AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI

Praca dyplomowa inżynierska

Kontekstowa konsolidacja i agregacja informacji ze stron internetowych z wykorzystaniem asocjacyjnych struktur AGDS w celu optymalizacji ich przetwarzania.

Autor: *Szymon Sobański*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr inż. Adrian Horzyk*

Kraków, 2014

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wprowadzenie

Wraz z rozpowszechnieniem dostępu do Internetu, można zauważyć zwiększenie ilości produkowanych informacji. Jak wynika z przygotowanego przez organizację IDC raportu, między latami 2009, a 2020, przewiduje się 44 krotny wzrost ilości danych dostępnych w sieci [4]. Jednym z najbardziej odczuwalnych skutków wzrostu ilości i dostępności danych, dotykający również przedsiębiorstwa, jest tzw. *Information Overload*, czyli nadmiar czynników branych pod uwagę w różnych procesach decyzyjnych. Wpływa to na obniżenie dynamiki, innowacyjności i konkurencyjności na rynku i ma wyraźny negatywny wpływ na gospodarkę [13].

Z drugiej strony, łatwo dostępne i zróżnicowane dane mogą służyć jako podstawa rozwoju, dając przede wszystkim dostęp do ogromnej bazy wiedzy zgromadzonej w sieci, jak i umożliwiając przedsiębiorcom zdobycie wiedzy o kosnumentach, łatwiejsze badanie rynku i dostosowanie oferowanych usług do potrzeb klientów [6, s. 1-26].

Głównym problemem zatem pozostaje kwestia sposobu porządkowania i filtracji dostępnych danych, w celu ich wykorzystania. Klasycznym przykładem aplikacji służacej do tego celu jest *wyszukiwarka internetowa* - witryna dostępna w sieci, która umożliwia przeszukiwanie Internetu pod kątem wprowadzanych przez użytkownika zapytań.

1.1. Cele pracy

Celem poniższej pracy jest zaproponowanie rozwiązania porządkującego dane pochodzące z sieci internetowej i przechowującego je w postaci grafu asocjacyjnego - AGDS [7, s. 112-117].

Poruszone zostają kwestie budowy oprogramowania eksplorującego sieć i pobierającego treść stron, przechowywania tak zdobytych informacji i przetwarzania ich w celu umieszczenia w grafie. Przedstawione zostają argumenty przemawiające za takim rozwiązaniem, poddana dyskusji zostaje różnica pomiędzy proponowanym podejściem, a rozwiązaniami wiodącymi obecnie prym w zastosowaniach komercyjnych.

Ponadto zaproponowano wykorzystanie tak przechowywanych danych do budowy prostej asocjacyjnej wyszukiwarki internetowej, opartej o neuroasocjacyjne grafy wiedzy ANAKG[7, s. 223-244].

1.2. Zawartość pracy

W rodziale 2 przedstawiono podstawowe informacje dotyczące pozyskiwania danych z witryn internetowych. Zilustrowane są podstawowe wymagania stawiane przed oprogramowaniem trawersującym sieć, tzw. *crawlerem*, poruszone są kwestie wydajności, poprawnej implementacji, oraz opisane są względy etyczne, którymi należy się kierować przy korzystaniu z takiego oprogramowania.

Rozdział 3 ma na celu opisanie problemów dotyczących ekstrakcji danych z dokumentu HTML. Zawiera on krótki opis formatu, przestawia jego zastowania i wyjaśnia, dlaczego poprawne uzyskiwanie informacji ze źródeł stron internetowych nie jest zadaniem trywialnym. Ponadto, zilustrowane są różne podejścia do parsowania zawartości dokumentów HTML, jak i przedstawiona jest pokrótce nomenklatura różnych typów stron, w kontekście wyszukiwarek internetowych.

Rozdział 4 prezentuje podstawowe informacje o asocjacyjnych grafach AGDS. Wyjaśnia powód zainteresowania takimi strukturami, wskazuje na ich zalety, przedstawia sposób tworzenia struktur i przytacza przykładowe grafy stworzone za pomocą oprogramowania będącego częścią projektu. Stara się również przedstawić konsekwencje sposobów budowy grafu i możliwość uzyskiwania informacji, zawartych w takich strukturach.

Następnie, w rozdziale 5, opisane są szczegóły implementacyjne aplikacji realizującej cele pracy. Wyjaśnione są szczegóły architektury, przytoczone zalety wybranch technologii, poruszony jest temat wydajności oprogramowania. Przedstawione jest również działanie całej aplikacji, wraz z wyjaśnieniem założeń testowych.

Rozdział 6 przedstawia aplikację wykorzystującą graf AGDS i zbudowaną w oparciu o niego sieć neuronową ANAKG do wyszukiwania stron internetowych. Przedstawia on podstawowe założenia stojące za budową sieci ANAKG, algorytm jej pobudzania, jak i sposób interpretacji wyników. Podjęta jest również próba ilustracji odpowiedzi sieci dla różnych parametrów symulacji.

2. Pozyskiwanie treści z Internetu

Pozyskiwaniem treści z sieci zajmuje się grupa oprogramowania nazywana sieciowymi pająkami(ang. *web crawlers*, *web spiders*) lub sieciowymi robotami(ang. *web robots*). Danymi wejściowymi jest zazwyczaj grupa stron, a właściwie adresów URL, nazywana **seedem** [2]. Ich zadanie zazwyczaj nie ogranicza się do pobrania jednej strony, ale do trawersowania całej sieci, bądź w celu pozyskania konkretnych informacji, bądź w celu jej eksploracji. Przy czym w związku z ilością danych dostępnych w Internecie zazwyczaj zakłada się pewne filtry i ograniczenia, w celu zwiększenia efektywności robota i zwiększenia szans na odwiedziny najbardziej wartościowych, dla danego zastosowania, stron.

W związku z dynamiczną naturą Internetu, koniecznym jest ciągłe odświeżanie posiadanych już informacji, poszukiwanie nowych stron i ew. eliminacja witryn, które juz nie funkcjonują. Wpływa to na charakterystykę działania pająków internetowych i oprogramowania z nimi współpracującego, które musi być w stanie stale przetwarzać dużą ilość informacji w krótkim czasie. W związku z ogromną ilością danych wymagana jest również pełna automatyzacja działania i samodzielne podejmowanie decyzji o odwiedzanych w danym czasie witrynach.

Zadanie, które wykonują roboty internetowe jest tylko z pozoru proste. W istocie zawiera w sobie wiele innych, jak utrzymywanie połączeń sieciowych i obsługa częstych błedów, unikanie "pułapek na pająkiźwiązanych z cyklami w strukturze sieci, czy przestrzeganie względów etycznych. Nie dziwi zatem pogląd założycieli firmy Google, którzy w jednym z artykułów stwierdzili, iż robot sieciowy jest najbardziej wyszukanym i najwrażliwszym komponentem wyszukiwarki internetowej[11].

2.1. Zasosowania robotów internetowych

Roboty internetowe można podzielić, wg wykonywanego zadania i zastosowań. Wszystkie kategorie charakteryzują się podobnym algorytmem pobierania stron, główne różnice wynikają z wyboru listy adresów do odwiedzenia, jak i ze sposobów priorytetyzacji pobierania informacji z posiadanej kolekcji URLi.

2.1.1. Przeszukiwanie uniwersalne

Roboty **uniwersalne** (ang. *unviersal*) [8, s. 311-315] mają za zadanie przeszukiwanie sieci, w celach eksploracyjnych i zazwyczaj służą wyszukiwarkom internetowym(Google, Bing, DuckDuckGo itd.).

Roboty **przeszukujące wszerz** mają za zadani zebranie informacji o jak największej ilości dokumentów, reprezentujących zasoby całej dostępnej sieci. Ich nazwa bierze się z analogii zadań przeszukiwania sieci i przeszukiwania grafu. Zazwyczaj stawiany jest też wymóg wysokiej jakości odwiedzanych stron, co może stać w sprzeczności do wymogu szeroko zakrojonej penetracji Internetu [2].

Roboty **przeszukujące wgłab** [2] starają się odwiedzać strony odpowiadające w określony sposób potrzebom aplikacji. Sposób doboru stron może różnić się w zależności od zastosowań: od prostych reguł dotyczących podzbioru adresów URL, domen, rozszerzeń plików, czy używanego języka.

2.1.2. Przeszukiwanie skupione

Kolejny rodzaj robotów przeszukujących sieć stanowią roboty **skupione** (ang, *focused*) [12]. Ich zadaniem jest zbieranie informacji o stronach należących do danej, z góry określonej, dziedziny. W tym celu wykorzystuje się często algorytmy uczenia maszynowego i sztucznej inteligencji[8, s. 327-330], które na podstawie zapytania lub znanego podzbioru dokumentów należących do interesującej użytkownika dziedziny, określają przydatność dokumentów pobieranych przez robota. Przeszukiwanie w ten sposób może odbywać się albo trybie *batchowym*, polegającym na pobieraniu i ocenianiu wielu stron dla zadanych z góry kategorii, niezależnych bezpośrednio od użytkownika, albo w trybie *online*, w odpowiedzi na konkretne zapytanie.

2.1.3. Eksploracja struktury sieci

Crawlery używane są również do badania struktury Internetu i sposobu w jaki zmienia się on w czasie. Ten typ zastosowań jest szczególnie wrażliwy na sposób wybierania kolejnych stron i zbiór punktów startowych. Zostało pokazane [9], iż niewłaściwie dobrane adresy stron startowych prowadzą do obrazu sieci niezgodnego ze stanem faktycznym.

2.1.4. Mirroring

Mirroring jest to sporzadzanie kopii istniejących stron internetowych, zwykle w celu poprawy dostępności treści danej witryny. Jest to prosta operacja dotycząca ograniczonego zbioru adresów, dlatego też nie wymaga zaawansowanych algorytmów stosowanych w innych robotach.

2.1.5. Analiza stron internetowych

To zastosowanie związane jest z administracją dużymi serwisami. Robot odwiedza zestam adresów w celu zbadania poprawności działania strony internetowej, wykrywania możliwych usterek lub analizy pod kątem bezpieczeństwa. Strony takie jak Wikipedia używają wyspecjalizowanego oprogramowania w celu automatyzacji wielu zadań, takich jak zbieranie informacji o podstronach, do których nie prowadzi żaden link w serwisie[2].

2.2. Działanie pająka internetowego

Diagram blokowy 2.1 przedstawia sekwencyjny algorytm działania pająka internetowego. Jest on bardzo prosty i nie nadaje się do zastosowań produkcyjnych, jednak ilustruje podstawowe koncepty charakterystyczne dla tego typu zagadnień. Głównym zarzutem wobec tej struktury jest brak współbieżności, każda strona jest pobierana i przetwarzana sekwencyjnie. Jak wynika z raportu Google mediana czasu ładowania pojedynczej strony internetowej wynosi powyżej 2 s[1], można się zatem spodzieać, iż będzie to operacja najdłużej trwająca spośród przedstawionych na schemacie.

2.2.1. Algorytm

Przy inicjalizacji robota na wejście podawany jest zestaw adresów URL, stanowiących tzw *seed*. Używane są one do inicjalizacji bierzącej unikalnej listy adresów URL czekających na odwiedzenie (ang. *frontier*). W kazdej iteracji crawler zdejmuje pierwszy adres z listy i pobiera stronę znadjującą się pod tym adresem. Następnie następuje ekstrakcja adresów URL znajdujących sie na stronie, normalizacja ich i zapis do listy adresów czekających na odwiedzenie. Finalnie, źródło strony zapisywane jest w lokalnym systemie plików i crawler jest gotowy do pobrania kolejnej witryny.

Zazwyczaj algorytm kończy działanie po pobraniu ustalonej z góry ilości stron. Inną możliwością zakończenia działania jest opróżnienie listy oczekujących adresów URL, ale jest to mało prawdopodne, ze względu na dużą ilość linków - średnio jest to dziesięć adresów URL na stronę[8, s.312].

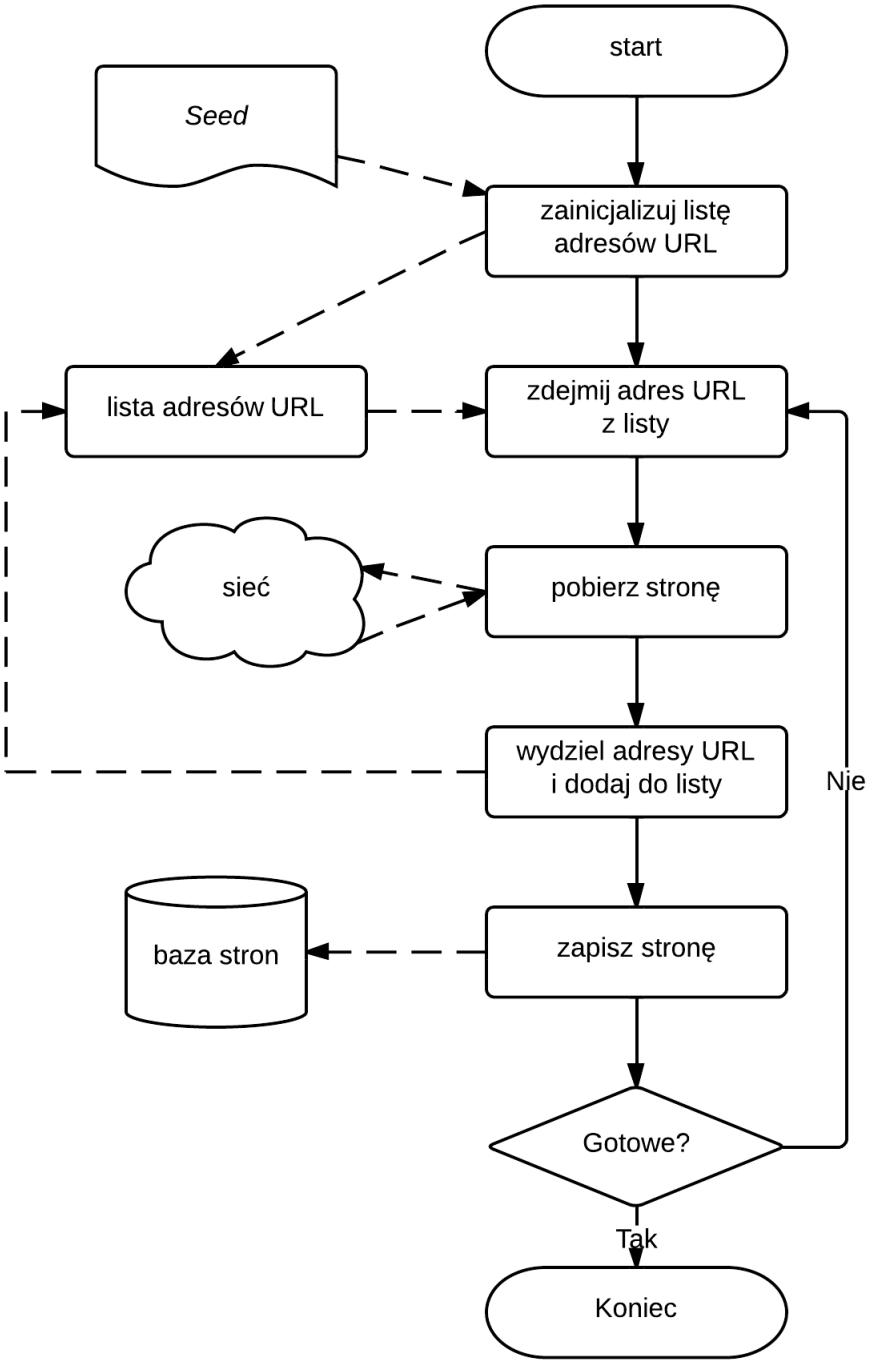
Dla robotów **przeszukujących wszerz** lista adresów URL implementowana jest zazwyczaj jako kolejka FIFO. Natomiast dla robotów **przeszukujące wgłąb** i **skupionych** używana jest kolejka priorytetowa, z priorytetem przypisanym na podstawie oceny przydatności danego adresu.

2.2.2. Uwagi implementacyjne

W związku z ilością scenariuszy możliwych do napotkania przy trawersowaniu sieci, jak i dużą ilością danych, która program przetwarza, stosowane są dodatkowe zabiegi, zwiększające szanse poprawnego działania aplikacji.

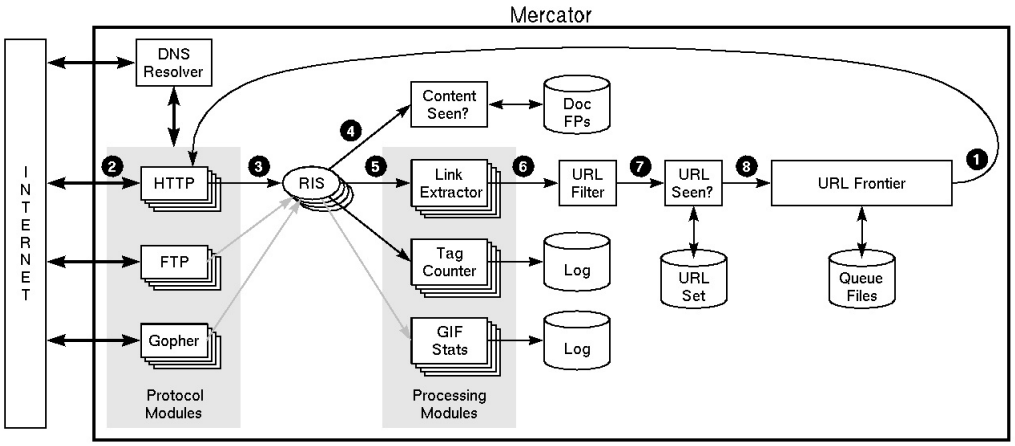
1. Stosowanie ścisłej kontroli nad ściąganiem stron: ograniczenie czasu trwania połączenia(stosowanie *timeoutów*), określenie górnej granicy ilości danych pobieranych z jednej strony(np. do 50 KB), wykrywanie nieskończonych pętli przekierowań, logowanie błędów(przekroczenie czasu połączenia, błędne kody odpowiedzi itd.).

2. Preprocessing pobranych źródeł, mający na celu eliminację częstych błędów występujących na stronach, takich jak np. niedomknięte tagi. Wykrycie kodowania i próba jego normalizacji - np. kodowanie wszystkich pobranych źródeł po stronie crawlera w formacie UTF-8. Na tym etapie można przeprowadzić również usuwanie słów nie niosących ze sobą istotnych treści(tzw. *stopwords*), takich jak np. wyrazy “a”, “aby”, “ach”, “aczkolwiek”, ... itd. Stosowane są również



Rysunek 2.1: Schemat blokowy algorytmu robota internetowego.

inne formy przetwarzania danych, takie jak *lematyzacja* sprowadzenie do podstawowej formy gramatycznej i *stemming* utworzenie tzw. rdzenia słowa, części wspólnej słów niosących podobną informację [10].



Rysunek 2.2: Schemat crawlera Mercator, implementującego m.in. cache DNS.

3. Doprowadzenie linków do postaci kanonicznej. Przede wszystkim należy zapewnić przedstawienie wszystkich URLi w postaci aboslutnej. Ponadto stosowane są heurystyki związane np. z usuwaniem numeru portu, o ile figuruje w danym adresie, pozbywaniem się nieznaczących sufiksów oznaczających typ dokumentu(np. .html), czy usuwanie tzw. fragmentu - części adresu następującej po znaku "#".

4. Unikanie tzw. pułapek na pająki (ang. *spide-traps*). *Pułapka na pająki* to strona, która generuje dynamicznie nieskończenie wiele adresów URL prowadzących do niej samej [5]. Przykładem takiej witryny jest aplikacja kalendarza, który generuje linki do stron przedstawiających poprzednie lub następne przedziały czasu. Aby uniknąć nieskończonej pętli stosuje się różnego rodzaju heurystyki, np. określa się górny limit ilości pobrań z danej domeny. Ma to jendak negatywny wpływ na aktualność informacji pobieranych ze stron o skończonej liczbe odnośników. Wpadnięcie wydajnego crawlera w taką pułapkę powoduje znaczne obciążenie serwerów i może być nawet interpretowane przez administratorów strony - pułapki jako atak denial of service [8, s. 322].

5. Wprowadzenie wielu wątków/procesów. Jak wspominano wcześniej, czas sekwencyjnego pobierania stron zależy w dużym stopniu od szybkości pobierania danych. Również, przy dużej ilości informacji, kosztowne może stać się również zapisaywanie danych na dysku. Jednym ze sposobów skrócenia czasu uzyskiwania nowych danych jest zastosowanie wielu równoległych wątków lub procesów, pobierających i przetwarzających dane niezależnie od siebie. Następuje wtedy jedynie konieczność synchronizacji dostępu i modyfikacji zarówno listy oczekujących do pobrania adresów URL. Takie podejście może w prosty sposób przyspieszyć działanie oprogramowania od 5 do 10 krotnie [8, s. 323].

   Innym ulepszeniem, które może zostać wprowadzone do wielowątkowej architektury, jest pobieranie danych w sposób asynchroniczny. Pozwala to na lepsze wykorzystanie dostępnej przepustowości sieci, niż w przypadku połączeń synchronicznych. Aby ograniczyć czas spędzony na translacji

adresów URL na adresy IP wprowadza się specjalny serwis wykonujący ją zawczasu dla adresów URL oczekujących w kolejce i cache'ujący rezultaty.

2.3. Ocena zbioru pobranych stron

Oprócz efektywnego sposobu pobierania informacji z sieci, ważne jest również zrozumienie cech posiadanego zbioru stron internetowych i jego ewaluacja. W tym przypadku dąży się zarówno do minimalizacji nieaktualnych informacji o witrynach internetowych, jak i do maksymalizacji ilości informacji o stronach w ogóle. Szczególnie ważne jest szybkie reagowanie na zmiany(dodanawanie nowych dokumentów, edycję i usuwanie istniejących), które w wypadku sieci postępują nadzwyczaj gwałtownie.

2.3.1. Świeżość i wiek

Szczególnie ważnym zagadnieniem jest **świeżość** (ang. *freshness*) [8, 2]. Miara ta jest określona dla pobranej strony *p* w chwili *t* jako

$$F_p(t) = \begin{cases} 1 & \text{jeżeli } p \text{ jest takie samo, jak lokalna kopia} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{2.1}$$

Dla tej samej pary można określić również **wiek** (ang. *age*) [2], dany zależnością:

$$A_p(t) = \begin{cases} 0 & \text{jeżeli } p \text{ nie jest modyfikowane w chwili } t \\ t - \text{czas modyfikacji } p & \text{w przeciwnym wypadku} \end{cases} \tag{2.2}$$

W zależności od podjętych założeń stosuje się różne algorytmy selekcji dokumentów do ponownego odwiedzenia.

2.3.2. Algorytmy odświeżania zbioru dokumentów

Pierwszy z przedstawionych algorytmów dąży do zminiejszenia liczby stron, które moga być nieaktualne. W tym modelu nie ma znaczenia historia historia zmian zapisanej strony, wszystkie dokumenty odświeżane są z taką samą częstotliwością. Wiąże się to z utrzymywaniem dobrej średniej **świeżości**, [3] kosztem dopuszczenia możliwości posiadania nieaktualnych wersji pewnych nadzwyczaj często zmieniających się stron.

Można również stosować inne podejście, które głosi, iż częstotliwość pobierania strony powinna być proporcjonalna do częstotliwości jej zmian w czasie. Wprawdzie takie podejście sprawdza się dla witryn edytowanych regularnie, jednak nadużywane prowadzi do zbyt częstych, nieprzynoszących nowych informacji, odwiedzin adresów historycznie zmieniających się z dużą częstotliwością. [3].

Mimo, iż pierwsze podejście jest bliższe optymalnemu, najlepsze rezultaty daje określenie rozkładu prawdopodobieństwa zmian dla poszczególnych witryn i dostosowywanie częstotliwości pobierania informacji z tych źródeł do otrzymanych rezultatów [2].

2.4. Charakterystyka zmian sieci w czase

Poniżej przytoczona zostaje zbiorcza charakterystyka zmian w różnych podzbiorach sieci, uzyskana z [2].

Tablica 2.1: Zbiorcze dane dotyczące różnych badań zmian zachodzących w sieci.

Podzbiór	Obserwacje
360 losowych stron	mediana życia strony około 2.5 roku, 33% było wciąż dostępnych po 6 latach
500 stron (arykułów naukowych on-line)	mediana życia strony około 4.5 roku
2 500 stron	średnia długość życia około 50 dni mediana wieku około 50 dni
4 200 stron (artykułów naukowych on-line)	mediana życia strony około 4 lat
720 000 stron	średnia długość życia między 60, a 240 dni. 40% stron w domenie `.com` zmienia się codziennie 50% stron w domenach `.gov` i `.edu` pozostaje bez zmian przez 4 miesiące
950 000 stron	średni wiek między 4 dniami, a 4 miesiącami. Strony o dużej iości linków wskazujących na nie zmieniane częściej.
4 miliony stron	8% tygodniowy przyrost nowych stron 62% stron tygodniowo dodaje nową treść 25% tygodniowy przyrost nowych linków 80% zmian jest niewielka
150 milionów stron	w 10 tygodniowym okresie 65% stron pozostaje bez zmian na 30% stron zachodzą jedynie niewielkie zmiany duża wariacja dostępności, w zależności od domeny
800 milionów stron	średnia długość życia około 140 dni

Jak widać dynamika zmian w sieci zależy w dużym stopniu od indywidualnych witryn, ich miejsca w grafie oraz celu, któremu służą. Praktycznie uniemożliwia to skonstruowanie ogólnego algorytmu odświeżającego posiadaną kolekcję danych w sposób optymalny.

2.5. Kwestie etyczne

3. Przetwarzanie danych pochodzących z witryn internetowych

4. Omówienie asocjacyjnych grafów AGDS

5. Implementacja

6. Asocjacyjna wyszukiwarka internetowa

7. Podsumowanie

Bibliografia

[1] http://analytics.blogspot.in/2012/04/global-site-speed-overview-how-fast-a html.

[2] Carlos Castillo and Ricardo Baeza-Yates. *Web Crawling*. Universitat Pompeu Fabra. http://grupoweb.upf.es/WRG/course/slides/crawling.pdf.

[3] Junghoo Cho and Hector Garcia Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4), 2008.

[4] John Gantz and David Reinsel. The digital universe decade – are you ready? http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf, 2010.

[5] Padmini Srinivasan Gautam Pant and Filippo Menczer. Crawling the Web. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.4776&rep=rep1&type=pdf.

[6] Michael J.A. Berry Gordon S.Linoff. *Data Mining Techniques*. Wiley, third edition, 2011.

[7] Adrian Horzyk. *Sztuczne systemy skojarzeniowe i asocjacyjna sztuczna inteligencja*. Akademicka Oficyna Wudawnicza Exit, Warszawa, 2013.

[8] Bing Liu. *Web Data Mining*. Springer, second edition, 2011.

[9] Marian Boguna Santo Fortunato Alessandro Vespignani M. Angeles Serrano, Ana Maguitman. Decoding the structure of the www: facts versus sampling biases. *ACM Transactions on the Web*, 1(10), 2007.

[10] Dariusz Mrozek. Wyszukiwanie pełnotekstowe. http://zti.polsl.pl/bdusm/BD3_FTS.pdf.

[11] Brin S. and P. Lawrence. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[12] Martin van den Berg Soumen Chakrabarti and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

[13] Jonathan B. Spira. Information Overload: Now 900 Billion – What is Your Organization's Exposure? `http://bit.ly/1coHvYF`, 2008.