazwach domen internetowych, zarejestrowanych w języku polskim. Analiza odbywa się na bazie częstości występowania poszczególnych par słów w pojedynczych nazwach. Równolegle z tym zadaniem celem pracy było wykorzystanie grafu, otrzymanego w wyniku analizy, do pogrupowania słów w spontanicznie stworzone kategorie na bazie ich kolokacji.

W ramach pracy zostały utworzone dwa programy. Pierwszy program służy do sporządzenia grafu. Przekształca on listę domen w listę ważonych krawędzi grafu zawierających połączenia pomiędzy słowami. Drugi program wczytuje utworzony w ten sposób graf, a następnie przeprowadza na nim proces analizy skupień, dążąc do połączenia węzłów najbardziej ze sobą powiązanych w spójne kategorie.

Wyniki oraz wnioski z przeprowadzonej analizy znajdują się w treści pracy.

Abstract

Title: *The analyser of names in DNS addresses*

The goal was to create a tool able to produce the graph of collocations between the words, which are contained in the online domain names registered for the Polish language. The analysis is carried out based on the frequency of particular pairs of words in a single domain name. The secondary task in the study was to use the graph, obtained by analysis, to group found words into spontaneously created categories, based on their collocations.

As part of the work there were created two programs. The first program is used to prepare the graph. It converts a list of domains into a graph with weighted edges, containing all connections between words. The second program reads this graph, and then runs the process of cluster analysis, aiming to connect closest nodes together into coherent categories.

The results and conclusions from the analysis can be found inside this thesis.

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Jak dotąd pierwsza i druga dekada XXI wieku są okresem niezwykle dynamicznego rozwoju technologii służących do komunikacji, a zwłaszcza Internetu. Sieć ta zaczyna mieć coraz większy wpływ nie tylko na nasze życie, ale również na organizację środowisk biznesowych, sposoby zarządzania firmami czy prowadzenia reklamy.

Jednak współczesny Internet nie byłby w stanie powstać, gdyby nie został stworzony system DNS, czyli rozproszona usługa stanowiąca hierarchiczną bazę danych zawierającą informacje na temat wszystkich domen zarejestrowanych w tej sieci. Najważniejszą z tych informacji jest adres IP oraz port maszyny, która oferuje pożądaną usługę.

Sam system powstał w roku 1983 jako alternatywa dla ręcznie zarządzanych plików zawierających przypisanie nazwy do adresu hosta. Była to praktyka stosowana wewnątrz sieci ARPANET, prekursora Internetu, oraz w pierwszych latach działania „sieci sieci", jak czasem określany jest Internet [1][2].

1.2 Zarys problemu

W porównaniu z innymi aspektami technologii dotyczących sieci, samemu nazewnictwu stron internetowych nie poświęca się prawie wcale uwagi, co wydaje się szczególnie dziwne obecnie, w erze Big Data i bardzo intensywnego rozwoju różnych technik sztucznej inteligencji. Problemem, którym będę się zajmował w ramach niniejszej pracy jest przeprowadzenie analizy częstości występowania obok siebie kolokacji par słów w nazwach domen internetowych. Innymi słowy będzie to określenie, jak często występują obok siebie poszczególne pary słów, a następnie wykorzystanie tej wiedzy w praktyce.

Należy zaznaczyć, że nie przeprowadzam operacji na poszczególnych wyrazach, lecz jako słowa rozumiem zbiór wszystkich odmian danego leksemu. W związku z tym analiza, którą przeprowadziłem skupia się na zbiorach wyrazów o tych samych znaczeniach, ale innych formach leksykalnych, a nie na samych wyrazach.

1.3 Cel pracy

Celem pracy jest stworzenie narzędzia umożliwiającego analizę kolokacji występowania poszczególnych słów w nazwach domen internetowych. Poprzez przeprowadzenie analizy rozumie się tutaj stworzenie grafu obrazującego badane zagadnienie, co stanowi jej pierwszą część. Kolejna część pracy polega na wykorzystaniu otrzymanego grafu w praktyce. Będzie to narzędzie służące do grupowania wierzchołków, tworząc przy tym kategorie w sposób spontaniczny.

1.4 Motywacja

Analiza częstości występowania słów oraz ich poszczególnych kolokacji w nazwach domen może dostarczyć bardzo interesujących oraz przydatnych danych, zwłaszcza dla osób rejestrujących domenę internetową. Na takiej bazie można stworzyć narzędzia służące do klasyfikacji zawartości domeny bazując na jej nazwie. Dodatkowo można wysunąć hipotezę, że w nazwie strony internetowej użycie częściej występującego słowa może wpłynąć na ilość wizyt, aczkolwiek udowodnienie tej tezy może być niezwykle trudne.

Samo zadanie jest dodatkowo interesujące z tego względu, że jest ono całkowicie innowacyjne. W publicznie dostępnej przestrzeni wiedzy nie ma żadnych informacji na temat metod ani istniejących rozwiązań podobnego zagadnienia. Oczywiście nie oznacza to, że nikt nigdy się tym nie zajmował. W następnym podrozdziale (1.5) jest podane kilka przykładów prac, których autorzy musieli przynajmniej częściowo zająć się tym samym problemem, na którym skupia się moja praca. Niestety, z jakichś powodów nie podzielili się oni wynikami ani narzędziami, których użyli do otrzymania wyników opisanych swoich opracowaniach.

1.5 Analiza istniejących rozwiązań

Pomimo wyszukiwania opracowań podobnych tematów, nie udało mi się znaleźć żadnych dotyczących rozwiązania podobnego problemu. Co prawda istnieją artykuły opisujące wyniki pracy z podobnych zagadnień [3], ale wszystkie są silnie ukierunkowane w stronę badanego problemu, i skupiają się na wynikach, nie na ogólnym sposobie ich otrzymania. Prace te są krok przed tym, co jest moim zadaniem w tej pracy, i mogłyby się oprzeć na moich wynikach oraz algorytmach.

1.6 Potencjalne wykorzystanie

Rozwiązywany problem może mieć wiele zastosowań. Głównym z nich może być możliwość sugerowania alternatywnych nazw dla domen internetowych, czyli na przykład program, który dla podanej zajętej nazwy domeny zwróci nazwę alternatywną, ale podobną znaczeniowo i używającą słów, które maksymalizują liczbę wejść na stronę (innymi słowy, są najczęściej używane w nazwach innych domen). Innym możliwym wykorzystaniem jest na przykład zwiększenie skuteczności oraz wydajności klasyfikacji zawartości domen internetowych, wprowadzając informacje bazujące na ich nazwach. Dodatkowo wybór słów znajdujących się w nazwie domeny

ma również pewien wpływ na optymalizację w wyszukiwarkach internetowych[1]. Dodatkowo istnieje cała gama dość zaskakujących zastosowań, z których wymienię na przykład możliwość wykrywania adresów rozsyłających spam [3].

1.7 Układ pracy

Dalsza struktura pracy jest następująca:

- Rozdział 2 zawiera omówienie problemów, które należało wziąć pod uwagę w celu rozwiązania postawionego problemu.
- W rozdziale 3 znajdują się słowne opisy stworzonego rozwiązania wraz z zastosowanymi oraz wypróbowanymi w nim algorytmami i narzędziami, a także uzasadnienie ich wyboru.
- Rozdział 4 skupia się na szczegółowej specyfikacji wytworzonych programów.
- W treści rozdziału 5 jest opis oraz analiza otrzymanych przeze mnie wyników.
- Pracę kończy rozdział 6, w którym znajduje się opis dalszych możliwości rozwoju projektu oraz podsumowanie całości pracy.

[1] Jest to jedynie opinia obiegowa, bardzo trudna do zweryfikowania, ponieważ wyszukiwarki nie publikują algorytmów, a zmiana nazwy domeny zmienia inne czynniki, które mają wpływ na wynik

Rozdział 2

Charakterystyka problemu

2.1 Analiza problemu

Zadanie, które musiałem wykonać polegało na wykonaniu czterech operacji. Pierwszą z tych operacji było znalezienie słów w nazwach domen. Problem dokładniej opisuje w punkcie 2.2.1, a jego rozwiązanie w następnym rozdziale, w sekcji 3.2. Z racji, że w języku polskim słowa podlegają odmianie, następnym krokiem musi być powiązanie poszczególnych odmian tego samego leksemu. Można to osiągnąć na kilka sposobów. W praktyce jednak najczęściej wykonuje się takie zadanie poprzez znalezienie rdzenia słowa lub jego lemmy, czyli formy podstawowej. Krok ten jest niezbędny, ponieważ jeśli dwa znalezione słowa reprezentują to samo znaczenie, ale są w innej odmianie, to do celów przeprowadzanej analizy powinienem uwzględnić je jako to samo słowo. Na opisie tego zagadnienia skupię się w części 2.2.2.

Kolejnym etapem jest samo stworzenie grafu, który może być naprawdę dużą strukturą danych, zawierającą połączenia kilku milionów węzłów ze sobą. Przewidywane problemy podczas tworzenia tego modułu zawiera sekcja 2.2.3. Ostatnia część tego rozdziału, oznaczona 2.2.4, skupia się na procesie grupowania węzłów w grafie w spontanicznie utworzone grupy, używając metody bazującej na analizie skupisk węzłów[2].

2.2 Szczegółowa analiza

2.2.1 Znajdowanie słów w nazwach domen

Znalezienie słów w nazwie domeny oznacza, że dla każdego podanego ciągu znaków należy go podzielić na wchodzące w jego skład tokeny. Token jest to pojęcie w lingwistyce analogiczne do tokenu występującego w trakcie budowy parsera, czyli pojedyncza jednostka leksykalna, a w tym przypadku jest to wyraz znajdujący się w analizowanym ciągu znaków [4]. W kroku tym należy również odfiltrować te znaki, które nie stanowią poprawnych słów. Proces ten nazywa się tokenizacją. Dla języków z należących do rodziny indoeuropejskich zwykle odbywa się poprzez dzielenie tekstu przy wystąpieniu białych znaków.

[2] Ang. *Cluster analysis*

Wydaje się to być najtrudniejszą częścią tego projektu. Trudność ta jest spowodowana przede wszystkim cechami, które niesie z sobą sama forma ciągu liter, jaki stanowi nazwa domeny. Mianowicie, w ogólnym przypadku forma ta cechuje się:

- Niejednoznacznością
- Ambiwalencją
- Brakiem kontekstu
- Dużą liczbą odmian

Niejednoznaczność

Fakt znalezienia określonego układu słów wcale nie oznacza, że odpowiedź ta jest poprawna. Można z takim samym sukcesem rozbić zwrot *niebo* na partykuły *nie* oraz *bo*, jak i potraktować je jako rzeczownik *niebo*. W związku z tym potrzebny jest mechanizm wykrywania oraz rozstrzygania takich niejednoznaczności.

Ambiwalencja

Nie ma żadnej gwarancji, że w nazwie domeny znajdzie się jakiekolwiek słowo, ani że nie wystąpią całkowicie losowe ciągi znaków. Co więcej, takie ciągi znaków mogą wejść w interferencję ze słowami, które w analizowanym tekście się znajdują, tworząc w ten sposób dodatkowe niejednoznaczności. Dlatego też w trakcie przetwarzania znalezienie kandydatów na słowa nie jest wystarczające. W analizowanym tekście należy też poszukać powodów, dla których znalezione słowo jest bądź nie jest poprawnym rozkładem. Czynnik ten zauważalnie komplikuje rozstrzyganie niejednoznaczności.

Brak kontekstu

Znakomita większość metod przetwarzania języka naturalnego czy klasyfikacji tekstu jest przeznaczona do operacji na poziomie zdań lub większych jednostek logicznych tekstu, ponieważ wymagają kontekstu słowa do działania [5]. Dzieje się tak, ponieważ pojedyncze słowa lub zbitki słów nie niosą ze sobą wystarczającej ilości informacji. W przypadku rozkładu słów w nazwach domen nie istnieje żaden kontekst. Właściwie to nie można nawet stwierdzić, że w tym przypadku istnieje jakakolwiek gramatyka. W związku z tym użycie do rozwiązania tego problemu bardziej zaawansowanych technik z zakresu przetwarzania tekstu czy sztucznej inteligencji nie jest możliwe.

Liczba odmian

W języku polskim większość słów podlega odmianie i występuje w kilku, a uwzględniając zdrobnienia i inne możliwe formy językowe, może występować nawet w kilkunastu różnie brzmiących formach. Dla pokazania skali tego problemu, słownik stworzony dla języka polskiego zawierający większość słów we możliwych wszystkich odmianach[3] zawiera 3.621.020 pozycji, natomiast analogiczny słownik dla języka

[3]Słownik mojego autorstwa można znaleźć pod tym adresem `https://github.com/wikii122/Dictionary`

angielskiego zawiera około 1.022.000 pozycji[4]. Wynika z tego, że korzystanie z predefiniowanych słowników zawierających wyrazy do tokenizacji może być niewydajne zarówno pod względem użytej pamięci, jak i czasu.

Wykrywanie niejednoznaczności

Mając kilka możliwych niejednoznacznych rozkładów określonej frazy, trzeba znaleźć sposób, aby określić która z możliwości jest poprawna. Ponieważ, jak zostało wcześniej wspomniane, słowa w nazwie domeny są pozbawione kontekstu, jakiekolwiek wnioskowanie na ten temat jest w praktyce całkowicie niemożliwe. W związku z tym należy odwołać się do możliwości, jakie niesie ze sobą statystyka i korzystać jedynie z własności znalezionych słów.

Należy wyznaczyć odpowiednią miarę zależności od tych własności. Niestety jest ich stosunkowo mało. Przy tworzeniu odpowiedniej metryki można uwzględnić miedzy innymi:

- częstość występowania słowa w języku pisanym,
- karę za długość słowa,
- nagrodę za długość słowa,
- zależności pomiędzy występowaniem sylab lub N-gramów.

Oczywiście metody wykorzystujące własności liter wymagają przeprowadzenia ogromnej liczby obliczeń i zgromadzenia wielkiej ilości danych dotyczących występowania obok siebie różnych elementów struktury słowa. Mimo to nie dają one żadnej gwarancji dobrej jakości rezultatów, a byłyby niezwykle czasochłonne, w związku z czym należy pozostawić je jako ostateczność na wypadek, gdyby inne metody nie dały zadowalających wyników. Na początek należy wypróbować miary wykorzystujące łatwiejsze do zdobycia dane. Największe nadzieje budzi miara oparta na częstości występowania słów, ale należy sprawdzić również inne możliwości.

Metoda ta ma pewną wadę. Mianowicie, mając kilka możliwych sposobów rozbioru ciągu znaków, zawsze zostanie wybrana ta sama. Jest to problem, ponieważ wybrany rozbiór raz może być poprawny, ale w innym przypadku może być już całkowicie błędny. Ponieważ nie istnieje kontekst słowa, jedynym rozwiązaniem jest wybranie najbardziej prawdopodobnej możliwości. Nie ma jednak żadnego sposobu sprawdzenia ani potwierdzenia poprawności wybranej możliwości. Oczywiście zawsze istnieje prawdopodobieństwo, że uda się znaleźć lepszy sposób wykrywania słów, niż opisane w tutaj.

2.2.2 Identyfikacja leksemów

Po zidentyfikowaniu poszczególnych wyrazów wchodzących w ciąg analizowanej nazwy należy pogrupować je według znaczeń. Znaczenie w tym kontekście jest wąskim pojęciem i oznacza wyrazy będące inną formą gramatyczną tego samego słowa. W związku z tym należy przypisać dla każdego znalezionego wyrazu jakąś formę indeksu, która byłaby wspólna dla wszystkich wariantów odmian tego samego słowa.

[4]Według statystyk Google z 2009 roku

Najbardziej oczywistym sposobem, jak to zrobić jest klasyfikacja metodą najbliższego sąsiada, używając jako miary na przykład odległości Levenshteina. Metoda ta jest stosunkowo prosta, ale istnieją inne, które mogą okazać się znacznie dokładniejsze i wydajniejsze.

Metody znajdowania tematu, lemmy oraz rdzenia słowa

Inną rodziną metod, z których można skorzystać, jest określenie wyższych jednostek leksykalnych znalezionych słów i użycie ich jako kluczy. Do form, które można w ten sposób znaleźć zalicza się temat słowa, jego lemma oraz rdzeń. W następnej części rozdziału krótko omówię każde z tych pojęć.

Rdzeń słowa

Zgodnie z definicją, rdzeń to główny morfem wyrazu, pozbawiony wszelkich afiksów [6]. Nie wnikając w szczegóły dotyczące lingwistyki, oznacza to, że rdzeń to część wyrazu, która nie jest przedrostkiem ani przyrostkiem. Interesującą cechą jest to, że nie jest zależny od odmiany ani od formy. Nie zależy też od tego, jaką częścią mowy jest analizowany wyraz.

W tym przypadku przeszkodą jest fakt, że pomimo poszukiwań nie udało mi się znaleźć żadnego narzędzia ani algorytmu gwarantującego znalezienie poprawnej formy rdzenia słowa.

Lemma słowa

Lemma to inaczej forma podstawowa danego słowa. Przy czym w czasie lematyzacji, czyli znajdowania lemmy, zachowane są cechy słowa, takie jak liczba w przypadku rzeczownika czy czas czasownika. Dodatkowo problemem, który praktycznie uniemożliwia użycie tej metody jest fakt, że większość narzędzi służących do lematyzacji korzysta z kontekstu słowa w czasie działania. Inną trudnością jest rozpoznawanie homonimów. Są to słowa mające identyczną formę, ale inne znaczenie. Do rozpoznania formy podstawowej w tej sytuacji również powinno się użyć kontekstu słowa.

Temat słowa

Temat słowa to część wyrazu, która nie ulega zmianie w czasie odmiany [6]. Jego tworzenie polega na odcięciu ze słowa wszystkich końcówek fleksyjnych, czyli tych przedrostków oraz przyrostków, które zależą od formy gramatycznej słowa.

Do ogólnego użytku przyjęło się, że jest wystarczające, jeżeli wynik narzędzia przeprowadzającego znajdowanie tematu zwróci tą samą wartość dla wszystkich spokrewnionych słów. Niewątpliwą zaletę stanowi też fakt, że takie narzędzie nie wymaga do działania kontekstu. Co więcej, jego realizacja jest możliwa na zasadzie słownikowej, używając bazy typu klucz-wartość. W związku z tym narzędzie to wydaje się być najbardziej obiecującym z do tej pory analizowanych.

Niestety, ma ono również wady. Przede wszystkim, znaleziony temat nie jest całkowicie niezależny od cech analizowanego słowa. W odróżnieniu od rdzenia, wynik zależy na przykład od części mowy analizowanego wyrazu. W związku ze sposobem działania narzędzi tego typu, problem wynikający z obecności homonimów będzie

znacznie bardziej widoczny. Wyniki otrzymane przez narzędzie przeprowadzające takie operacje nie zawsze będą jednoznaczne.

Algorytm SoundexPL

Alternatywą do użycia wyżej wymienionych metod jest użycie zupełnie innego podejścia, wykorzystującego inne cechy znalezionego słowa. Jedną z takich alternatyw jest algorytm Soundex, w wersji dla języka polskiego opracowanej jako część pracy magisterskiej przez Michała Kosmulskiego [7]. Proponuje on użycie algorytmu fonetycznego, który przypisuje każdemu słowu 4-znakowy kod, zależny od brzmienia słowa. Kod otrzymuje się ze słowa, w którym usuwa się znaki nie zmieniające brzmienia, następnie przypisuje się symbol do znaków i dwuznaków o tym samym brzmieniu. Poszukiwana wartość to pierwsze cztery symbole w znalezionym kodzie. Jednak, jak sam autor zaznacza, jakość takiego rozwiązania jako alternatywy do narzędzia przeprowadzającego proces stemowania jest bardzo niska.

Homonimy

Jak już zostało wcześniej wspomniane, dużą przeszkodą w tym elemencie projektu będą homonimy. Są to te wyrazy, które można interpretować na różne sposoby, pomimo faktu, że składają się na nie te same litery. Niestety, ponownie muszę stwierdzić, że dla analizowanych słów nie istnieje żaden kontekst, w związku z czym rozstrzygnięcie wieloznacznych wyników w sposób jednoznaczny nie jest możliwe. Z taką niejednoznacznością można poradzić sobie na kilka sposobów:

- Każdą możliwość potraktować jako poprawną. Wiąże się to z wprowadzeniem do przestrzeni odpowiedzi wielu fałszywych danych.

- Przypisanie poszczególnym możliwościom wagi równej prawdopodobieństwu ich poprawności ($P = \frac{1}{Liczbamozliwosci}$). Ta opcja również wprowadza zakłócenia związane z wprowadzeniem niepoprawnych danych, ale w mniejszym stopniu. Wadą jest też zmniejszenie wagi poprawnych elementów.

- Traktowanie pierwszej możliwości jako poprawnej. Może to usunąć poprawną odpowiedź i dodać błędną, ale w przeciwieństwie do pozostałych opisanych metod dodawana jest dokładnie jedna błędna możliwość, więc w pewnych przypadkach błąd może zmaleć.

- Przypisanie rozwiązaniom wag, ale biorąc pod uwagę częstość występowania każdego znaczenia. Z przedstawionych możliwości ta wydaje się być obarczona najmniejszym błędem, ale wymaga ona obszernego słownika frekwencyjnego dla znaczeń słów. Takie dane oczywiście dla nazw domen nie istnieją, a zdobycie ich byłoby tożsame ze znalezieniem rozwiązania dla opisanego w tej części problemu problemu.

Wymienione możliwości należy w trakcie realizacji pracy ocenić i wybrać obarczoną najmniejszym błędem.

Stoplista

Dodatkowym problemem w trakcie przetwarzania na pewno okażą się niektóre słowa, które są wynikiem błędu w procesie tokenizacji, ale występują na tyle często, żeby zachwiać wynik. Do takich słów zapewne będą należeć na przykład krótkie spójniki. Dlatego należy pozbyć się tego rodzaju zakłóceń poprzez stworzenie listy słów zakazanych, które byłyby usuwane z dalszego przetwarzania. Ponieważ usunięcie znalezionego słowa z próbki może wprowadzić fałszywe powiązania, najprawdopodobniej będzie należało taką próbkę całkowicie usunąć z puli.

2.2.3 Przechowywanie zależności między słowami w pamięci

Struktura tworzonego grafu, ponieważ to jest najlepsza forma do reprezentacji analizowanych danych, jest prosta. Węzłami w grafie będą znalezione indeksy dla słów, niezależne od cech słowa. Natomiast krawędzie będą ważone liczbą wystąpień konkretnej pary słów w danych wejściowych. A bardziej szczegółowo, krawędzie będą określone sumą wag poszczególnych znalezionych elementów, wynikającą ze sposobu rozstrzygania problemu homonimów.

Przechowywanie otrzymanych danych w pamięci operacyjnej będzie albo najprostszym, albo najtrudniejszym z zadań danego projektu. Zapis na nośnik fizyczny raczej nie będzie tworzył problemów, zwłaszcza biorąc pod uwagę pojemności współczesnych dysków. Natomiast ilość danych mieszcząca się w pamięci RAM jest bardzo ograniczona. Wstępne określenie ilości danych, jakie zajmą zależności nie jest na tym etapie możliwe. Najbardziej pesymistyczny przypadek to gdy otrzymany graf ma formę grafu pełnego, czyli wszystkie jego wierzchołki są ze sobą połączone. Przypadek ten na szczęście nie jest zbyt prawdopodobny.

Sama implementacja grafu musi być przemyślana w taki sposób, żeby była możliwa jego wydajna modyfikacja i przeszukiwanie. Będą to elementy niezbędne do przeprowadzenia procesu grupowania. Ważne jest również, żeby nie istniała potrzeba na każdym kroku powielania całego grafu, czyli aby istniała było możliwość zapisania w jakiejś formie aktualnego wyglądu grafu na dysku.

2.2.4 Grupowanie węzłów w grafie

Ostatnim etapem projektu będzie pogrupowanie węzłów w grafie. Proces ten odbędzie się na zasadzie grupowania aglomeracyjnego, w sposób iteracyjny. W każdej iteracji zostaną wybrane dwa węzły w grafie. Wybór będzie kierował się na razie nieokreśloną miarą odległości, zależnej od wag na krawędziach grafu. Jeśli będą to dwa słowa, zostaną one usunięte z grafu a w ich miejsce zostanie wstawiona grupa słów. W pozostałych przypadkach znalezione słowo lub grupa zostanie dołączona do drugiej grupy w przetwarzanej parze. Krok ten będzie powtarzany do momentu, aż wszystkie krawędzie w grafie będą usunięte. W ten sposób, każdy samodzielny (spójny) podgraf w niespójnym grafie będzie sprowadzony do jednej końcowej grupy, o ile nie składa się z on z tylko jednego słowa, co nie jest możliwe przy dodawaniu par słów. W optymalnym przypadku przetwarzanie skończy się z jedną grupą. Wymaganą do tego celu informacją jest w jaki sposób można oceniać bliskość pomiędzy wierzchołkami grafu. Cechy, jakie można brać pod uwagę to, między innymi:

- Waga krawędzi

- Suma wag krawędzi wchodzących do wierzchołka
- Liczba krawędzi wchodzących do wierzchołka
- Liczba elementów zawartych w grupie, jeśli wierzchołek jest grupą
- Liczba elementów grupy, które miało połączenie do danego wierzchołka
- Liczba wierzchołków połączonych z analizowanymi wierzchołkami

Jednakże to, co w trakcie przetwarzania jest interesujące, to nie jego rezultat, ale stany pośrednie. Właśnie w stanach pośrednich znajdują się poszukiwane cechy przeprowadzonego grupowania. Jest to zbiór znalezionych grup, które wciąż są w trakcie łączenia. Początkowe elementy oraz końcowe najprawdopodobniej nie będą interesujące, z uwagi na małe zróżnicowanie wyników w zależności od użytej miary. Stan początkowy oczywiście zawsze zawiera niepogrupowane słowa. W stanie końcowym powinna znaleźć się dokładnie jedna grupa w ramach każdej spójnej składowej, zawartej wewnątrz grafu. Natomiast pozostałe, te bliższe środka, będą zawierały odpowiedzi na interesujące mnie w tej pracy pytania. Do pytań, na które poszukiwać będę odpowiedzi, należą te:

- Czy powstaną grupy powiązane w jakiś sposób znaczeniowo, jako synonimy?
- Czy będą miały inne, bardziej abstrakcyjne powiązania, w formie na przykład utartych zwrotów?
- Czy istnieją powiązania w inny sposób niż znaczeniowo, na przykład fonetycznie?
- Czy może nie istnieją żadne cechy wspólne dla znalezionych grup?

Rozdział 3

Rozwiązanie

3.1 Architektura

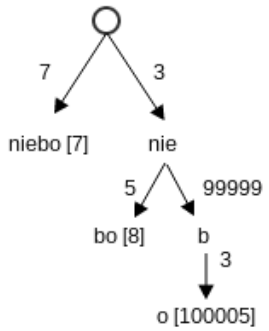
Utworzone rozwiązanie składa się z 3 komponentów, zgrupowanych w dwa wykonywalne skrypty. Pierwszy skrypt służy do przekształcania listy domen internetowych w listę połączeń pomiędzy wierzchołkami grafu, zawierającą wagi poszczególnych krawędzi. Drugi skrypt służy do pogrupowania wierzchołków gotowego grafu w kategorie. W tym rozdziale opiszę techniki oraz metody, których użyłem do rozwiązania problemu, natomiast w rozdziale 4 znajdą się szczegóły implementacyjne projektu.

3.2 Tokenizator

Tokenizator jest modułem służącym do znajdowanie tokenów w tekście. Narzędzie to przyjmuje jako dane wejściowe domenę internetową, natomiast zwraca listę słów, które zostały w analizowanym ciągu znaków znalezione. Nazwa domeny jest przetwarzana tak, aby zminimalizować różnice pomiędzy różnymi formami zapisu. Przede wszystkim, wielkość liter nie jest brana pod uwagę. Dla celów pracy nie jest to potrzebna informacja. Zwłaszcza, że zgodnie ze specyfikacją, system DNS nie rozpoznaje wielkości liter [8].

Wykrywanie słów

Działanie algorytmu znajdowania słów można zobrazować w ten sposób. Podany ciąg znaków jest rozkładany do formy drzewa, zawierającego wszystkie możliwe rozkłady na słowa. Wszystkim kombinacjom przypisany jest opisany poniżej koszt, różny dla każdego wyrazu. Wartościom nieuwzględnionym w słowniku przypisywana jest bardzo wysoka kara. Następnie wybierana jest ta ścieżka, dla której koszt przejścia do końca drzewa jest najmniejszy. Taki rozkład, z pominięciem większości błędnych ścieżek dla czytelności, pokazany jest na rysunku 3.1.



Rysunek 3.1: Uproszczony rozkład słowa niebo

Waga

Do wykrywania słów sprawdziłem wiele rodzajów wag oceniających rezultat znajdowania, między innymi te opisane w rozdziale 2.2.1. Najlepsze rezultaty w trakcie działania udało się osiągnąć metodą opartą na prawie Zipfa, czyli korzystającą pośrednio z częstości występowania słowa w języku, z wprowadzoną niewielką nagrodą za długość słowa.

Prawo Zipfa opisuje rozkład częstości występowania słów w języku w zależności od ich umiejscowienia w rankingu częstości występowania. Ranking ten to po prostu lista słów posortowana malejąco według liczby wystąpień słowa w korpusie języka, numerowana od jedynki. Zgodnie z tym rozkładem, liczba wystąpień słowa na indeksie o numerze N będzie równa $L_N = (\frac{L_1}{N})^A$, gdzie A jest stałe i $A \approx 1$. Wynika z tego, że drugie co do występowania słowo występuje dwa razy rzadziej niż pierwsze, trzecie trzy razy rzadziej, i ta tendencja jest kontynuowana dla wszystkich słów [9].

Dysponując odpowiednio posortowaną listą słów można na tej zasadzie stworzyć miarę, która pozwala na rozkład słowa metodą opisaną powyżej z zadowalającą jakością. Ujmując problem bardziej dokładnie, można stwierdzić, że prawdopodobieństwo wystąpienia konkretnego słowa jako następnego, nie analizując w żaden sposób gramatyki czy znaczenia, wynosi:

$$P_w(n) = \frac{1}{\log(n * \log N)}$$

gdzie n jest to pozycja słowa w rankingu wystąpień, a N stanowi całkowitą liczbę słów.

Dodatkowo odkryłem, że wprowadzenie niewielkiej nagrody za długość słowa nieznacznie poprawia wyniki, ale wprowadza przekłamania przy najczęściej używanych słowach, więc postanowiłem nie korzystać z tej modyfikacji.

3.3 Stemmer

Do sprowadzenia tekstu do jednorodnej formy postanowiłem użyć gotowego narzędzia o nazwie Morfologik[5]. Służy ono do znajdowania formy podstawowej słowa i jest przez twórców nazywany mianem stemmera. Słowo to pochodzi od angielskiej nazwy procesu znajdowania wspólnych oznaczeń dla wyrazów pochodzących

[5]Strona projektu: https://github.com/morfologik/morfologik-stemming

z tej samej rodziny, zwykle stanowiących temat słowa, z angielskiego właśnie stem. Jednakże nie musi to być koniecznie temat, ponieważ tak naprawdę jedynym warunkiem stawianym narzędziom tego rodzaju jest zwracanie tych samych indeksów dla słów pochodzących z tej samej rodziny. Z tego też powodu, aby uniknąć niejednoznaczności, w dalszej części pracy wynik działania tego narzędzia będę nazywał właśnie stemem.

Jest to zasadniczo jedyne publicznie dostępne narzędzie tego typu dla języka polskiego. Posiada ono również swoje wady. Główną wadą, na jaką się natknąłem, była niedeterministyczna liczba rezultatów zwracanych w przypadku wystąpienia niejednoznaczności co do formy słowa, oraz wystąpienia homonimów, jak słusznie przewidziałem w podrozdziale 2.2.2. Niestety program nie informował, ile linii wyjścia dotyczy pojedynczego słowa. Kolejny problem był natury wydajnościowej i związany był z uruchamianiem przy każdym odpytaniu programu[6] całej maszyny wirtualnej Javy.

W związku z tym musiałem rozwiązać ten problem poprzez serializację wyników przetwarzania omawianego narzędzia, zwracanych dla wszystkich słów w używanym przeze mnie słowniku, do pliku, w celu przerobienia ich do użytecznej dla mnie formy. Zwłaszcza, że zgodnie z założeniami opisanymi w rozdziale 3.2, otrzymany słownik stemów należy pozbawić polskich znaków. W trakcie tego procesu zauważyłem też jeszcze jedną przeszkodę, a mianowicie pokrycie dla słów przez Morfologika w porównaniu z moim słownikiem jest stosunkowo niskie. Nie znaczy to na szczęście, że posiada on mało słów, lecz jedynie że pokrywa około 85% słów znajdujących się w używanym przeze mnie słowniku, co może, lecz nie powinno, wpłynąć na wynik. W każdym razie słowa nie obejmowane poprzez słownik stemów z oczywistych przyczyn nie mogą być brane pod uwagę.

Wynik pracy tego skryptu to lista par stemów, reprezentująca leżące obok siebie w nazwie domeny słowa, z przypisaną wagą będącą ilorazem miar jakości wyników procesu stemowania. Przy czym jeżeli znaleziono tylko jedno słowo, również znajduje się ono na liście jako osobna pozycja. Natomiast jeśli rozpoznano więcej niż dwa słowa, każda para leżących obok siebie słów jest dodana do listy. Analogicznie, w przypadku kilku możliwych interpretacji, wszystkie wykryte rozwiązania również są dodawane do listy jako osobne pary, przy czym ich waga jest odpowiednio mniejsza, zgodnie z wcześniejszym opisem. Dlatego też pary te nie stanowią krawędzi grafu, a jedynie części składowe tych krawędzi. Zastosowałem takie rozwiązanie, ponieważ w ten sposób zostanie zachowana informacja o wystąpieniach poszczególnych par, która może być przydatna do dalszego przetwarzania.

Problem homonimów

Z homonimami zdecydowałem radzić sobie poprzez przypisanie każdej kombinacji prawdopodobieństwa, że jest to poprawna interpretacja. Ponieważ na tym etapie na temat słowa nie ma praktycznie żadnych informacji, które mogłyby wpływać na wartość tego prawdopodobieństwa, założyłem rozkład jednorodny dyskretny. Każdemu rozwiązaniu jest przypisywana waga $w = \frac{1}{N}$, gdzie N jest liczbą znalezionych rozwiązań.

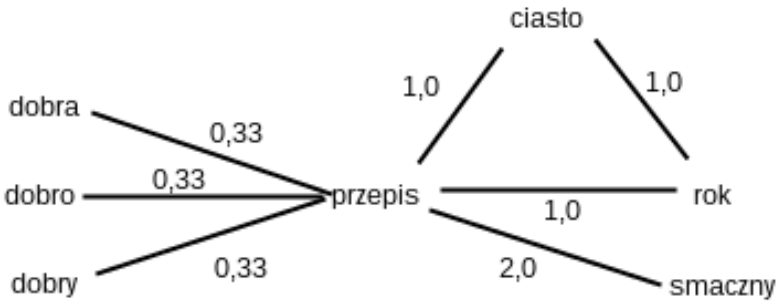
[6] Wielokrotne uruchamianie odbywało w celu uniknięcia błędów związanych z niedeterministycznym wyjściem

Alternatywą mogło być uwzględnienie częstości występowania analizowanego stemu przy określaniu miary. Rozwiązanie to ma jednak pewną wadę, ponieważ do znajdowania słów również używane są częstości ich występowania. W wyniku tego użycie tej metody wzmacniałoby potencjalne błędy, w związku z czym postanowiłem z niej nie korzystać.

3.4 Graf

Graf to zarówno struktura danych, jak i moduł służący do wczytania danych z pliku. Wczytuje on plik wygenerowany przez poprzedni program, a następnie składa z poszczególnych składowych graf w pamięci. Samo narzędzie jest na tyle elastyczne, że zadziała z danymi z dowolnego źródła, tak długo jak będą przestrzegały określonego formatu. Brane są pod uwagę tylko linie zawierające trzy kolumny. Pierwsze dwie to unikalne nazwy wierzchołków. Trzecia to waga składowej krawędzi pomiędzy tymi wierzchołkami. Linie zawierające pojedyncze słowa (dwie kolumny) są ignorowane, ponieważ nie wnoszą żadnych dodatkowych informacji do grafu.

Tworzona struktura danych jest dokładniej opisana w rozdziale 4.3. Natomiast jej interpretacja jest następująca. Na początku przetwarzania każdy węzeł zawiera jedno słowo. Jest ono połączone krawędziami z innymi słowami, obok których występowało dane słowo, jak pokazano na rysunku 3.2. Waga każdej krawędzi oznacza siłę połączenia, czyli częstość występowania poszczególnych słów, z uwzględnieniem modyfikatorów z poprzednich kroków.



Rysunek 3.2: Przykładowy graf utworzony dla nazw przepisroku.pl, dobreprzepisy.com, smaczny-przepis.com, smaczneprzepisy.pl, przepisy-ciasto.net, ciastoroku.pl

3.5 Grupowanie

Grupowanie jest iteracyjnym procesem, polegającym na złożeniu silnie połączonych węzłów w grupy. W każdym kolejnym kroku wybierane są dwa węzły. Wybór następuje według odległości, znajdując węzły które są sobie najbliższe, jako para o największej mierze bliskości. Miarą tą jest iloraz wagi krawędzi przez sumę wszystkich krawędzi wchodzących do węzłów. Następnie jeśli wybrana para zawiera przynajmniej jedną grupę, drugi węzeł jest dołączany do takiej grupy. Natomiast

w przeciwnym przypadku oba słowa są usuwane z grafu, a na ich miejsce tworzona jest grupa zawierająca oba słowa. Do niej przeniesione są wszystkie połączenia usuniętych elementów z innymi węzłami w grafie.

Proces ten jest powtarzany aż do momentu, gdy przestają istnieć powiązane ze sobą węzły. Wynikiem przetwarzania jest więc jedna grupa na każdą spójną składową grafu. Ale dla mojej pracy to nie wynikowe grupy są w trakcie procesu interesujące, lecz stany pośrednie. Po zakończeniu pracy programu należy ręcznie obejrzeć zmiany powstałe w każdym kroku, a następnie ocenić w jakim stopniu słowa w utworzonych w każdej iteracji grupach są ze sobą powiązane.

Sukcesem będzie sytuacja, gdy w skład większości grup będą wchodzić słowa powiązane znaczeniowo. W tym wypadku wyniki pracy będą miały całą gamę zastosowań, które będą mogły być w przyszłości rozwijane. Do nich zalicza się na przykład przypisanie poszczególnym grupom kategorii, do których należą zawarte w nich słowa.

Rozdział 4

Specyfikacja

4.1 Technologia

Do celów pracy powstały dwa programy, oraz kilka skryptów pomocniczych. Implementacja wszystkich elementów pracy została zrealizowana w języku Python, w wersji 3.4.2. Do uruchomienia narzędzi może być wymagana instalacja dodatkowych bibliotek, których lista znajduje się w pliku *requirements.txt*, wraz z numerem użytej wersji dla każdego pakietu[7]. Ostatecznie na liście zależności znajduje się jedynie biblioteka tqdm, odpowiedzialna za pokazywanie paska postępu w czasie działania programu, wszystkie inne zależności pochodzą z biblioteki standardowej Pythona.

4.2 Program analyzer

Działanie

Program analyzer znajduje się w katalogu *analyzer*, a jego uruchomienie następuje poprzez komendę *python3 analyzer.py*. Do działania wymaga pliku zawierającego listę domen o nazwie *names.txt* znajdującego się w tym samym katalogu, natomiast jego wynikiem jest plik *stems.txt*, zawierający krawędzie grafu wraz z wagą poszczególnych składowych krawędzi.

Oczekiwany format pliku z nazwami domen to plik tekstowy zawierający jedną nazwę w każdej linii. Natomiast format pliku z krawędziami to lista zawierająca dwie lub trzy kolumny w linii. W przypadku dwóch kolumn jest to stem słowa, które wystąpiło jako jedyne w nazwie, oraz jego waga. Natomiast gdy są trzy kolumny, to są to dwa stemy, symbolizujące parę słów, w kolejności alfabetycznej, oraz waga. Przy czym należy tu zauważyć, że ani stemy, ani pary stemów nie są unikalnymi kombinacjami, w związku z czym mogą występować wielokrotnie, a każde wystąpienie należy traktować jako osobną składową tej samej krawędzi. Innymi słowy, wagi krawędzi pomiędzy tymi samymi elementami powinno się ostatecznie do siebie dodać. Nie jest to wykonywane w ramach programu, ponieważ wiąże się to z pewną stratą informacji, która może się okazać wartościowa dla narzędzia przetwarzającego dane wyjściowe. Średni czas działania programu to około półtorej godziny dla danych opisanych w podrozdziale 5.1.1.

[7]Tak zwane zamrożone zależności

Użyte algorytmy i struktury danych

Do przechowywania licznych danych używam głównie list[8]. W przypadku, gdy niezbędne jest przypisanie wartości do kluczy albo w jakikolwiek inny sposób skojarzenie dwóch wartości, używam słowników[9]. Wyjątkiem są struktury, które okazały się zbyt duże, by zmieścić się do pamięci operacyjnej. Do nich należą dwa skojarzenia - słowa z kosztem wyliczonym na bazie jego częstości występowania oraz słowa z jego formą podstawową. W tych przypadkach użyłem pakietu *shelve*[10]. Jest to rodzaj indeksowanej łańcuchami znaków mapy, trzymającej przechowywane dane na dysku twardym przy użyciu protokołu serializacji pickle.

Z algorytmów, których użyłem przede wszystkim można wyszczególnić ten służący do tworzenia oraz przeszukiwania drzewa wszystkich możliwych rozkładów zbioru znaków na słowa. Cechuje się on złożonością czasową $O(n^2)$ oraz pamięciową $O(n)$, a jego przebieg wygląda następująco:

1. Rozpocznij z listą kosztów zawierającą wartość zero
2. Dla każdego i od zera do długości analizowanego ciągu znaków
   (a) Stwórz pustą listę T
   (b) Dla każdego j od zera do i dodaj na koniec T sumę obliczonego wcześniej kosztu podciągu od j-tego do i-tego znaku analizowanego napisu i pozycji j z listy kosztów
   (c) Dodaj do listy kosztów najmniejszą wartość z listy stworzonej w poprzednim punkcie
3. Stwórz pustą listę znalezionych słów
4. Przypisz l długość analizowanego łańcucha i dopóki l jest większe od zera
   (a) Z podciągów kończących się na l-tym znaku ciągu wybrać ten, którego suma kosztu oraz obliczonego dla poprzedzającego łańcucha w liście kosztów jest równa kosztowi z końca listy kosztów.
   (b) Znaleziony łańcuch należy dodać do listy znalezionych słów
   (c) l zmniejszyć o długość znalezionego słowa
5. Lista znalezionych słów zawiera słowa w odwrotnej kolejności i należy ją odwrócić

Pozostałe algorytmy użyte w programie można określić jako na tyle trywialne, że mogą zostać pominięte jako mało interesujące.

4.3 Program grouper

Działanie

Drugim programem powstałym w ramach pracy jest grouper. Odpowiada za niego plik grouper.py, uruchamiany przy użyciu polecenia *python3 grouper.py*. Jako

[8]Tu należy zauważyć, że lista w Pythonie różni się od przyjętej definicji tej struktury
[9]Struktura danych nazywana zwykle HashMapą

wejście wczytuje on plik *stems.txt*, utworzony przez poprzedni program, natomiast jego wynikiem jest plik *result.shelve*, trwały słownik biblioteki shelve zawierający stany grafu z każdej iteracji. Średni czas działania narzędzia po podaniu na wejście wyników poprzedniego programu, otrzymanych z danych opisanych w 5.1.1, wynosi około 5 dni.

Graf

Implementacja grafu zrealizowana jest w następujący sposób. Główny kontener grafu zawiera zbiór wszystkich wierzchołków, a z kolei każdy wierzchołek posiada słownik, indeksowany innymi węzłami. W słowniku tym znajdują się wszystkie wierzchołki, z którymi dany wierzchołek jest połączony, a przypisana do niego jest waga (na tym etapie już przetworzona) odczytana z danych wejściowych. Są dwa typy węzłów - słowo i grupa. Słowo oferuje podejrzenie połączeń z innymi węzłami oraz na operację połączenia z innym słowem. W wyniku tej operacji oba węzły zostają usunięte, a do grafu wstawiany jest nowy wierzchołek, grupa utworzona z obu słów. Grupa oferuje dodatkowo operację łączenia z inną grupą. Łączenie ze słowem w przypadku grupy polega na usunięciu słowa z grafu i dodaniu do grupy. Natomiast połączenie grup usuwa jedną z grup z grafu, a wszystkie jej słowa dodawane są do grupy drugiej. W każdym wypadku połączenia z usuwanego elementu dodawane są do grupy przyjmującej słowa.

Grupowanie

Proces grupowania jest iteracyjny. Każda iteracja polega na znalezieniu pary węzłów o najmniejszej odległości - największej wadze w tym przypadku, ponieważ waga odzwierciedla bliskość. Następnie, dla znalezionej pary przeprowadzana jest operacja grupowania. Iteracja ta jest powtarzana tak długo, jak długo pozostają w grafie połączone ze sobą wierzchołki. W związku z tym, w trakcie procesu każda spójna składowa grafu zostaje sprowadzona do dokładnie jednego wierzchołka.

Jest wiele sposobów, w jaki dokładnie obliczana jest bliskość. W dalszej części pracy będę nazywał sposoby obliczania bliskości kluczami. Przede wszystkim, ale nie wyczerpująco, można wskazać następujące formy kluczy:

- $e1.sum$
- $value$
- $\frac{e1.sum}{e1.elements}$
- $\frac{value}{e1.sum}$
- $\frac{value}{e1.cons}$
- $\frac{e1.cons+e2.cons}{e1.elements+e2.elements}$
- $\frac{value}{e1.sum+e2.sum}$
- $\frac{value}{e1.elements+e2.elements}$

gdzie:

- e1, e2 to analizowane elementy
- parametr cons to liczba krawedzi wychodzących z wierzchołka
- parametr sum to suma wag wszystkich krawędzi wchodzących do wierzchołka
- parametr elements to liczba elementów zawartych w grupie (1 dla napisu)
- value to waga krawędzi między e1 i e2

Wyniki z przeprowadzonych prób dla różnych sposobów wyliczania bliskości zaprezentowane będą w następnym rozdziale w części 5.2.

4.4 Dodatkowe skrypty

W ramach pracy stworzyłem też kilka niewielkich narzędzi do automatyzacji niektórych procesów. Większość znajduje się w katalogu *tools*. Do nich zaliczają się przede wszystkim:

- listdiff.py - skrypt służący do przetworzenia plików .shelve zwracanych wskutek działania programu grouper. Zwraca na standardowe wyjście słowa w grupie, która została stworzona wskutek każdej iteracji.
- prepare_dict.py, prepare_dict_freq.py - narzędzia służące do oczyszczania słownika z polskich znaków oraz duplikatów.

Rozdział 5

Wyniki

5.1 Graf

5.1.1 Dane wejściowe

W trakcie przetwarzania udało mi się otrzymać kilka stosunkowo interesujących informacji dotyczących samego grafu. Na wejściu programu znalazł się plik zawierający pięćdziesiąt milionów nazw domen. Po przefiltrowaniu pod kątem duplikatów oraz najczęściej spotykanych pozycji wygenerowanych maszynowo w pliku pozostało dokładnie 3,272,538 unikalnych pozycji. Na bazie tych danych został sporządzony graf.

5.1.2 Własności grafu

Utworzony graf po przetworzeniu, polegającym przede wszystkim na połączeniu krawędzi biegnących miedzy tymi samymi wierzchołkami, posiada 33,839 wierzchołków, do których dochodzą krawędzie. Dodatkowo na wyjściu programu *analyzer* w 961,109 nazwach nie znaleziono par, ale pojedyńcze słowa. W takiej roli wystąpiło 54,267 różnych słów.

Można wysnuć z tego faktu wniosek, iż prawdopodobieństwo, że dowolna, arbitralnie wybrana, nazwa domeny będzie zawierała nieuwzględnione w grafie słowo wynosi około 1%. W grafie można wyróżnić 24 spójne składowe, o różnej wielkości, wahającej się od kilku wierzchołków do kilkudziesięciu tysięcy. Oznacza to, że w rezultacie przeprowadzonych eksperymentów uzyska się 24 niepołączone grupy.

5.2 Rezultaty eksperymentów

Otrzymane wyniki

Dla większości analizowanych kluczy wyniki są stosunkowo trudne do zinterpretowania. Poza nielicznymi przypadkami bardzo ciężko znaleźć powiązanie znaczeniowe między słowami w poszczególnych grupach. Należy tu zauważyć, że niewielkie, na przykład trzyliterowe grupy zawierają oczywiście słowa jak najbardziej ze sobą powiązane, ale gdy liczność grupy przekroczy chociażby 6 słów, do grup zaczynają dołączać bardzo odlegle związane z nią słowa, bądź związane blisko, lecz z niewielką częścią słów zawartych w grupie.

Oczywiście, zdarzają się wyjątki. Na przykład, bardzo dobry wynik stanowi następująca grupa:

```
['utlenic', 'woda', 'utleniony', 'swiecony', 'oligocen',
   'wypompowywac', 'odsalac', 'sniegi', 'pogolic',
   'witaminizowac', 'zmiekczacz', 'przypalic', 'kolonski',
   'destylowac', 'kryniczny', 'natleniac', 'perfumowac',
   'uzdatniac', 'zrodlany', 'zasadowy', 'toaletowa']
```

Jak widać wszystkie słowa związane są w jakiś sposób z wodą. Innym przykładem może być grupa związana z pożyczaniem pieniędzy

```
['warianta', 'kredyt', 'hipotetyczny', 'ochmistrz', 'dolarowy',
   'lombardowy', 'wariant', 'refinansowanie', 'przyplyw',
   'zaciagac', 'zlotowkowy', 'wielopokoleniowy', 'preferencyjny',
   'ekspresowo', 'refinansowac', 'akceptacyjny', 'redyskonto',
   'konsumpcyjny', 'udzielac', 'dogodny', 'ulepszyc']
```

Na podstawie tych grup widać też, dlaczego w ogólności jakość otrzymanych rezultatów dość szybko się degeneruje. Problem stanowi przede wszystkim rodzaj słów, jakie są dołączane do grupy. Jest to pojedynczy rzeczownik, oraz wiele przymiotników go opisujących, oraz niekiedy czasownik. W takiej sytuacji, w grupie zaczynają dominować przymiotniki i przyciągają one inne słowa, które często z nimi występują. Nowo dodane słowa niekoniecznie muszą być związane w jakiś sposób z istniejącymi w grupie rzeczownikami, natomiast mogą być opisywane przez inny zestaw przymiotników. Sprawia to, że temat grupy zostaje przesunięty w stronę nowo dodanych słów, co w kolejnych iteracjach prowadzi do następnych przesunięć. W skutek tego coś, co można nazwać „znaczeniowym tematem" grupy, czemu można przypisać kategorię się rozmywa i w kolejnych krokach traci jakiekolwiek znaczenie. Taką grupą jest na przykład

```
['palatium', 'hotel', 'beta-karoten', 'przepioreczka', 'mocarz',
   'mizantropia', 'bazuna', 'nadpis', 'kategoryzacja',
   'przasniczka', 'liburna', 'zubrowka', 'butikowy', 'nidzki',
   'senacki', 'behapowiec', 'berberys', 'podstoli', 'dobrodziej',
   'hrabski', 'benefis', 'bakista', 'szafarnia', 'habenda',
   'basztowy', 'zawrzec', 'gniesc']
```

Najlepszy wynik

Dla jednego klucza wyniki były nieco inne, i znacznie ciekawsze. Klucz ten wyrażony jest wzorem (oznaczenia zdefiniowane w rozdziale 4.3)

$$\frac{value}{e1.elements + e2.elements}$$

i zarówno przebieg grupowania, jak i jakość grup w jego przypadku były zdecydowanie inne niż w pozostałych przypadkach.

Samo grupowanie do połowy odbywało się normalnie, ale od około 16-tysięcznej iteracji jedna z grup zaczęła intensywnie rosnąć, a po kolejnych dwóch tysiącach iteracji rozpoczęła kolejno dołączać do siebie kolejne grupy, całkowicie dominując proces. Zjawisko jest to jak najbardziej naturalne, i może, a nawet powinno się zdarzyć, ale w pozostałych przypadkach działo się tak dopiero pod koniec przetwarzania, a nie w połowie cyklu. Natomiast to, co przede wszystkim wyróżnia ten klucz to bardzo duża ilość grup którym można przypisać znaczenie.

```
['udarowy', 'wiertarka', 'mlot', 'wlocznia', 'mloto',
   'nalot', 'nisko', 'pradowy', 'silownik',
   'pneumatyczny', 'wkretak', 'elektromagnetyczny',
   'zawora', 'zaczep', 'kulowy', 'zawor', 'zwrotny',
   'wciagarka', 'nozyce', 'uniwersalny', 'nozyca',
   'gilotyna', 'hydrauliczny', 'mikrofalowy', 'kuchenka',
   'gazowy', 'latarnia', 'butel', 'kompozytowy', 'butla',
   'elektryczny', 'instalacja', 'centralny', 'odkurzacz',
   'centralne', 'odkurzac', 'zadaszenie', 'swietlik',
   'zadaszyc', 'basenowy', 'promiennik', 'tarasowy',
   'deska', 'podlogowy', 'ogrzewanie', 'ogrzewac',
   'kolektor', 'sloneczny', 'trakcyjny', 'bateria',
   'akumulator', 'modul', 'solarny', 'naprawczy',
   'zestaw', 'reanimacyjny', 'zestawic', 'panel',
   'regulator', 'temperatura', 'nastawnik', 'akredytowac',
   'rejestrator', 'czujka', 'pozarowy', 'czujek', 'dym',
   'cofac', 'czujnik', 'alarmowy', 'mierniczy',
   'przyrzad', 'pomiarowy', 'przetwornik', 'miarowy',
   'aparatura', 'krawiectwo', 'reduktor', 'cisnienie',
   'systema', 'ukryc', 'kamera', 'termowizyjny',
   'pomiara', 'pomiar', 'kopula', 'geodezyjny', 'wizyjny',
   'gogle', 'monitoring', 'alarm', 'system', 'ogniwo',
   'paliwowy', 'ogniwac', 'wodorowy', 'hydrofor',
   'zbiornik', 'napalic', 'amatorka', 'dostep',
   'kontrola', 'zapalniczka', 'benzynowy', 'stacja',
   'kynolog', 'alternatywny', 'paliwo', 'cieply', 'pompa',
   'cieplo']
```

Jak widać, większość słów w tej grupie powiązana jest z w jakiś sposób z przemysłem. To tylko jeden przykład, ale znajdowałem dla danych otrzymanych przy użyciu tego klucza inne grupy liczące ponad setkę pozycji, w których dla większości słów również było możliwe znalezienie wspólnej kategorii. Warto zauważyć, że grupy te są o wiele większe niż przy pozostałych sprawdzanych kluczach. W tamtych przypadkach znalezienie grupy 30 słów, w której można było znaleźć taką zależność było sytuacją wyjątkową i ogromnym sukcesem.

Sytuacje te niestety wciąż stanowią wyjątki, ponieważ większość znalezionych grup zawierała słowa w żaden sposób nie powiązane ze sobą znaczeniowo. Możliwe, że byłoby tych grup więcej, i w ostatecznym rozrachunku byłyby lepsze, gdyby nie fakt, że praktycznie połowa przetwarzania polegała na dołączaniu słów do tej samej, całkowicie chaotycznej jeśli chodzi o zawartość, grupy. Co prawda można w niej wyróżnić pewne trendy, ale są one raczej kwestią przypadkową i nie wskazują na istnienie jakichś zależności.

Wadliwe klucze

Część prób nie zakończyła się pomyślnie. Co nie znaczy, że program zawiódł, ale jedynie, że otrzymane wyniki dla niektórych kluczy nie miały żadnego praktycznego znaczenia. W tych przypadkach powstawała jedna grupa, która anektowała wszystkie słowa od początku do końca, a praktycznie żadna inna nie powstawała, przynajmniej w ramach tej samej wspólnej składowej. Zgodnie ze specyfikacją opisaną w 4.3, niemożliwym jest, aby jedna grupa objęła różne spójne składowe, więc grup było dokładnie tyle samo, co niepowiązanych ze sobą ścieżką części grafu, czyli 24. Stało się tak między innymi dla kluczy wyrażonych wzorami:

- $e1.sum$
- $\frac{e1.cons}{e1.elements}$
- $value$

Jest to spowodowane faktem, że połączenie wierzchołków na zasadzie tych własności powoduje zwiększenie analizowanej wielkości klucza dla grupy powstałej w danej iteracji. Jednak połączenia w następnym kroku wybierane są na zasadzie znalezienia maksimum właśnie z tej miary, a ponieważ przynajmniej jeden z elementów pary został wybrany z powodu posiadania największej wartości klucza, jej dodatkowe zwiększenie gwarantuje wybranie ponownie tego elementu.

Analiza wyników opartych na podstawie przetwarzania przy użyciu tych kluczy nic nie wnosi do treści pracy, w związku z czym ją pominę.

5.3 Wnioski

Stworzony graf wnosi kilka interesujących informacji na temat rozkładu słów znajdujących się w nazwach domen, zwłaszcza pod kątem bogactwa słownictwa. Warto zauważyć, że ilość słów używanych w polskich nazwach domen jest bardzo zbliżona do ilości słów znanych przez przeciętnego Polaka, czyli wedle szacunków ok 30,000 [11]. Żeby wnioskować na temat spójności przestrzeni nazw potrzebne byłyby zauważalnie większe zbiory danych, których nie posiadam, a których zdobycie wiązałoby się z gromadzeniem informacji przez lata. Dysponując obecną wiedzą nie mogę powiedzieć nic na temat tego, czy rzeczywiście istnieją zbiory słów, które nie łączą się z grupami, czy to jedynie zbiór danych nie zawierał takiego połączenia.

Widać również, że prawie $\frac{1}{3}$ nazw domen jest krótka i zawiera tylko pojedyncze słowa, a można stwierdzić, że jest ich więcej, ponieważ podana wyżej liczba nie uwzględnia tych nazw, w których nie znaleziono żadnych wyrazów, a mogą zawierać na przykład słowo zniekształcone, skrót (wp.pl) lub słowo zagraniczne (allegro.pl). Dodatkowo prawie 2 miliony nazw jest krótsze, a jedynie nieco ponad milion dłuższe niż średnia długość nazwy, która wynosi 16 znaków. Fakt ten nasuwa wniosek, że ludzie wolą krótkie, łatwe do wpisania identyfikatory domen niż długie, opisowe w których dodatkowo można się pomylić.

Przeprowadzone eksperymenty poza kilkoma wyjątkami nie dały oczekiwanych rezultatów. Szybki przyrost zakłóceń i rozmycie się tematu grupy po niewielkiej ilości iteracji pokazuje, że nie istnieją żadne rewolucyjne zależności, które zmieniłyby nasze postrzeganie Internetu. Co więcej, wynika z tego, że użyty sposób grupowania nie nadaje się do analizy danych tego rodzaju. Winą w tym wypadku powinno się obarczyć brak kontekstu dla znalezionych słów, przez co do danych wprowadzane są niejednoznaczności i błędy których nie można uniknąć, a dodatkowo utrudnia to zastosowanie bardziej zaawansowanych technik w rodzaju wnioskowania. Problemem jest również sama struktura danych. To nie są zdania, lecz niezwykle krótkie wyrażenia, zwykle złożone z dwóch lub trzech słów, z których najczęściej jedno jest rzeczownikiem, a pozostałe określają ten rzeczownik. Jednak ten sam przymiotnik może opisywać niezwiązane z sobą rzeczowniki, bądź samemu być z nim niezwiązanym. Na przykład „zielony” nie niesie ze sobą żadnych informacji o słowie, z jakim

zostało użyte, poza opisem koloru, a kolor taki może mieć zarówno liść, jak i obudowa telefonu. Prowadzi to do powiązania ze sobą całkowicie niezwiązanych pojęć, co skutkuje całkowitym rozmyciem znaczenia grupy.

Rozdział 6

Zakończenie

6.1 Podsumowanie

Celem pracy było stworzenie narzędzi do tworzenia grafu zawierającego zależności częstotliwościowe w nazwach domen, oraz wykorzystanie otrzymanego grafu w praktyce, przeprowadzając grupowanie węzłów oraz analizę wyników. Obie części pracy zostały zakończone sukcesem. W ramach pracy stworzyłem dwa narzędzia realizujące właśnie te cele, jak i również słownik frekwencyjny dla języka polskiego[10].

Pierwszym programem, który napisałem jest analyzer. Służy on do znajdowania poszczególnych wyrazów w nazwach domen oraz sprowadzenia ich do formy podstawowej, aby było można zidentyfikować je niezależnie od odmiany słowa. Efektem jego działania jest lista zawierająca krawędzie grafu. Drugi program nosi nazwę grouper. Jego zadaniem jest wczytanie pliku utworzonego przez poprzednie narzędzie, a następnie przeprowadzenie grupowania na otrzymanych danych. Grupowanie jest procesem iteracyjnym. W wyniku jego działania powstaje plik zawierający zserializowane przy użyciu użyciu protokołu pickle stany ze wszystkich iteracji.

Podczas pisania pracy głównym problemem było to, że przetwarzanie maszynowe języka polskiego nie jest tak rozwiniętą dziedziną, jak analogiczny proces przetwarzany dla języka angielskiego, oraz że język polski jest znacznie bardziej złożony niż większość innych języków.

Mimo zrealizowania założeń pracy, otrzymane rezultaty okazały się mocno rozczarowujące. Przede wszystkim wyniki przeprowadzonych prób grupowania nie okazały się tak obiecujące, jak się zapowiadały w trakcie planowania pracy.

6.2 Dalszy rozwój

Niestety, nie uważam, aby otrzymane wyniki miały jakieś możliwe realne wykorzystanie w praktyce. Otrzymane w skutek grupowania grupy słów posiadają elementy, które nie są ze sobą w wystarczający sposób powiązane, aby móc planować ich wykorzystanie w jakikolwiek sposób. Wyniki przeprowadzonych doświadczeń rzutują na użycie danych na inne sposoby, które napotkają analogiczne, opisane

[10]Słownik oraz źródła można znaleźć pod tym adresem `https://github.com/wikii122/Dictionary`

wcześniej przeszkody. W związku z tymi problemami zastosowanie otrzymanych rezultatów do narzędzi opisanych w części 1.6 w większości wypadków nie wydaje się możliwa do zrealizowania.

Możliwe natomiast jest kontynuowanie pracy poprzez przeanalizowanie innych sposobów grupowania. Wydaje się, że gdyby na początku ustalić kilka predefiniowanych kategorii o zdefiniowanych znaczeniach, a następnie przypisać do każdej po kilka domen grupowanie mogłoby przynieść lepsze rezultaty. Klasyczny klasyfikator, działający chociażby metodą najbliższego sąsiada mógłby osiągnąć użyteczne wyniki. Dodatkowo, na bazie otrzymanych wskutek takiego doświadczenia danych realizacja przynajmniej części użyteczności opisanej w 1.6 byłaby możliwa. Właściwie sama próba byłaby to prosta realizacja klasyfikatora przypisującego kategorię do nazwy domeny.

Oczywiście wielką korzyścią dla dalszego rozwoju projektu mogłoby być też opracowane tokenizatora zdolnego rozwiązywać niejednoznaczności w sposób lepszy niż probabilistyczny. Można to osiągnąć na przykład poprzez dodatkowe przeanalizowanie zawartości strony i otrzymanie w ten sposób kontekstu dla znalezionych w nazwie domeny słów. Jest to jednak rozwiązanie stosunkowo trudne i zdecydowanie wykracza poza zakres tej pracy.

Bibliografia

[1] A. Tanebaum, *Sieci Komputerowe*, 4th ed. Helion, 2004.

[2] J. Klensin. (2003) Role of the domain name system (dns). IETF. [Online]. Available: https://tools.ietf.org/html/rfc3467

[3] K. E. S. Wei Wang, "Breaking bad: Detecting malicious domains using word segmentation," At&T, 2015. [Online]. Available: http://arxiv.org/pdf/1506.04111.pdf

[4] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools (2Nd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006, p. 111.

[5] M. Baites, "Models of natural language understanding," 2003. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC40721/

[6] P. Kroeger, *Analyzing Grammar*, 2005, p. 248–250.

[7] M. Kosmulski, "Reprezentacja dokumentów tekstowych w modelu przestrzeni wektorowej," 2005.

[8] D. Eastlake. (2006) Domain name system (dns) case insensitivity clarification. IETF. [Online]. Available: https://tools.ietf.org/html/rfc4343

[9] D. M. W. Powers, "Applications and explanations of zipf's law," in *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*, ser. NeMLaP3/CoNLL '98. Stroudsburg, PA, USA: Association for Computational Linguistics, 1998, pp. 151–160. [Online]. Available: http://dl.acm.org/citation.cfm?id=1603899.1603924

[10] P. S. Foundation. Shelve — python object persistence. [Online]. Available: https://docs.python.org/3.4/library/shelve.html

[11] M. Bańko. Ile jest słów w języku polskim. [Online]. Available: http://sjp.pwn.pl/poradnia/haslo/;2384