Akademia Górniczo-Hutnicza

im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

PRACA MAGISTERSKA

ANDRZEJ JASIŃSKI

SPECJALISTYCZNA INTERNETOWA WYSZUKIWARKA OGŁOSZEŃ PRZETARGOWYCH

PROMOTOR:
dr Adrian Horzyk

Kraków 2009

OŚWIADCZENIE AUTORA PRACY

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

.....

PODPIS

AGH

University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

MASTER OF SCIENCE THESIS

ANDRZEJ JASIŃSKI

SPECIALIZED TENDER SEARCH ENGINE

SUPERVISOR:

Adrian Horzyk Ph.D

Krakow 2009

Serdecznie dziękuję promotorowi za udzielanie cennych rad i wskazówek podczas pisania tej pracy.

Spis treści

1. Wprowadzenie

Problem kategoryzacji oraz wydobywania informacji z dokumentów jest zgłębiany przez wiele firm, instytucji oraz ośrodków naukowych. Jest to spowodowane bardzo szerokim wachlarzem zastosowań, od gromadzenia i indeksowania danych, tworzenie wyszukiwarek po data mining, filtry antyspamowe i wiele innych. Szczególnym zastosowaniem jest wyszukiwanie informacji ze specyficznej dziedziny. Takim właśnie zadaniem jest odnajdywanie informacji o ogłoszeniach przetargowych. Pomimo istnienia portali gromadzących dane o zamówieniach przetargowych, ciekawe jest zbadanie możliwości stworzenia automatycznego systemu zbierania informacji o przetargach oraz zamówieniach publicznych. Jest to duże wyzwanie, obejmujące problemy analizy tekstu, niezawodności oprogramowania oraz gromadzenia dużych ilości informacji. Wszystkie te zagadnienia są interesujące, stąd wybór takiego tematu pracy magisterskiej.

1.1. Cele pracy

Celem poniższej pracy jest zaprojektowanie i wykonanie systemu do znajdywania w internecie ogłoszeń przetargowych oraz ich gromadzenia i indeksowania. System ma kategoryzować odnalezione ogłoszenia, wyszukiwać na nich najważniejsze dane. System gromadzący informacje powinien poprawnie rozpoznawać ogłoszenie przetargowe, niezależnie od jego struktury oraz typu dokumentu, czy jego budowy. Powinien uwzględniać strony, na których znajduje się wiele ogłoszeń i przeglądać je częściej, aby zwiększyć ilość zgromadzonych przetargów. Użytkownik musi posiadać możliwość przeglądania bazy danych, w poszukiwaniu interesujących słów kluczowych oraz zdefiniowanych branż. Dodatkowo, żąda się możliwości tworzenia raportów. Pod pojęciem tym rozumie się okresowe sprawdzanie bazy danych pod kątem określonych przez użytkownika kryteriów (takich samych jak podczas zwykłego wyszukiwania ogłoszeń).

1.2. Zawartość pracy

W rodziale 2 przedstawiono istniejące już rozwiązania, z opisem ich możliwości, zalet, wad oraz ograniczeń. W rozdziale 3 znajduje się wstęp teoretyczny, zawierający informacje o stosowanych w innych systemach oraz użytych w tej pracy algorytmach. Rozdział 4 opisuje ogólny projekt systemu oraz w zwięzły sposób przedstawia poszczególne etapy przetwarzania dokumentów - od pobrania ich z Internetu, po zapisanie do bazy. W części 5 pokazano realizację systemu oraz jego działanie na podstawie interfejsu aplikacji internetowej. Znajdują się w nim wyniki działania systemu oraz porównanie do istniejących rozwiązań. Rodział 6 stanowi podsumowanie pracy.

2. Istniejące rozwiązania

W rozdziale tym przedstawiono istniejące na rynku rozwiązania. Nie udało się znaleźć podczas pracy badawczej systemu działającego na tej samej zasadzie. Istnieją natomiast serwisy, które posiadają zbliżoną funkcjonalność:

1. http://www.przetargi.pl/ (serwis płatny).
2. http://www.przetargi.info/ (serwis płatny).
3. http://msp.money.pl/przetargi/ (częściowo płatny).
4. http://www.egospodarka.pl/ (bezpłatny).

Z oczywistych względów trudno jest dokładnie podać algorytm działania wyżej wymienionych stron, ale z informacji na nich dostępnych ([8], [9]) wynika, że serwisy te przeglądają tylko określone strony (instytucji publicznych, Dziennik Urzędowy Unii Europejskiej, serwisy informacyjne). Wszystkie wyżej wymienione serwisy umożliwiają odnajdywanie ogłoszeń według wielu kryteriów (słowa kluczowe, branże, lokalizacja, data ogłoszenia). Posiadają bardzo obszerne bazy liczone w setkach tysięcy ogłoszeń [8]. Wadą jest konieczność uiszczenia opłaty za korzystanie z dostępnych baz. W poniższych sekcjach pokrótce opisane są wymienione serwisy. Pierwsze dwa są całkowicie płatne, wyszukiwarka przetargów money.pl wyświetla tylko podstawowe informacje o ofercie, za szczegółowe (w tym podmiot ogłaszający) trzeba zapłacić. Darmowa jest wyszukiwarka egospodarka.pl.

2.1. przetargi.pl

Autorzy serwisu deklarują posiadanie ok 2 milionów ogłoszeń, w tym ofert kupna, sprzedaży, przetargów [9]. Wyszukiwarka ta jest płatna i wymaga rejestracji, aby przeglądać zgromadzone zasoby.

Serwis nie oferuje niestety dostępu testowego, możliwe jest jedynie przeglądanie podstawowych informacji. Wyszukiwarka pozwala na bardzo szczegółowe określenie interesujących przetargów. Można dokonywać zawężania wyników według:

- numeru ogłoszenia,

Rysunek 2.1: Podstawowa forma wyszukiwania oraz prezentacja wyników.

- kategorii (przetarg, inwestycja, oferta handlowa, kupno, sprzedaż),
- zakresie terytorialnym (Polska, UE, inne),
- słów kluczowych,
- branży (maksymalnie czterech, w każdej pięć podbranży),
- mieście,
- organizatorze,
- wadium i wartości przetargu,
- terminie składania ofert,
- źródle informacji,
- kodzie CPV.

Jak widać, możliwości wyszukiwania są dość spore. Niestety, nie udało się odnaleźć żadnego ogłoszenia dotyczącego przetargu/oferty na maszyny vendingowe.

2.2. przetargi.info

Jest to bardzo przyjazny serwis, z dużą ilością ogłoszeń (ponad 130 tysięcy). Poza standardowym wyszukiwaniem ofert umożliwia on generowanie raportów, które są następnie dostarczane do użytkownika jako wiadomości e-mail. Wyszukiwarka ta jest płatna i wymaga rejestracji, aby przeglądać zgromadzone zasoby. Podczas rejestracji użytkownik ma możliwość stworzenia profilu swojej działalności, w celu określenia przetargów, które mogą go potencjalnie interesować (rys. 2.3). Dzięki temu, na skrzynkę pocztową, podaną podczas rejestracji, będą wysyłane nowe ogłoszenia z podanej branży.

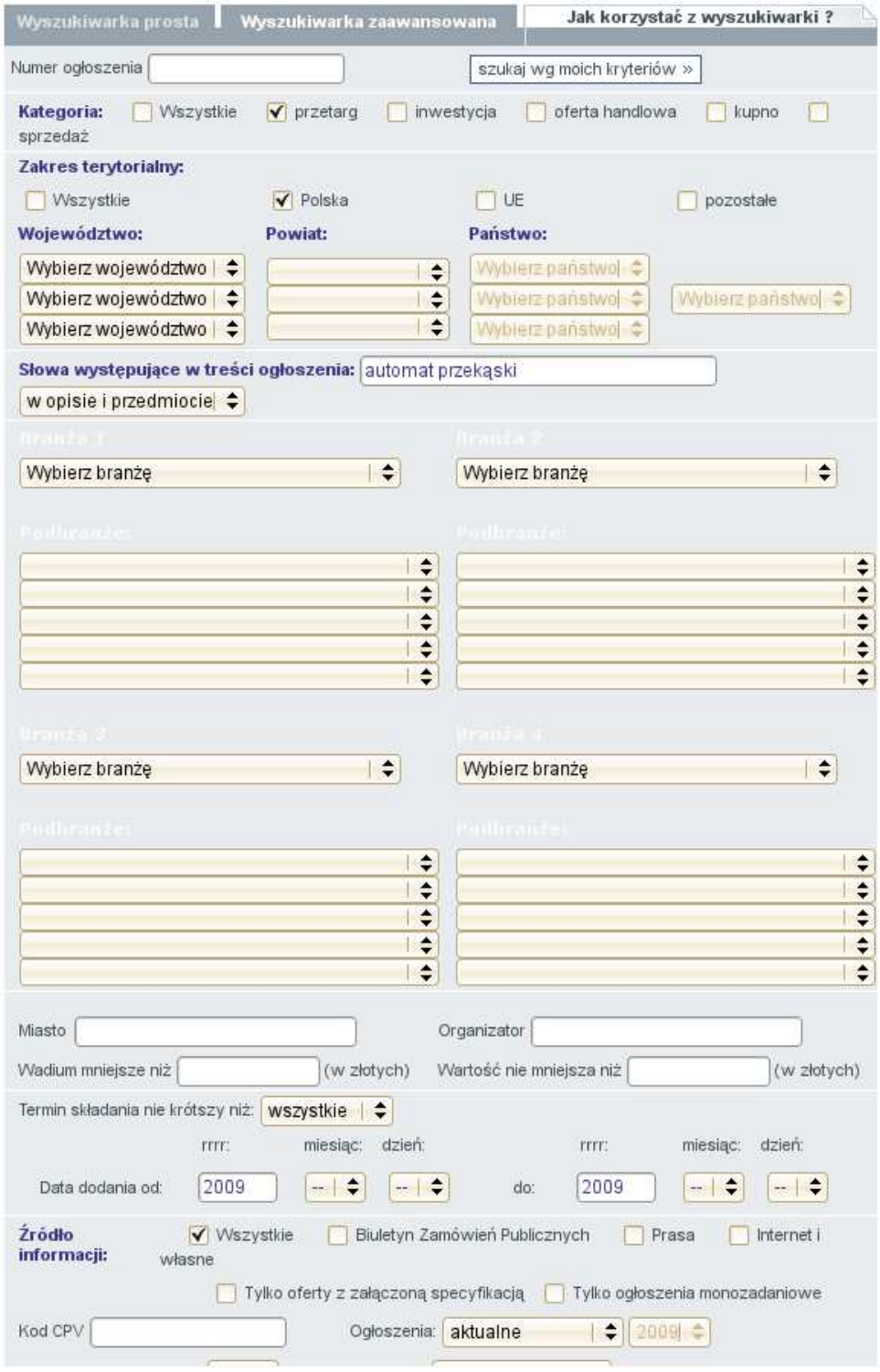
Lista branż jest bardzo duża, choć nie pokrywa wszystkich możliwych profili (nie znaleziono niczego zbliżonego do “vending”).

Po określeniu swojego profilu działalności użytkownik otrzymuje listę ogłoszeń pasujących do jego branży (przykładowa treść e-maila znajduje na rysunku 2.4).

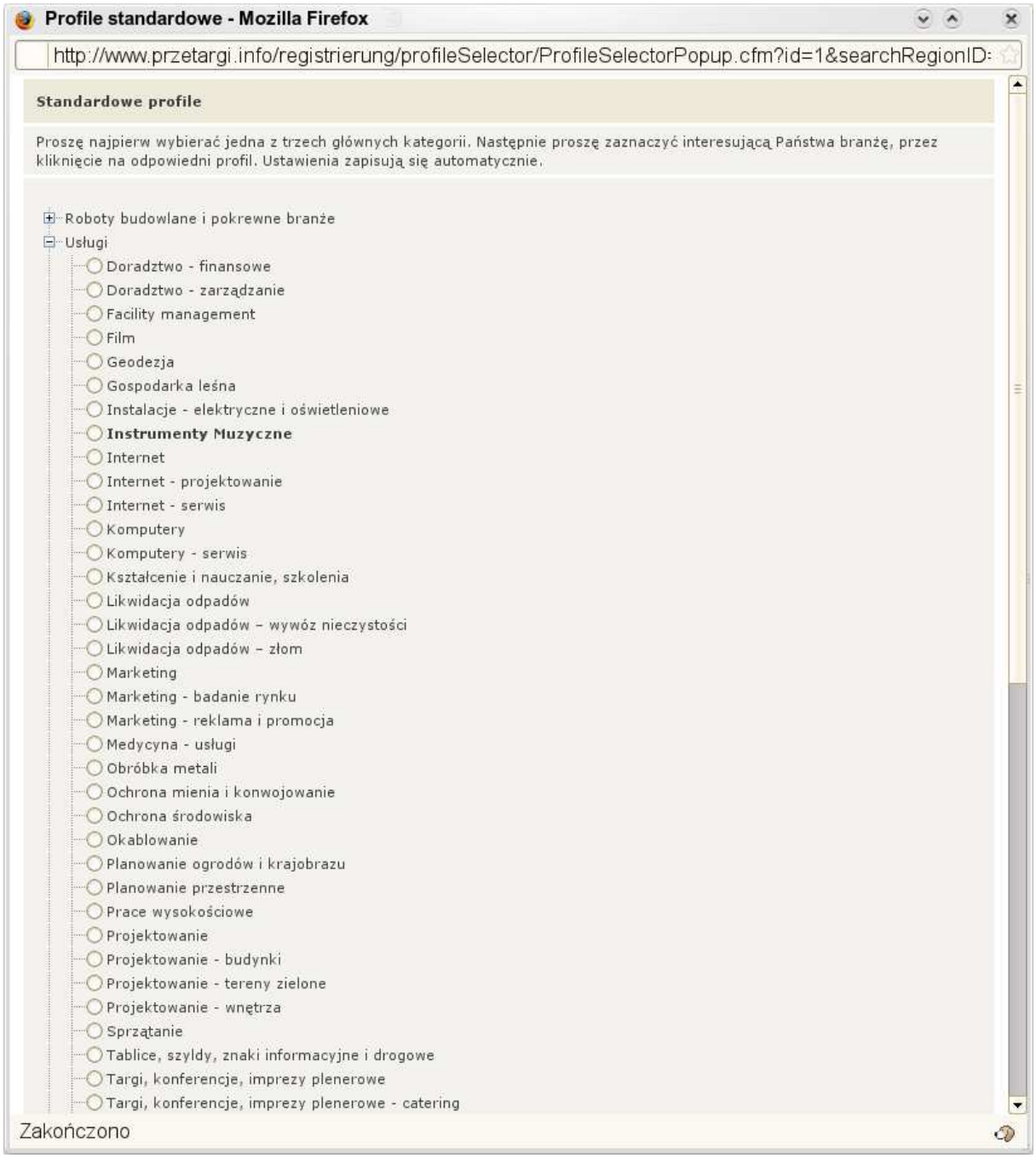
Ponadto istnieje możliwość przeszukiwania bazy danych serwisu (rys. 2.5).

Serwis przetargi.info oferuje najbogatsze możliwości wyszukiwania ofert. Do dyspozycji są filtry:

- region (państwo),
- tytuł,



Rysunek 2.2: Opcje zawężania rezultatów wyszukiwania systemu przetargi.pl.



Rysunek 2.3: Określanie profilu działalności w portalu przetargi.info.

przetargi.info
Dostarczamy zamówienia dla Ciebie!

Strona główna | FAQ | O nas

Numer Użytkownika: 175962

Szanowny Pan ,

Twój profil wyszukiwania "Urządzenia biurowe" w naszej bazie danych odpowiada aktualnie **727** zawiadomieniom o przetargach (aktualne oferty jak również wyniki). Od jutra będziesz otrzymywać przetargi e-mailem codziennie, jak tylko zostaną one opublikowane.

Prosimy o sprawdzenie czy poniższe przetargi odpowiadają Twojej branży. **Możliwe, że Twój profil nie został jeszcze dostosowany do Twoich potrzeb**. Jeśli przesłane przetargi nie spełniają Twoich oczekiwań, bardzo chętnie pomożemy Ci w optymalizacji Twojego profilu wyszukiwania.
Zadzwoń do naszych konsultantów **+48-12-4261470 lub +48-12-4261471** w godzinach: **Poniedziałek – Piątek 8:00 – 16:30**. Służymy radą i fachową pomocą.

Aktualne przetargi:

Meble

Typ zamówienia:	Dostawy	**Data składania ofert:**	05.10.2009
Region:	Polska	**I.D.:**	4773154
Typ dokumentu:	Przetarg (wszystkie procedury)	**Zamawiający:**	Zarząd Obsługi Ministerstwa Spraw Zagranicznych

Maszyny i aparatura badawcza i pomiarowa

Typ zamówienia:	Dostawy	**Data składania ofert:**	01.10.2009
Miejsce realizacji:	Lubelski		
Region:	Polska	**I.D.:**	4773112
Typ dokumentu:	Przetarg (wszystkie procedury)	**Zamawiający:**	Katolicki Uniwersytet Lubelski Jana Pawła II

Jednorazowe, niechemiczne artykuły medyczne i hematologiczne

Typ zamówienia:	Dostawy	**Data składania ofert:**	30.09.2009
Miejsce realizacji:	Lubuskie		
Region:	Polska	**I.D.:**	4761505
Typ dokumentu:	Przetarg (wszystkie procedury)	**Zamawiający:**	Samodzielny Publiczny Szpital Wojewódzki w Gorzowie Wlkp.

Rysunek 2.4: Przykładowa wiadomość zawierająca ogłoszenia pasujące do określonej branży.

Rysunek 2.5: Wyszukiwarka ogłoszeń portalu przetargi.info.

- opis ogłoszenia,
- treść ogłoszenia,
- data ogłoszenia,
- kod CPV,
- zamawiający,
- rodzaj zamówienia(roboty budowlane, usługi, dostawy, zamówienia mieszane),
- województwa.

Dla filtrów tytułu, opisu i treści istnieje możliwość podania słów obowiązkowych, wykluczonych oraz poszukiwanych fraz.

2.3. egospodarka

Serwis egospodarka.pl udostępnia darmową wyszukiwarkę ogłoszeń przetargowych. Można wyszukiwać ogłoszenia według:

- słów kluczowych,
- miast,
- branży,
- daty składania ofert,
- typu ogłoszenia.

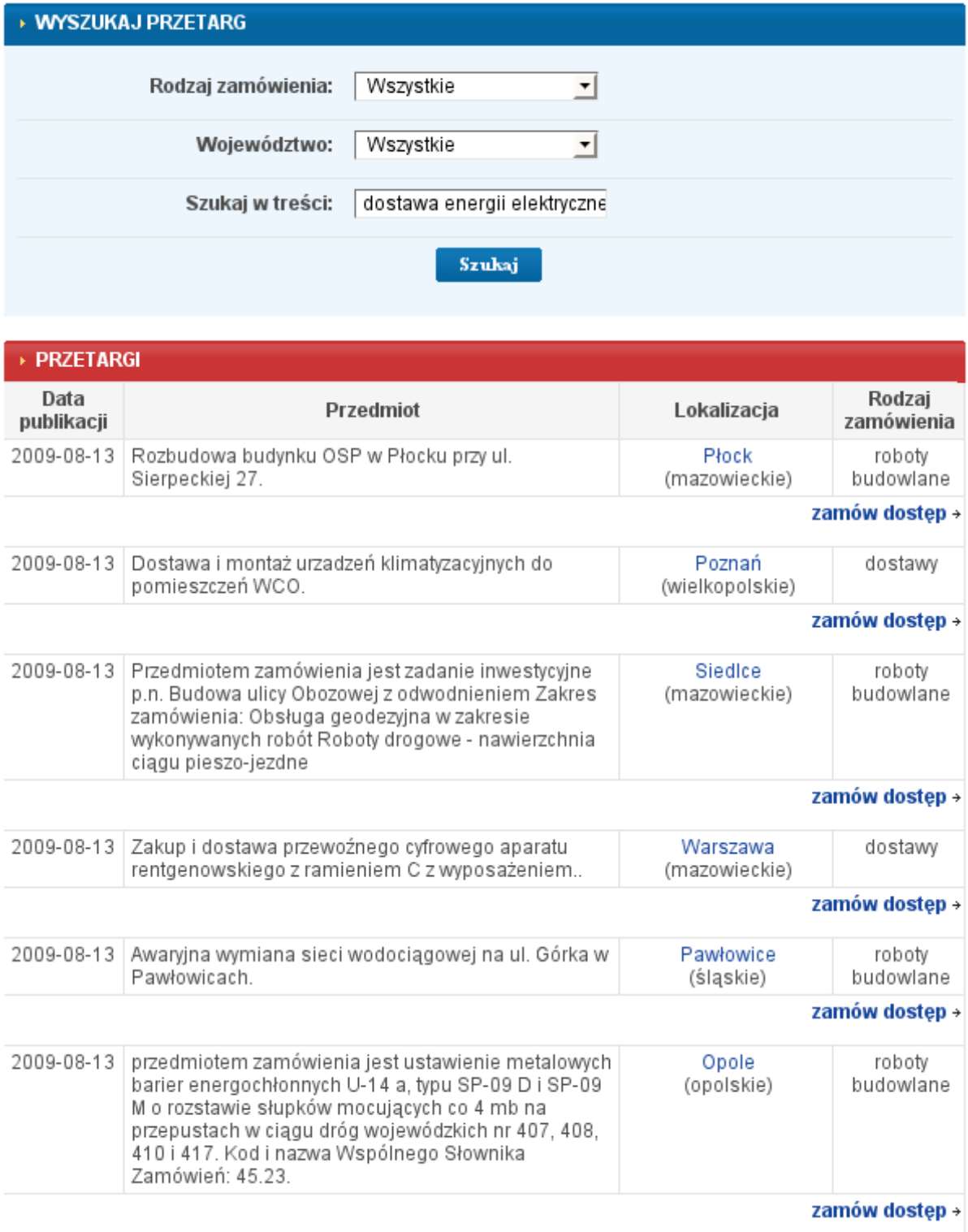
2.4. money.pl

Za usługi tego portalu nie trzeba płacić, aby uzyskać dostęp do podstawowych informacji o przetargach. Można wyszukiwać przetargi według:

- kategorii (usługi, dostawy, roboty budowlane),
- województwa,
- słów kluczowych.

Następnie system prezentuje wyniki i umożliwia zapoznanie się z podstawowymi informacjami o przetargu. Chcąc uzyskać bardziej szczegółowe informacje, należy się zarejestrować i dokonać zapłaty.

Rysunek 2.6: Wyszukiwarka portalu egospodarka.pl.



WYSZUKAJ PRZETARG

Rodzaj zamówienia: Wszystkie

Województwo: Wszystkie

Szukaj w treści: dostawa energii elektryczne

Szukaj

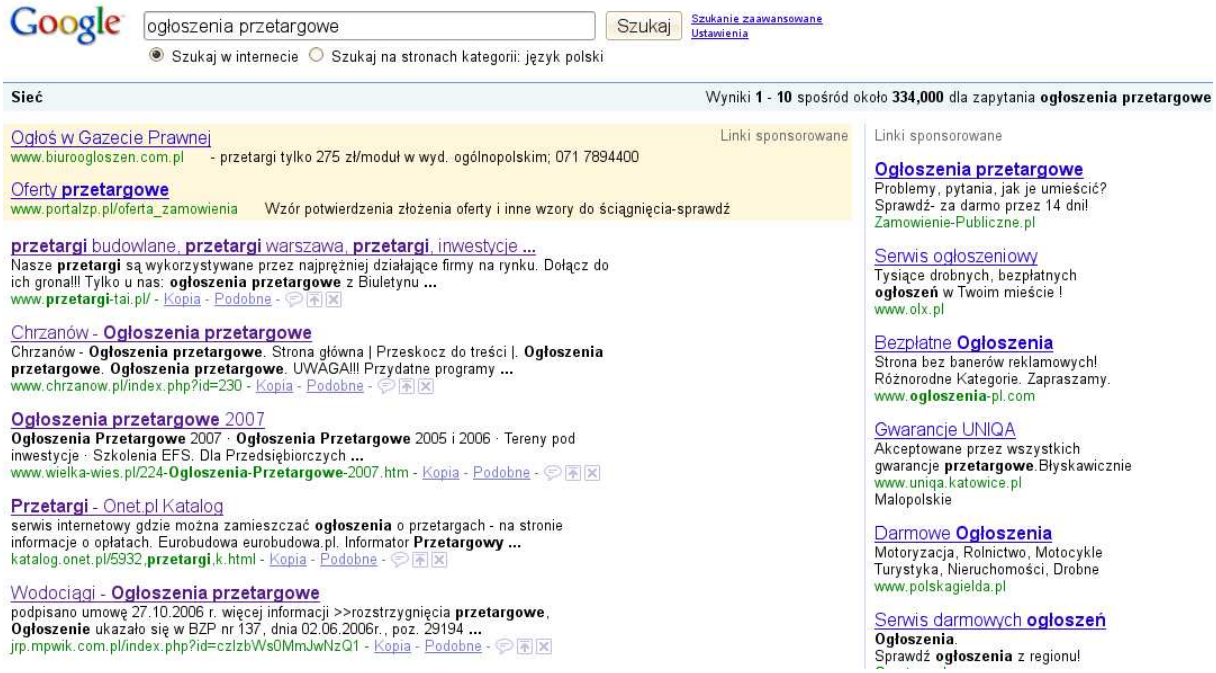
PRZETARGI

Data publikacji	Przedmiot	Lokalizacja	Rodzaj zamówienia
2009-08-13	Rozbudowa budynku OSP w Płocku przy ul. Sierpeckiej 27.	Płock (mazowieckie)	roboty budowlane
			zamów dostęp →
2009-08-13	Dostawa i montaż urzadzeń klimatyzacyjnych do pomieszczeń WCO.	Poznań (wielkopolskie)	dostawy
			zamów dostęp →
2009-08-13	Przedmiotem zamówienia jest zadanie inwestycyjne p.n. Budowa ulicy Obozowej z odwodnieniem Zakres zamówienia: Obsługa geodezyjna w zakresie wykonywanych robót Roboty drogowe - nawierzchnia ciągu pieszo-jezdne	Siedlce (mazowieckie)	roboty budowlane
			zamów dostęp →
2009-08-13	Zakup i dostawa przewoźnego cyfrowego aparatu rentgenowskiego z ramieniem C z wyposażeniem..	Warszawa (mazowieckie)	dostawy
			zamów dostęp →
2009-08-13	Awaryjna wymiana sieci wodociągowej na ul. Górka w Pawłowicach.	Pawłowice (śląskie)	roboty budowlane
			zamów dostęp →
2009-08-13	przedmiotem zamówienia jest ustawienie metalowych barier energochłonnych U-14 a, typu SP-09 D i SP-09 M o rozstawie słupków mocujących co 4 mb na przepustach w ciągu dróg wojewódzkich nr 407, 408, 410 i 417. Kod i nazwa Wspólnego Słownika Zamówień: 45.23.	Opole (opolskie)	roboty budowlane
			zamów dostęp →

Rysunek 2.7: Wyszukiwarka portalu money.pl

2.5. Google.com

Do wyszukiwania ogłoszeń przetargowych można oczywiście używać wyszukiwarek ogólnych takich jak Google. Konstruując odpowiednie pytanie, można odszukać informacje o przetargach. Wadą tej metody jest trudność sformułowania zapytania, które zwróci jak najwięcej wyników oraz konieczność ręcznej dyskryminacji błędnych wyników. Te zaś biorą się z tego, że wiele stron jest specjalnie wypozycjonowanych (techniki SEO), zwłaszcza w przypadku portali o tematyce przetargowej. Ponadto odnalezione ogłoszenia mogą być nieaktualne.



Rysunek 2.8: Wyniki wyszukiwania frazy “ogłoszenia przetargowe” w wyszukiwarce Google.

Wyszukiwarka Google góruje nad innymi portalami pod względem liczby zindeksowanych stron oraz bardzo zaawansowanych mechanizmów indeksujących i wyszukujących (korekta błędów, słowa podobne, odmiany). Dzięki temu można znaleźć ogłoszenia na bardziej egzotyczne produkty/usługi, do których niekoniecznie dotrą inne portale, jednak na użytkowniku spoczywa konieczność odfiltrowania stron nieznaczących oraz nieaktualnych.

3. Wstęp teoretyczny

W rozdziale tym przedstawiono algorytmy i techniki stosowane do odnajdywania informacji w Internecie. W zagadnieniu poszukiwania interesujących informacji można wydzielić dwa podstawowe etapy:

1. kategoryzacja dokumentów,
2. wyszukiwanie konkretnych informacji w dokumencie.

Zadaniem pierwszego kroku jest podział dokumentów na określone kategorie. W niniejszej pracy, wyróżniono dwie zasadnicze kategorie dokumentów:

1. ogłoszenia przetargowe,
2. inne strony.

Ponadto w celu zwiększenia skuteczności, zarówno podczas kategoryzacji jak i analizy dokumentów, stosuje się dodatkowe techniki. Są to m.in. stemming słów, wyliczanie podobieństw tekstów, lematyzacja.

3.1. Kategoryzacja dokumentów

Podstawową czynnością podczas przeszukiwania Internetu jest poprawna klasyfikacja odnalezionych dokumentów. Przed dokładną analizą strony, co jest zadaniem często czasochłonnym i skomplikowanym, należy stwierdzić czy jest to zasadne. Nie ma większego sensu dokładna analiza np. portalu społecznościowego, skoro nie występują na nim żadne informacje o przetargach i nie jest wcale związany z tematyką przetargową. Można jedynie pobieżnie przejrzeć taką stronę pod kątem występowania określonych słów i w przypadku stwierdzenia ich braku, bądź niewystarczającej ilości pominięcie strony z dalszej analizy. Określenie “niewystarczającej ilości” jest dość złożone. Algorytm stosowany do wstępnej klasyfikacji dokumentu ocenia go na podstawie wielu kryteriów. Dokładne jego działanie zostanie opisane szerzej w rozdziale 4.

Algorytmy stosowane do klasyfikacji dokumentów można podzielić na podstawowe grupy:

1. metody wektorowe/probabilistyczne (np TF, TF-IDF, Naiwny klasyfikator Bayesa),
2. drzewa decyzyjne,
3. reguły decyzyjne,
4. sieci neuronowe (np. Perceptron),
5. algorytmy genetyczne.

3.1.1. Algorytm TF

Jest to najprostszy z algorytmów klasyfikacji. Opiera się na zsumowaniu wystąpień danego słowa w dokumecie, a następnie podzieleniu tej liczby przez długość dokumentu (ilość wszystkich słów).

$$TF(D,w) = \frac{Tn(w)}{|D|} \tag{3.1}$$

gdzie: Tn(w) to ilość wystąpień słowa “w”,
|D| - ilość wszystkich słów w tekście.
Prostota tego algorytmu narzuca jednak pewne wady. Nie pozwala ona na wyróżnienie w ilu dokumentach dane słowo występuje. Przez to możliwe jest utrudnione klasyfikowanie dokumentów, w przypadku gdy słowo występuje w wielu dokumentach.

3.1.2. Algorytm TF-IDF

Algorytm TF-IDF (TF – term frequency IDF – inverse document frequency) wyznacza miarę w oparciu o ilość wystąpień słowa w dokumencie oraz ilości dokumentów, w których słowo występuje. Stosowany jest w ocenie istotności dokumentu w wyszukiwarkach oraz grupowaniu dokumentów.

$$TFidf(D,w) = \frac{Tn(w)}{|D|} * log(\frac{N}{DN(w)}) \tag{3.2}$$

gdzie:
Tn(w) to ilość wystąpień słowa “w”,
|D| - ilość wszystkich słów w tekście,
N - ilość wszystkich dokumentów,
DN(w) - ilość dokumentów zawierających słowo w.
Algorytm TF-IDF, w przeciwieństwie do swojej prostszej wersji czyli TF, uwzględnia ilość dokumentów w których dane słowo występuje. Dzięki temu, większy współczynnik TFIDF będą miały słowa, które występują w niewielu dokumentach, natomiast występujące we wszystkich tekstach otrzymają wynik równy 0. Wadą algorytmu TF-IDF jest konieczność przechowywania informacji o ilości słów we wszystkich dokumentach z osobna, co przy większych rozmiarach bazy, wiąże się z znacznym zapotrzebowaniem na pamięć [10].

3.1.3. Naiwny klasyfikator bayesowski [2]

Naiwny klasyfikator Bayesa opiera się na twierdzeniu stworzonym w 1763 roku przez Thomasa Bayesa:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{N} P(B|A_j)P(A_j)} \tag{3.3}$$

gdzie zdarzenia $A_1, A_2, ..., A_n$ są zdarzeniami niezależnymi.

Na tej podstawie można zdefiniować optymalny klasyfikator bayesowski. Jednak na złożoność obliczeń nie jest on praktycznie stosowany. Do klasyfikacji dokumentów używa się tzw. naiwnego klasyfikatora Bayesa. Naiwność tej metody wynika z założenia, że występowanie pewnych prawdopodobieństw (np. pewne słowa występują częściej obok innych, określonych słów niż bez żadnego związku) jest niezależne od siebie, co znacznie ułatwia wyliczenie prawdopodobieństw przynależności określonego zdarzenia do wybranej kategorii, jak również określenia kategorii najbardziej prawdopodobnej (a posteriori). Zadaniem naiwnego klasyfikatora Bayes'a jest zatem zakwalifikowanie nowego przypadku do jednej z kategorii decyzyjnych przy czym liczba kategorii musi być skończona i zdefiniowana (a priori). Do przypisania nowego zdarzenia do wybranej kategorii klasyfikator wykorzystuje twierdzenie Bayes'a, które przedstawiono w poprzednim paragrafie. Korzystając z definicji prawdopodobieństwa warunkowego oraz kolejnych przekształceń naiwny klasyfikator Bayes'a można zapisać jako:

$$NKB(x) = \arg\max_{c \in C} P(c|x) = \arg\max_{c \in C} \frac{P(x|c)P(c)}{P(x)} = \arg\max_{c \in C} P(c) \prod_i P(x_i|c) \tag{3.4}$$

gdzie:

NKB - naiwny klasyfikator Bayesa,

c - kategoria,

x - zdarzenie.

Wadą stosowania klasyfikatora Bayesa jest konieczność uczenia go na bardzo dużych zbiorach (rzędu kilkudziesięciu tysięcy dokumentów). Jeżeli zbiór uczący będzie zbyt mały, wówczas klasyfikator będzie zwracał nieprawdziwe wyniki. Ten algorytm stosuje się do filtracji spamu, klasyfikowania wiadomości.

3.1.4. Drzewa decyzyjne

Drzewa decyzyjne odwzorowują zależności pomiędzy dokumentami ze zbioru treningowego za pomocą struktury drzewa binarnego [18]. W węzłach umieszczone są pytania, na które odpowiedź jest w postaci binarnej (prawda albo fałsz). W liściach są kategorie, określone w procesie budowania drzewa na podstawie zbioru treningowego. Klasyfikacja polega na kolejnym odpowiadaniu na pytania, począwszy od korzenia drzewa aż po właściwy liść (kategorię). Prostota tego sposobu jest dużą zaletą. Algorytm ten

posiada jednak wiele wad. Pierwsza wynika z procesu budowy drzewa. Wymaga się, aby każdy dokument treningowy został sklasyfikowany prawidłowo. Aby spełnić ten warunek, odpowiednio dobierane są kwerendy oraz struktura drzewa. Dokumenty ze zbioru testowego będą poprawnie klasyfikowane, natomiast nowe już niekoniecznie. Drugą wadą jest wielkość drzewa jakie może powstać na etapie jego budowy. Zbyt duże drzewo decyzyjne powoduje spadek wydajności działania klasyfikatora. Można temu zapobiec określając maksymalną wysokość drzewa oraz minimalnej ilości dokumentów przypisanych do pojedynczej kategorii.

3.1.5. Reguły decyzyjne

Reguły decyzyjne określają dla każdej kategorii zbiór reguł opisujących jej profil. Reguła jest w postaci nazwy kategorii oraz słowa kluczowego, które najlepiej pasuje do dokumentów z danej kategorii [18]. Następnie tworzy się zbiór reguł, łączący je operatorami logicznymi. Zaletą tej metody jest jej stosunkowa prostota. Można również rozróżniać homonimy (słowa, które w zależności od kontekstu mają różne znaczenie). Dla przykładu, słowo "śledź" oznacza rybę, ale w kontekście namiotu słowo to określa mocowanie namiotu.

Wadą reguł decyzyjnych jest fakt, że trudno przypisać dokument tylko do jednej kategorii.

3.1.6. Sieci neuronowe [11]

Historia

Dziedzina sieci neuronowych zaistniała dopiero wraz z wydaniem historycznej pracy McCulloch'a i Pitts'a w 1943 roku, w której po raz pierwszy przedstawiono matematyczny opis komórki nerwowej oraz powiązanie go z problemem przetwarzania danych, co rozwinięto w kolejnych pracach tych samych autorów.

Zaprezentowany model wywarł wielki wpływ na późniejszy rozwój tej dziedziny. W 1949 roku Donald Hebb, odkrył, że informacja może być przechowywana w strukturze połączeń pomiędzy neuronami i jako pierwszy zaproponował metodę uczenia sieci polegającą na zmianach wag połączeń między neuronami (reguła Hebba). W latach 50-tych zaczęto budować pierwsze sieci neuronowe.

Pierwszym szeroko znanym przykładem zbudowanej i ciekawie działającej sieci neuropodobnej jest perceptron (Rosenblatt 1968). Sieć ta była przedstawiona jako układ częściowo elektromechaniczny, częściowo elektroniczny. Została ona zbudowana w 1957 roku w Cornell Aeronautical Laboratory. Jej zadaniem było rozpoznawanie znaków. Po próbach okazało się, że nie rozpoznawała bardziej złożonych znaków i wykazywała wrażliwość na zmianę skali obiektów, ich położenie w polu widzenia oraz zmiany kształtu. Zaletą była zdolność do zachowania poprawnego działania nawet po uszkodzeniu pewnej części

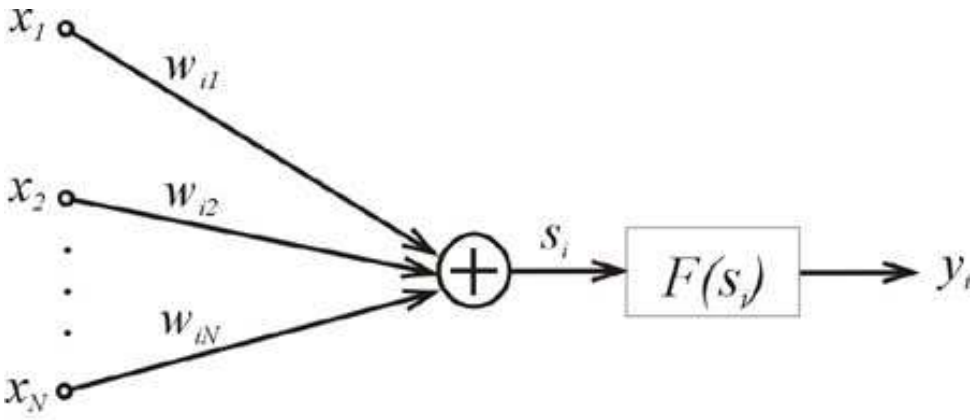
elementów. Po ogłoszeniu wyników przez twórców nastąpił gwałtowny rozwój tego typu sieci neuronowych na całym świecie

Własności

Podstawową własnością sieci neuronowych jest brak z góry określonego algorytmu rozwiązywania postawionego problemu. Wystarczy skonstruować sieć, a następnie na podstawie zbioru uczącego nauczyć sieć odpowiedzi na poszczególne elementy zbioru. Można dzięki temu osiągać dobre wyniki przy klasyfikowaniu dokumentów, które ciężko poprawnie kategoryzować na podstawie metod statystycznych. Dodatkową zaletą jest fakt zrównoleglenia obliczeń, dzięki czemu mogą one być szybsze niż programy wykonywane sekwencyjnie.

Podstawowy model - McCullocha-Pittsa

Na podstawie zasad działania rzeczywistego neuronu stworzono wiele modeli matematycznych, w których uwzględnione zostały w większym lub mniejszym stopniu właściwości rzeczywistych komórek nerwowych. Schemat obwodowy powiązany z większością tych modeli odpowiada modelowi McCullocha-Pittsa (rys. 3.1). Do wejść doprowadzane są sygnały dochodzące z neuronów warstwy poprzedniej. Każdy



Rysunek 3.1: Schemat budowy pojedynczego neuronu [11]

sygnał x_j mnożony jest przez odpowiadającą mu wartość liczbową zwaną wagą wij. Wpływa ona na percepcję danego sygnału wejściowego i jego udział w tworzeniu sygnału wyjściowego przez neuron. Zsumowane iloczyny sygnałów i wag stanowią argument funkcji aktywacji neuronu $f(s_i)$.

Zastosowanie

Sieci neuronowe stosuje się m.in. do klasyfikacji dokumentów. Program antyspamowy SpamAssassin stosuje wielowarstwowy perceptron z propagacją wsteczną do oceny czy wiadomość jest spamem czy nie [14][13]. Podstawowym problemem przy stosowaniu sieci neuronowej jako klasyfikatora jest zdefiniowanie wejścia sieci. W przypadku przetwarzania tekstu niełatwo jest wyróżnić zestaw cech opi-

sujących dokument, nie dysponujemy bowiem stałym wejściem, tak jak w przypadku przetwarzania obrazów. Można rozważyć użycie jako wejścia częstości występowania określonych słów. Wadą tego podejścia jest jednak brak informacji o kontekście ich występowania, co w rezultacie może dawać fałszywe wyniki jeśli treść dokumentu zawiera słowa kluczowe (np. jest dokumentem opisującym procedurę przetargową).

3.2. Wyszukiwanie informacji

Wyszukiwanie informacji polega na dokładnej analizie dokumentu w celu wydobycia z niego istotnych dla użytkownika informacji. W przypadku ogłoszeń przetargowych mogą być to:

- daty: realizacji, publikacji ogłoszenia, otwarcia ofert, zamknięcia.
- numery telefonów.
- województwo.
- kwota przetargu.
- numer CPV (Wspólny Słownik Zamówień).
- rodzaj przetargu: ograniczony, nieograniczony.
- branża.

Wykonany system wydobywa z dokumentu informację o branży (jest to dodatkowy rodzaj klasyfikacji oparty o algorytm TF), oraz datach dotyczących przetargu, oparte o poszukiwanie wzorców pasujących do wyrażenia regularnego oraz analizie jego otoczenia.

3.2.1. Wyrażenia regularne [17]

Wyrażenia regularne (ang. regular expressions) – wzorce, które opisują łańcuchy symboli. Teoria wyrażeń regularnych jest związana z teorią języków regularnych. Wyrażenia regularne pozwalają na stwierdzenie czy dany łańcuch tekstowy spełnia wyrażenie. Można za ich pomocą wydobywać z tekstu fragmenty pasujące do wzorca.

Z punktu widzenia teoretycznego, wyrażenia regularne opisują języki regularne. Praktycznie stosuje się jednak bardziej rozbudowaną składnię, która niekoniecznie musi definiować język regularny. Przykładem są referencje wsteczne (backreferencing).

Wyrażenia regularne są częścią narzędzi systemowych (szczególnie powszechnie stosowanych w systemach rodziny *NIX) oraz bibliotekami do praktycznie wszystkich języków programowania (Java, C#, Python itp.).

Dwie najpopularniejsze składnie wyrażeń regularnych to składnia uniksowa i składnia perlowa. Składnia perlowa jest znacznie bardziej rozbudowana. Jest ona używana nie tylko w języku Perl, ale także w innych językach programowania: Ruby, bibliotece PCRE do C i w narzędziu powłoki o nazwie pcregrep (znanego też jako pgrep). Perlową składnię stosuje się również w maskach przepisań mod rewrite[17].

3.2.2. Wyszukiwanie pełnotekstowe

Aby skutecznie przeszukiwać duże zbiory dokumentów, niezbędne jest wykorzystanie mechanizmów wyszukiwania pełnotekstowego. Wyszukiwanie takie polega na indeksowaniu słów w dokumencie, i tworzeniu referencji indeksowanych słów do dokumentów. W celu ograniczenia rozmiaru indeksu, należy usunąć z tekstu słowa nieznaczące (“stop words”) takie jak spójniki, rodzajniki itp. Należy również przeprowadzić lematyzację bądź stemming, aby jeszcze zmniejszyć rozmiar indeksu i zwiększyć ilość odnajdywanych wyników. Stemming słów może się odbywać algorytmicznie (np. Snowball) bądź przy użyciu słowników (np. ispell). W miarę możliwości można również słowa o podobnym znaczeniu zapisywać w podstawowej postaci przy użyciu słownika wyrazów bliskoznacznych (Thesaurus). W przypadku PostgreSQL indeks budowany jest jako drzewo GIN (Generalized Inverted Index) albo GiST. W przypadku wyszukiwania pełnotekstowego implementacja nazywa się “tsearch2”. Istnieją duże różnice implementacyjne pomiędzy GiST a GIN. W przypadku GiST dokument indeksuje się jako wektor hashy o stałej długości dla wszystkich słów. Możliwe jest więc odnalezienie dokumentów, w których hashe słów mają taką samą wartość jak hashe słów zapytania, choć mogą być to zupełnie różne wyrazy. GIN przechowuje lexemy, więc nie stwarza takich problemów. Niestety, indeks GIN tworzy się wielokrotnie wolniej, jest też powolniejszy w aktualizacji i zajmuje więcej miejsca niż GiST. [12]. Poniżej znajduje się przykładowa analiza słowa “Dostawa energii elektrycznej”:

```
SELECT token, lexemes FROM
    ts_debug('public.polish','Cykliczna dostawa oleju opałowego')
    WHERE alias <> 'blank';
```

```
   token   |   lexemes
-----------+-------------
 Cykliczna | {cykliczny}
 dostawa   | {dostawa}
 oleju     | {olej}
 opałowego | {opałowy}
```

3.3. Operacje stosowane podczas przetwarzania dokumentu

3.3.1. Stemming i lematyzacja

Stemming

Stemming polega na wyznaczeniu stemu (tematu), czyli znajdywaniu części wyrazu, która nie uczestniczy w odmianie. Można wyróżnić stem fleksyjny (domow -> domownik, domowy) oraz stem słowotwórczy (dom->domownik, domator, domowy) [16]. Istnieje kilka technik stemmingu:

1. **Brute force**: polega na utworzeniu, a następnie przeszukiwaniu tablicy zależności pomiędzy formami zdania. Zaletą tej metody jest możliwość stemmingu wyrazów, które są formami odmiany nieregularnej, w przypadku których nie zadziałają poprawnie stemmery algorytmiczne. Wadą natomiast jest konieczność przechowywania tabeli oraz czasochłonność jej przeszukiwania. Ponadto, ciężko jest utworzyć kompletną tabelę dla danego języka ze względu na ogrom słów.

2. **Usuwanie końcówek (suffix stripping)**: algorytmy te opierają się na zasadach tworzenia odmiany słowa. Są szczególnie skuteczne dla języków o prostej fleksji (takich jak angielski), gdzie większość słów pochodnych tworzy się przez dodanie określonych końcówek. Algorytm ten jednak nie zadziała dla odmiany nieregularnej.

Należy zwrócić uwagę, że stem nie musi być poprawnym wyrazem w danym języku (np. w języku polskim nie ma słowa “domow”). Wykorzystany w programie stemmer to Stempel oraz Lameryzator. Stempel jest stemmerem algorytmicznym, natomiast Lameryzator oparty jest o słowniki ispella oraz automat skończony (FSA).

Lematyzacja

Lematyzacja jest procesem przywracania formy odmienionej słowa do jego formy podstawowej. Dla przykładu lematyzacja słowa “kota” da w wyniku słowo “kot” Lematyzacja jest procesem bardziej złożonym niż stemming, gdyż forma podstawowa musi być poprawnym słowem języka.

3.3.2. Miary podobieństw tekstów

Odległość Levenstheina [1]

Miara ta została opracowana przez Vladimira Levenstheina w roku 1965. Polega ona na wyliczeniu jak najmniejszej ilości działań, które prowadzą do uzyskania z pierwszego napisu drugi. Wyróżnia się następujące działania:

1. Dodanie nowego znaku do napisu.

2. Usuniecie znaku z napisu.

3. Zamianę znaku na inny.

Dla przykładu więc, dwa identyczne napisy będą miały zerową odległość, gdyż nie wymagają przeprowadzenia żadnej operacji na porównywanym tekście. Odległość pomiędzy słowami “granat” i “granit” wynosi 1 (zamiana “a” na “i”). Przy szacowaniu podobieństw słów ważne jest zauważenie faktu, że różne słowa mają odległość wynoszącą co najmniej jeden a co najwyżej długość słowa dłuższego.

Soundex [15]

Jest to algorytm fonetyczny, opracowany na początku wieku XX (lata 1918-1922). Opiera się on na wyliczeniu tzw. kodu Soundex o długości 4 znaków. Słowa podobnie brzmiące będą miały ten sam kod. Wyliczanie kodu odbywa się według następujących reguł:

1. Pierwszy znak kodu to pierwsza litera słowa.

2. Usunięte zostają samogłoski oraz litery “h” i “w”.

3. Pozostałym literom przypisuje się liczby:

   b, f, p, v => 1

   c, g, j, k, q, s, x, z => 2

   d, t => 3

   l => 4

   m,n => 5

   r => 6

4. Do kodu jest brane pod uwagę tylko pierwsze wystąpienie kodu znaku, pozostałe są pomijane.

5. Jeżeli występują więcej niż 3 znaki, dalsze są usuwane, jeśli kod ma mniej niż trzy to wypełnia się brakujący kod zerami.

Algorytm ten pozwala znaleźć podobnie brzmiące słowa, nawet jeśli wystąpi literówka w słowie, bądź słowo zostanie źle napisane (ale podobnie fonetycznie). Wadą jego jest natomiast to, że opisuje on reguły dla języka angielskiego, co powoduje, że porównanie np słów “beton” i “prąd” wskazuje na ich podobieństwo.

Q-Gram [3]

Q-Gram jest metodą przybliżonego porównywania tekstów. Polega na podzieleniu porównywanych ciągów znaków na kawałki (“gram”) o długości q. Następnie porównuje się ilość identycznych “gramów”

w obu ciągach w odniesieniu do całkowitej ich ilości. Za takim postępowaniem stoi intuicja. Jeżeli ciągi c_1 i c_2 są do siebie podobne, to będą miały dużą liczbę wspólnych "gramów". Umożliwia to określenia stopnia podobieństwa słów, poprawiając skuteczność wyszukiwania informacji. Zmniejsza to bowiem wrażliwość wyszukiwarki na błędy (literówki, błędy gramatyczne itp.) oraz odmianę słów. Z drugiej jednak strony, może być przyczyną znajdywania niewłaściwych słów, które pomimo podobieństwa, mają inne znaczenia.

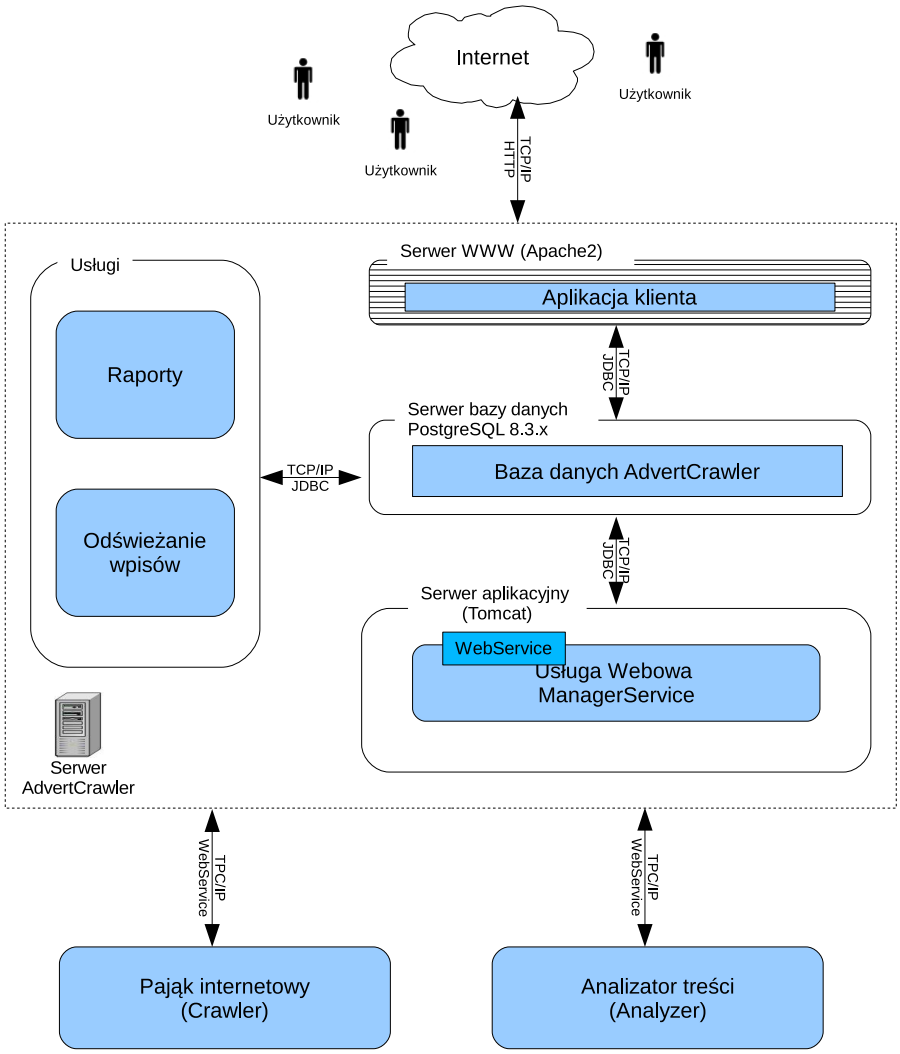
4. Projekt

4.1. Ogólny projekt architektury systemu

Na rysunku 4.1 przedstawiono ogólną architekturę systemu. Jak widać składa się ona z dwóch podstawowych elementów: serwera oraz pająków i analizatorów. Zadaniem serwera jest udostępnianie użytkownikowi interfejsu w postaci wyszukiwarki internetowej oraz interfejsu dla programów pobierających i oceniających treść z Internetu. Obok strony internetowej, pająków oraz analizatorów znajdują się jeszcze dwa programy. Pierwszy z nich, oznaczony na rysunku jako "Raporty" zajmuje się okresowym przeglądaniem bazy w poszukiwaniu nowych ogłoszeń, spełniających kryteria podane przez użytkownika. Jest to w zasadzie zwykłe wyszukiwanie ogłoszeń, takie samo jakie może robić użytkownik. Różnica polega na tym, że to system pamięta o sprawdzaniu bazy i tworzy na bieżąco listę interesujących ogłoszeń.

Kolejnym z programów pobocznych jest "Odświeżacz". Przeglądanie Internetu może zająć bardzo dużo czasu i niekoniecznie jest czynnością pożądaną. Wymaga się, aby system wracał do stron, na których pojawiają się ogłoszenia. Można to wykonać, poprzez analizowanie stron "bazowych" dla zgromadzonych ogłoszeń. Strona "bazowa" oznacza witrynę, której odnośnik wskazał na ogłoszenie przetargowe. Jeśli znajdziemy stronę, na której było wiele ogłoszeń, to jest prawdopodobne, że będą się one pojawiać na niej częściej. Takie adresy są ciekawe i warto jest ponownie je indeksować. To jest właśnie zadanie "Odświeżacza". Dodatkowo usuwa on strony, które nie okazały się ogłoszeniami oraz nie zawierały odnośników do ogłoszeń.

Programy (Analizator i Pająk) komunikują się z serwerem za pomocą wywołań usług sieciowych udostępnianych przez serwer. Serwer przechowuje dane w bazie, która udostępnia również mechanizm wyszukiwania pełnotekstowego. Jako silnik bazy został wybrany PostgreSQL w wersji 8.3.7 z wbudowanym modułem "tsearch2". Odświeżacz i Generator raportów korzystają wprost z połączenia przez sterowniki JDBC.



Rysunek 4.1: Ogólna architektura systemu.

4.1.1. Pająk

Pająk (robot) to aplikacja, której zadaniem jest pobieranie stron z Internetu, szybka ocena istotności treści oraz ekstrakcja odnośników do innych stron, a następnie kolejkowanie potencjalnych ogłoszeń do dalszej analizy.

Ocena strony

Pierwszą czynnością było zapoznanie się ze strukturą typowego ogłoszenia przetargowego. Rysunek 4.2 prezentuje przykładową ofertę. Jak widać, składa się ona z kilku typowych części:

- **Nagłówek** - określa tytuł przetargu, podmiot zamawiający, kwotę.
- **Określenie przedmiotu zamówienia** - w sekcji tej opisany jest zakres przetargu, jego typ, kod słownika CPV, informacja o możliwości składani ofert częściowych i wariantowych oraz terminy realizacji.
- **Warunki składania ofert** - zawierają informację o wymaganiach jakie muszą zostać spełnione, aby móc przystąpić do przetargu. Tutaj podane są też daty składania i zamknięcia ofert.

Ogłoszenia przetargowe skonstruowane są zazwyczaj w powyżej opisany sposób. Niestety, bardzo dużo ofert posiada całkowicie inną formę, co znacznie komplikuje proces analizy dokumentu. Nie zawsze podane są informacje o wszystkich datach, sekcje nie są wyróżnione w tak jawny sposób, niekiedy jest tylko krótka informacja o ogłoszeniu.

Można zauważyć, że na wszystkich występuje słowo "przetarg", "zamówienie" bądź "oferta". Można więc przyjąć ten fakt jako podstawowe kryterium oceny. Jeśli na stronie nie występuje żadno z wyżej podanych słów, to taka strona nie ma związku z tematyką przetargową. Każda z fraz ma przypisaną wagę:

1. przetarg **3**,
2. ogłasza **2**,
3. termin **1**,
4. przedmiot **3**,
5. oferta **1**,
6. realizac **1**,
7. zamówie **1**.

Najwyższą wagę ma słowo przetarg, gdyż ono jednoznacznie wskazuje na stronę o interesującej nas tematyce. Jednak nie na wszystkich stronach słowo to występuje, czasami można spotkać tylko "oferta" bądź "zamówienie". Ponieważ te słowa mogą się pojawić na różnych stronach, stąd lista ta jest rozbudowana o dodatkowe wyrażenia, wskazujące na możliwe ogłoszenie. System wyszukuje kluczowe frazy, oraz wylicza podobieństwo słów przy pomocy metryki Q-Gram, a następnie mnoży wynik razy wagę słowa. System nie bierze pod uwagę częstości występowania słowa, a jeden najwyższy wynik.

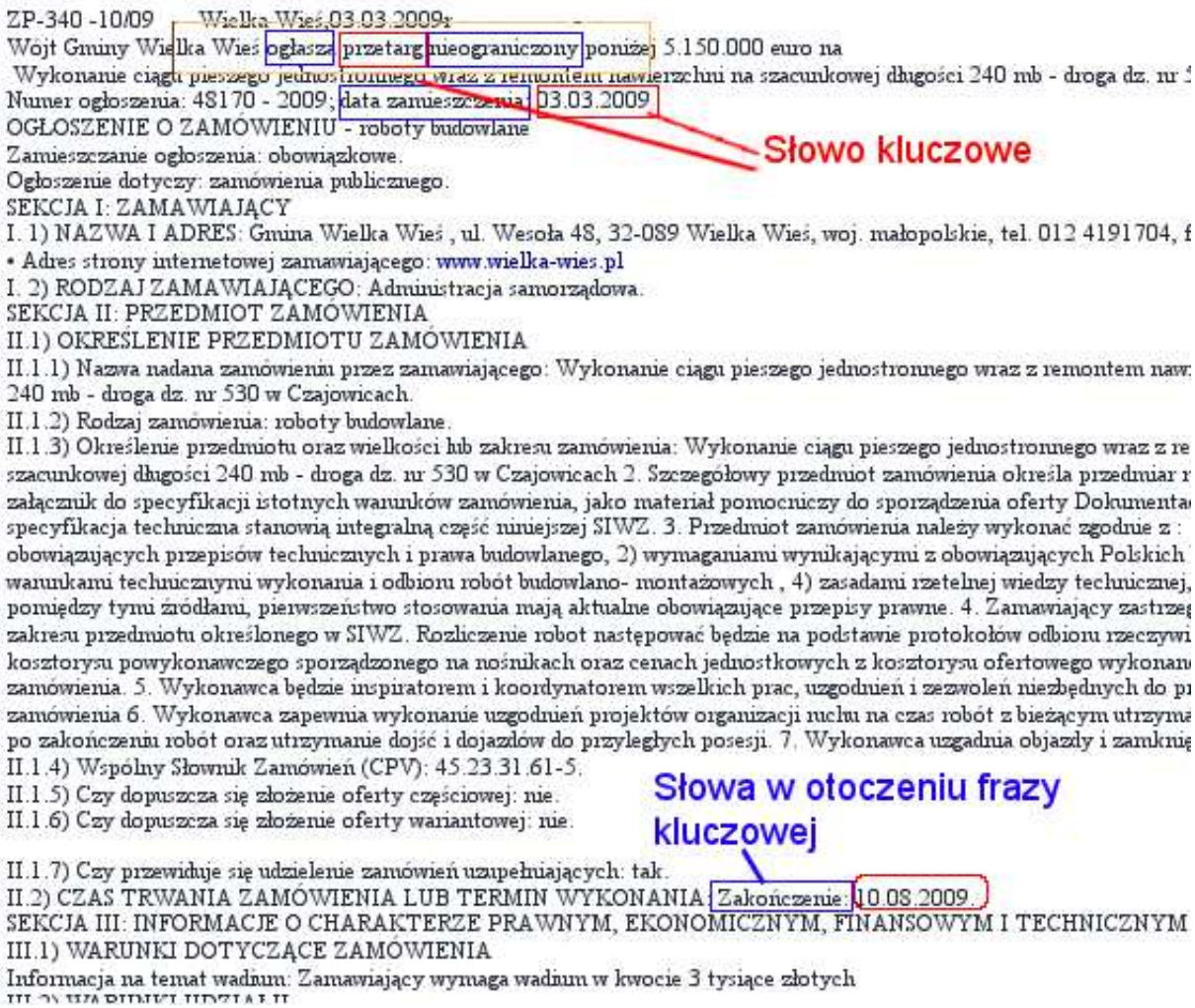
Badane jest również otoczenie odnalezionej frazy, ponieważ jak zostało pokazane na rysunku 4.2 - wokół słów kluczowych występują często inne słowa, wskazujące, że znaleźliśmy ogłoszenie przetargowe. Słowa te to:

1. ogłasza,
2. ograniczon,
3. publiczn.

Za odnalezienie w otoczeniu słowa kluczowego również są przyznawane punkty (podobieństwo słowa * 1). Następnie wyliczana jest suma punktów. Należy zauważyć, że są to podstawowe rdzenie słów. Poszukiwanie podstawowych rdzeni wynika z faktu, że pająk nie wykonuje stemmingu słów w dokumencie, gdyż jest to operacja dość kosztowna. System przyznaje ujemne punkty za frazy w adresie URL:

1. page,
2. order,
3. sort.

Frazy te oznaczają, że dana strona wynika tylko ze zmiany kolejności elementów na liście ogłoszeń. Konieczność dyskryminacji tych fraz wynika z faktu, że czasami strona może zostać zaklasyfikowana jako ogłoszenie, mimo, że w rzeczywistości jest to tylko prezentacja wszystkich ogłoszeń. Taka strona będzie miała niską punktację, minimalnie ponad progiem (w chwili obecnej równym 6.0). Odjęcie punktów za niedozwolone frazy w adresie spowoduje, że nie zostanie ona zaklasyfikowana jako ogłoszenie i nie będzie poddawana dalszej analizie.



Rysunek 4.2: Przykład typowego ogłoszenia przetargowego, z zaznaczonymi interesującymi frazami.

Jeżeli punktacja strony przekroczy wynik 6.0 to jest ona uznawana za prawdopodobne ogłoszenie przetargowe i dodana do kolejki analizatora.

Ekstrakcja i ważenie odnośników

Pająk dokonuje nastepnie ekstrakcji linków i nadaje im wagę. Każde spełnienie warunku, określonego poniżej, dodaje 1 do wyniku.

1. Czy dokument jest ogłoszeniem przetargowym?

2. Czy adres URL bądź treść zawierają słowo kluczowe wskazuje na przetarg?

Odnalezione wyniki, wraz z ich wagami oraz adresem strony na której zostały znalezione, trafiają do bazy danych do kolejki pająka. Dodanie wagi dla linku pozwala na zwiększenie efektywności działania całego systemu, gdyż w pierwszej kolejności będą sprawdzane strony, na których bardziej prawdopodobne jest wystąpienie przetargów.

Ograniczenia

Aby zwiększyć wydajność systemu możliwe jest rozproszenie pająków i analizatorów (łączą się one z bazą poprzez wywołania usługi sieciowej). W tej chwili system nie może działać z pełną prędkością, ponieważ brak jest mechanizmu rozpraszania ruchu, a kilka pająków działających jednocześnie może wygenerować dużą ilość zapytań do analizowanych serwerów (podczas testów, 3 instancje pająka generowały ciągły ruch na poziomie 2Mb/s). Grozi to interwencją administratora serwera i zablokowaniem adresów na których działa pająk. Ponadto, robot musi uwzględniać obostrzenia nakładane przez plik "robots.txt" oraz nagłówek dokumentu HTML. Jeżeli właściciel serwera bądź serwisu nie życzą sobie aby ich strony były indeksowane należy tego honorować, gdyż w przeciwnym wypadku grozi wpisanie pająka na czarną listę aplikacji, które nie mają dostępu do treści danego serwisu [7]. Program potrafi czytać plik "robots.txt" oraz sprawdza czy strona może być indeksowana(INDEX) oraz czy robot może przeglądać podstrony(FOLLOW).

4.1.2. Analizator treści

Kolejnym programem jest analizator treści. Wyciąga on tekst dokumentu (z pominieciem linków, styli oraz niektórych pól formularzy), a następnie buduje histogram słów, z pominieciem fraz nieznaczących (tzw. "stop-words"). Dokonywany jest również stemming słów, w celu zmniejszenia histogramu oraz zwiększenia skuteczności kategoryzacji dokumentu.

Kategoryzacja dokumentów

Kategoryzacja dokumentu opiera się na naiwnym klasyfikatorze Bayesa, stwierdzającym w pierwszej kolejności czy analizowany dokument jest faktycznie ogłoszeniem. W dalszej kolejności, na podstawie słowników branżowych, jest wybierana podstawowa branża ogłoszenia. System dokonuje porównania histogramu dokumentu ze słownikami. Analiza polega na sprawdzeniu czy histogram nie zawiera słowa zdefiniowanego dla branży. Za zgodność jest dodawany jeden punkt. "Wygrywa" branża z największą ilością trafień. Jest to prosty algorytm, jednak dość skuteczny. Zaletą tak prostego podejścia jest fakt, że nie wymaga ono żadnego uczenia. Wystarczy dopisać branżę i zbudować słownik słów kluczowych.

Poszukiwanie dat

Po określeniu branży następuje proces ekstrakcji dat. Na rysunku 4.2 zaznaczono schemat działania analizatora. Czerwoną obwódką zaznaczono słowa kluczowe, bądź dopasowane wzorce (wyszukiwanie

dat). Kolorem niebieskim otoczono słowa, które są poszukiwane w otoczeniu odnalezionej frazy kluczowej.

Wyrażenie regularne, które zostało zastosowane, dopasowuje daty w różnym formacie: RRRR-MM-DD, DD-MM-RRRR, RRRR.MM.DD, DD.MM.RRRR, DD MM RRRR, RRRR MM DD. W Polsce najczęściej stosowane są te formaty. Nie zostaną sparsowane daty z dwucyfrowym sposobem zapisu roku. Trudno określić, na której pozycji występuje rok (tyczy się to dat po 2000 roku). Po odnalezieniu daty jest sprawdzane jej otoczenie w poszukiwaniu kluczowych fraz. Stwierdzono, że ogłoszenia zazwyczaj posiadają następujące typy dat:

1. data publikacji ogłoszenia,
2. data otwarcia,
3. data zamknięcia,
4. data realizacji zamówienia.

Ważenie dokumentu

Ogłoszenia przetargowe, które można znaleźć w Internecie charakteryzują się bardzo różną budową, co znacznie utrudnia proces analizy. Często są to kompletne dokumenty, czasami jedynie wzmianka o przetargu. Trudno więc określić, jakie formaty dokumentów mają być uznawane jako ogłoszenie, a jakie odrzucane. Stąd też na pierwszym etapie analizy, dokonywanym przez pająka, nie jest brana pod uwagę ilość powtórzeń słowa, a jedynie fakt jego wystąpienia. Jednak takie podejście powoduje problem tzw. "false-positives" czyli fałszywych pozytywów. Polega to na tym, że strony, które przetargami nie są, zostają sklasyfikowane jako ogłoszenia. Takie dokumenty, po dokładnej ocenie, powinny zostać odrzucone przez analizator. Niestety, bardzo ciężko jest ocenić bezbłędnie czy dana strona jest faktycznym ogłoszeniem przetargowym, czy tylko niewiele znaczącym wpisem na forum dyskusyjnym. W związku z tym, należy wyznaczać wagę dokumentu. Przyjęto, że za każdą odnalezioną informację (data, kategoria) będzie przypisany jeden punkt. Jeżeli treść zostanie sklasyfikowana jako przetarg, dodany zostanie punkt do wyniku. Dzięki temu, podczas wyszukiwania przetargów, takie "fałszywe" wyniki znajdą na samym końcu.

4.1.3. Baza danych

Dużym wyzwaniem jest ilość danych, jakich można się spodziewać. Podczas testów, system (jedna instancja pająka i analizatora) był w stanie zgromadzić w ciągu 30 godzin około 300 000 unikalnych adresów URL oraz około 100 000 dokumentów sklasyfikowanych jako przetargi. Aby ułatwić i przyspieszyć

wyszukiwanie ogłoszeń, konieczne było zastosowanie mechanizmów wyszukiwania pełnotekstowego. Wybrany został wbudowany mechanizm bazy PostgreSQL, ze względu na łatwość wykorzystania i wystarczającą funkcjonalność. Bardziej zaawansowanym systemem indeksowania jest np. Apache Solr bądź Apache Lucene [4] [5]. PostgreSQL FTS umożliwia budowanie indeksów oraz zadawanie zapytań boolowskich (operatory OR, AND i NOT). Aby skorzystać z ww. mechanizmów, konieczne jest jedynie zainstalowanie PostgreSQL w wersji 8.3 lub wyższej i poprawne skonfigurowanie modułu tsearch2. Wydajność bazy danych jest zadowalająca pod warunkiem, że nie używa się wbudowanego mechanizmu rankingu wyników, gdyż przy dużej bazie (ok. 100 000) proces oceniania zajmuje znaczącą ilość czasu (nawet kilkanaście sekund na pojedyncze zapytanie).

5. Realizacja

5.1. Informacje ogólne

W rozdziale tym opisane zostanie, od strony technicznej, jak wygląda realizacja systemu. System składa się on z dwóch podstawowych części. Jest to serwer, który udostępniający bazę oraz interfejs dla pająków i analizatorów. Mogą one łączyć się z serwerem poprzez protokół SOAP. Możliwe jest więc korzystanie z programów, napisanych w dowolnym języku, w którym dostępne są biblioteki do tego protokołu. Kompletny dokument WSDL, opisujący usługę, znajduje się w załączniku A. Ze względu na wydajność oraz rozbudowane narzędzia programistyczne, całość została napisana w języku Java. Pierwsze wersje były napisane w języku Python. Ze względu na mniejszą wydajność oraz brak natywnej (tj. napisanej w Pythonie) biblioteki do stemmingu słów konieczna była zmiana platformy. Jedyna dostępna biblioteka była w postaci binarnej i skompilowanej dla środowiska 32 bitowego, co uniemożliwiało uruchomienie systemu na komputerze z 64 bitowym systemem operacyjnym.

5.2. Elementy systemu

5.2.1. Pająk

Zadaniem pająka jest pobranie zawartości strony, wyodrębnienie tekstu, linków i przekazanie do dalszej analizy. System potrafi odczytywać pliki w postaci tekstowej (typ MIME “text”), dokumenty PDF(“application/pdf”) oraz dokumenty MS Word (“application/msword”). Warunkiem poprawnego odczytania plików w formacie PDF i Word jest postać tekstowa umieszczonego w nich ogłoszenia (system nie ma mechanizmów OCR). Ponadto system nie ściąga plików większych niż 1MB oraz przerywa przetwarzanie po 10 minutach. System w obecnej wersji nie bierze pod uwagę elementów składni HTML, ponieważ jest ona bardzo często niewłaściwie stosowana i trudna do przewidzenia. W przeciwieństwie do PDF czy plików Word, dokumenty HTML nie mają jasno zdefiniowanego standardu (oczywiście istnieją zalecenia, np. organizacji W3C). Z tego względu strony HTML mają bardzo nieprzewidywalny kod źródłowy. Stwarza to istotny problem, gdyż dostępne biblioteki do parsowania kodu HTML niekiedy nie potrafią poprawnie odtworzyć drzewa dokumentu, co skutkuje z kolei nieskutecznym wyodrębnianiem

tekstu. W tym celu najpierw system wykonuje korektę kodu HTML, usuwa elementy odnośników, styli oraz skryptów, a dopiero wówczas wyodrębnia tekst. Takie podejście jest dużo bardziej czasochłonne ale owocuje poprawną analizą strony. W aplikacji są również zaszyte mechanizmy odgadywania kodowania (czasami można spotkać strony, które deklarują kodowanie znaków typu "ISO-8592").

Odczyt z dokumentów "doc", w postaci których można odnaleźć znaczną ilość ogłoszeń, jest możliwy dzięki bibliotece Apache POI (dołączona do źródeł projektu). Analiza dokumentów w postaci plików programu Microsoft Word nie stwarza większych problemów, a dodanie obsługi tego formatu wzbogaca znacznie możliwości systemu.

Pająk potrafi również wyciągać informacje z plików PDF. W tym przypadku nie wykorzystywana jest żadna biblioteka Javy, gdyż istniejące są zbyt duże, a jedyna prosta biblioteka (PDFBox) stwarzała problemy i działała mało stabilnie. Na szczęście, systemy Linux posiadają zestaw narzędzi dostarczanych z pakietem "xpdf". Jednym z programów, który został wykorzystany, jest "pdf2text". Potrafi on wyciągać zawartość tekstową z dokumentów i zapisywać ją do postaci pliku tekstowego bądź prostego dokumentu HTML. Wykorzystano drugą możliwość, gdyż analizator pracuje na dokumentach w postaci HTML (umożliwia to dalszą jego rozbudowę o bardziej skomplikowane mechanizmy parsowania).

Po wyodrębnieniu tekstu, pająk dokonuje: jego oceny (algorytm opisany w rozdziale 4), ekstrakcji odnośników (również względnych, z odrzuceniem kotwic), oraz nadaje im wagi (opis w poprzednim rozdziale). Następnie wysyła je do usługi sieciowej w postaci tablicy struktur (opisanej plikiem WSDL - dodatek A).

Po odebraniu takiej tablicy następuje zapisanie jej do bazy. Wszystkie odnośniki są dodawane do kolejki pająka. Jeżeli któraś strona została rozpoznana jako ogłoszenie, to jej treść zostanie wpisana do kolejki analizatora.

Pająk wymaga odpowiedniej konfiguracji do poprawnego działania. Plik konfiguracyjny jest dokumentem XML, składającym się z wpisów w postaci:

```
<entry key="nazwa_klucza">wartość</entry>
```

Dla pająka najważniejsze są:

- *webserviceurl*: określa adres WebService udostępniającego funkcje pobierania zadań pająka i wpisywania wyników do bazy.
- Wpisy konfiguracyjne systemy log4j.

5.2.2. Analizator

Analizator dokonuje oceny dokumentu (w chwili obecnej tylko na podstawie jego reprezentacji tekstowej). Nie są analizowane znaczniki HTML takie jak pogrubienia, nagłówki czy tytuł. Elementy te są

zbyt często używane w niewłaściwy sposób i mogłyby wprowadzać błędy do wyniku. Koncepcja działania analizatora jest opisana w rozdziale 4. Analizator wymaga również poprawnego skonfigurowania plikiem o budowie opisanej w sekcji 5.2.1. W przypadku analizatora istotne są wpisy:

- *webserviceurl*: określa adres WebService udostępniającego funkcje pobierania zadań pająka i wpisywania wyników do bazy.
- *stripper.dicFolder*: ścieżka do katalogu z plikami słowników branż.
- *stripper.excludesFolder*: ścieżka do katalogu z plikami słów wykluczonych.
- *classificator.excludesFile*: ścieżka do pliku z stop-listą.
- *stripper.DBFileMap*: plik mapujący nazwę branży - ID branży.
- *dates.publish*: lista wyrazów(oddzielonych średnikami), które występują w pobliżu daty publikacji.
- *dates.realisation*: lista wyrazów(oddzielonych średnikami), które występują w pobliżu daty realizacji.
- *dates.open*: lista wyrazów(oddzielonych średnikami), które występują w pobliżu daty otwarcia.
- *dates.close*: lista wyrazów(oddzielonych średnikami), które występują w pobliżu daty otwarcia.
- Wpisy konfiguracyjne systemy log4j.

Pliki słowników branż są prostymi plikami tekstowymi, w których kolejne słowa kluczowe podane są w nowej linii. Muszą one mieć rozszerzenie ".dic". Podobnie skonstruowane są pliki z słowami wykluczonymi.

5.2.3. Odświeżacz

"Odświeżacz" jest prostym programem, który powinien być uruchamiany cyklicznie. Jego zadaniem jest usuwanie z bazy nieistotnych adresów URL (czyli takich, które zostały przeglądnięte przez analizator i nie zaklasyfikowane jako ogłoszenie). Pozwala to na pewną redukcję rozmiarów bazy. Bardzo istotnym zadaniem odświeżacza jest przejrzenie bazy zgromadzonych ogłoszeń, ustalenie ich źródła (refferer) oraz jego krotność. Jeśli któryś adres URL źródła powtarza się częściej niż zadana ilość razy (teraz ustalona jest wartość 5 powtórzeń) wówczas można podejrzewać, że na tej stronie mogą pojawiać się ogłoszenia. Taka strona zostanie dodana do kolejki pająka, który sprawdzi czy nie ma na niej nowych ogłoszeń. Dzięki temu rozwiązaniu strony, na których pojawiają się ogłoszenia, będą okresowo ponownie

sprawdzane. Plik konfiguracyjny ma nazwę *Refresher.properties* i jest plikiem tekstowym, składającym się z par klucz=wartość. Należy ustawić następujące parametry:

- *connection.hostConfig*: ciąg opisujący połączenie do bazy danych przy pomocy sterownika JDBC.
- *connection.user*: użytkownik bazy danych.
- *connection.password*: hasło użytkownika.
- *refresh.countLimit*: minimalna krotność, kwalifikująca źródło do ponownego indeksowania.
- Wpisy konfiguracyjne log4j.

5.2.4. Generator raportów

Jest to program odpowiedzialny za tworzenie raportów dla użytkowników. Raporty są tworzone na podstawie kryteriów podanych przez użytkownika systemu. Program ten przegląda bazę zdefiniowanych raportów i wyznacza te, które powinny zostać wygenerowane/odświeżone. Następnie przegląda bazę w poszukiwaniu ogłoszeń spełniających kryteria podane w definicji raportu. Na tej podstawie dodaje nowe strony do wyników raportu. Użytkownik może je przeglądać za pomocą interfejsu strony web. Plik konfiguracyjny ma nazwę *Reports.properties* i jest plikiem tekstowym, składającym się z par klucz=wartość. Należy ustawić następujące parametry:

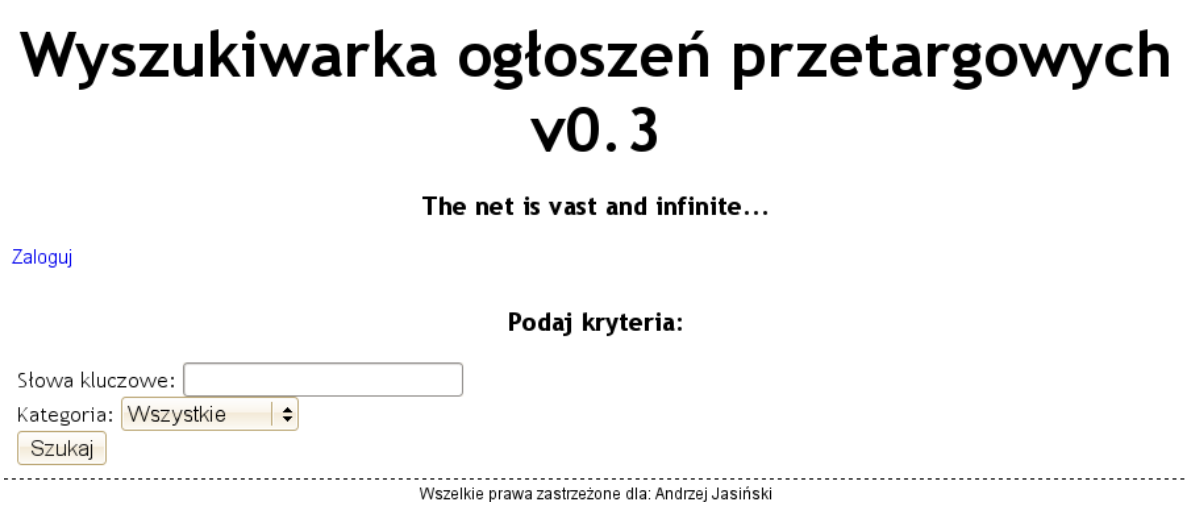
- *connection.hostConfig*: ciąg opisujący połączenie do bazy danych przy pomocy sterownika JDBC.
- *connection.user*: użytkownik bazy danych.
- *connection.password*: hasło użytkownika.
- Wpisy konfiguracyjne log4j.

5.2.5. Aplikacja Web

Z punktu widzenia użytkownika, najważniejszy jest interfejs w postaci strony internetowej, na której można poszukiwać ogłoszeń, według słów kluczowych oraz kategorii. Aplikacja ta jest wykonana w języku PHP przy użyciu sytemu szablonów Smarty. Po zalogowaniu, użytkownik dodatkowo ma możliwość tworzenia raportów. Funkcjonalność ta polega na okresowym sprawdzaniu bazy w poszukiwaniu nowych przetargów, które spełniają określone kryteria (takie jak przy wyszukiwaniu). Dzięki temu, użytkownik nie musi sam przeglądać bazy, zamiast tego sprawdza jedynie czy w zdefiniowanych raportach nie ma nowych wyników. Ponadto istnieje możliwość zasilenia kolejki pająka wynikami z przeglądarki Google. Może być to przydatne, ponieważ dotarcie do wszystkich stron z ogłoszeniami (czyli dostępnych

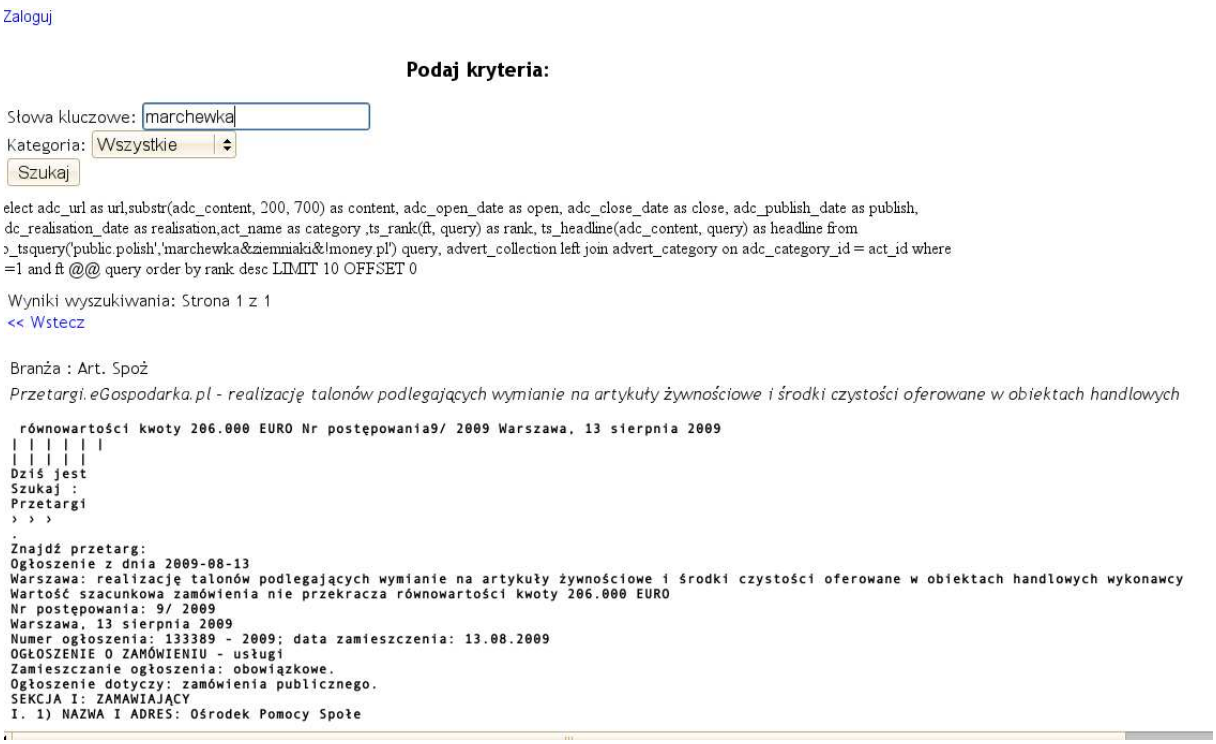
za darmo baz, stron urzędów miast, witryn BIP itp.) może zająć dużo czasu. Dodanie funkcji gromadzenia wyników z wyszukiwarki ogólnej umożliwia zasilenie bazy potencjalnymi przetargami na mniej popularne usługi/produkty. Systemy, takie jak Google, mają w swoich bazach zindeksowaną ogromną liczbę stron (w 2008 roku przekroczona została liczba 1 000 000 000 000 unikalnych adresów URL [6]). Dzięki temu, można poszerzyć indeks o ciekawe strony.

Poniżej zamieszczone zostały przykładowe ekrany wykonanego systemu.



Rysunek 5.1: Głowna strona interfejsu.

Rysunek 5.1 przedstawia główną stronę aplikacji widoczną dla użytkownika niezalogowanego. Ma on tylko możliwość wpisania słów kluczowych do wyszukiwarki. Można stosować operatory logiczne, zgodnie z mechanizmem PostgreSQL FTS. Możliwe są operatory AND(&), OR(|) oraz NOT(!). Należy zwrócić uwagę, że nie stosuje się spacji pomiędzy słowami kluczowymi a operatorem. Tak więc, przykładowe zapytanie może mieć postać "ala&ma&kota". Ekran 5.2 prezentuje wyniki wyszukiwania. Aby

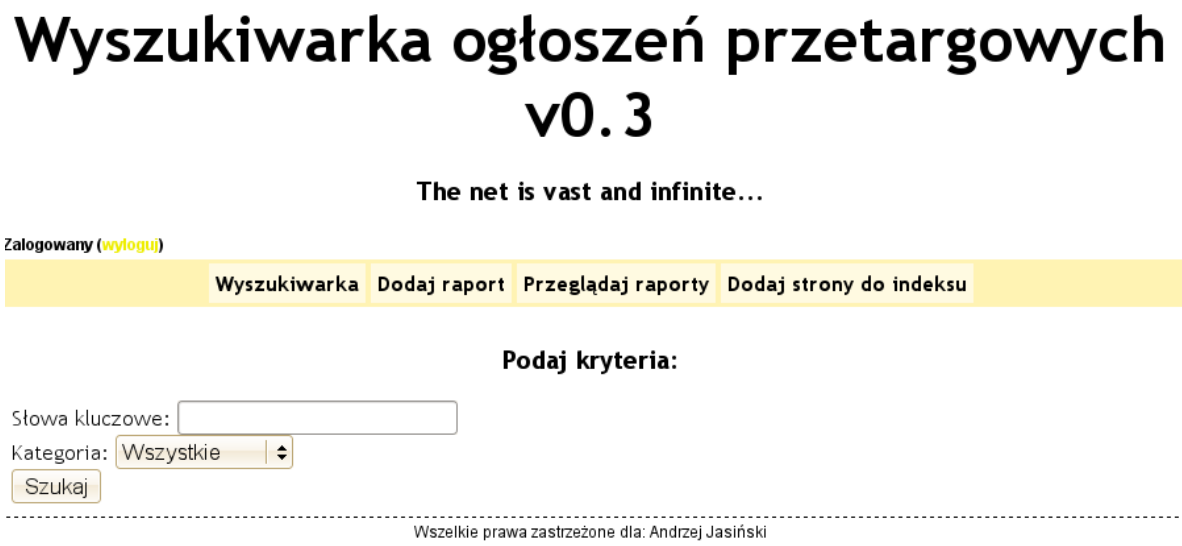


Rysunek 5.2: Wyniki wyszukiwania.

można było szybko stwierdzić, czy jest to w ogóle ogłoszenie oraz czy jest ono wstępnie interesujące, program przedstawia skróconą treść ogłoszenia. Ponadto użytkownik ma możliwość przeczytania całej

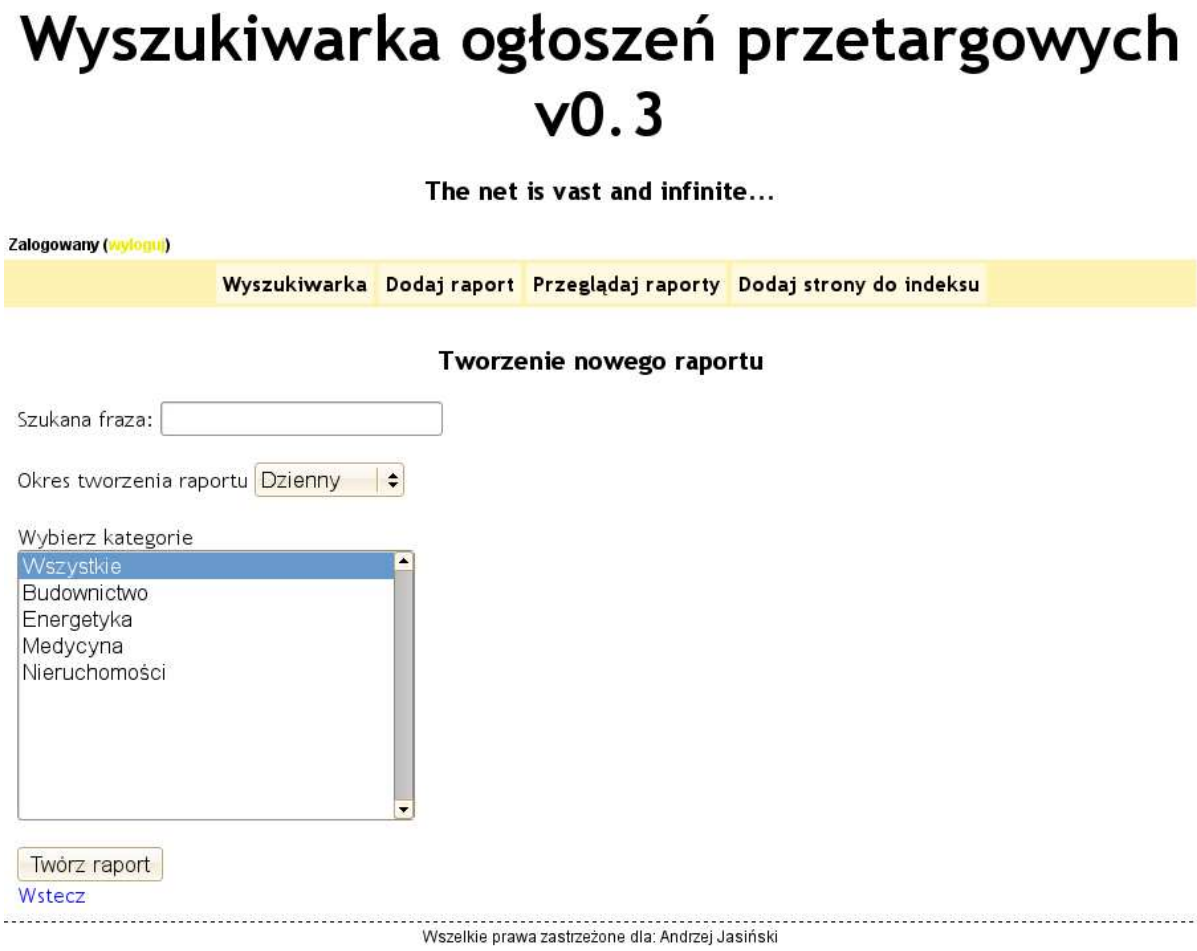
treści przetargu oraz przejścia do oryginalnego dokumentu, za pomocą odnośnika podanego w dolnej części ogłoszenia.

Po zalogowaniu do systemu, użytkownikowi zostaje zaprezentowany ekran jak na zrzucie 5.3. Dostępne jest menu, z którego można przeglądać bazę, zarządzać raportami oraz zasilać system w wyniki z zewnętrznej wyszukiwarki (obecnie Google). Nie został wykonany żaden system zarządzania użytkownikami. W chwili obecnej zdefiniowany jest tylko jeden użytkownik. Tworzenie raportu (obraz 5.4)



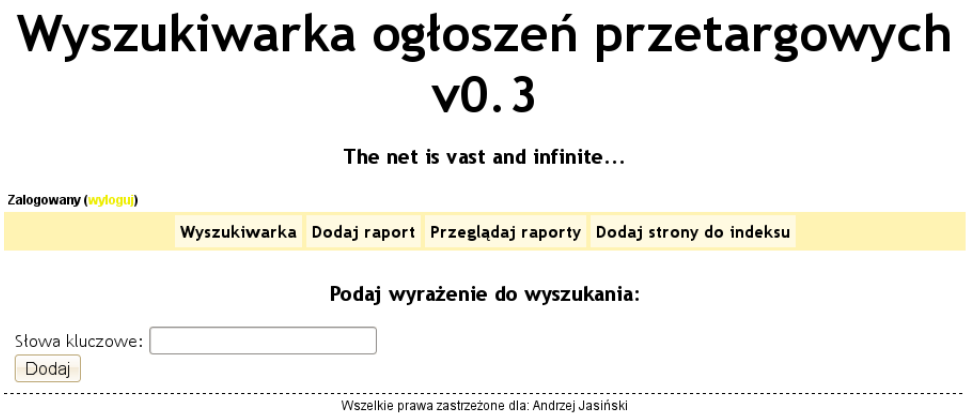
Rysunek 5.3: Głowna strona interfejsu (po zalogowaniu do systemu).

polega na określeniu słów kluczowych zgodnie z zasadami opisanymi powyżej. Dodatkowo możliwe jest wybranie kategorii oraz zdefiniowanie częstości odświeżania raportu. Do wyboru jest generowanie dzienne, tygodniowe i miesięczne. Obraz 5.5 prezentuje listę zdefiniowanych raportów. Użytkownik ma możliwość przeglądania ogłoszeń dla każdego raportu. Strona z listą wyników ma taką samą postać jak w wypadku wyszukiwania przetargów (rys. 5.2). Dodatkowo można usunąć dowolny z raportów. Rysunek 5.6 prezentuje dodawanie nowych stron do indeksu (czyli do przeglądania przez pająka). Należy wpisać poszukiwaną frazę zgodnie z zasadami wyszukiwarki Google. Następuje wówczas połączenie z systemem Google, zadanie zapytania, a następnie dodanie do bazy wyników (z wyłączeniem stron generowanych przez Google). Adresy URL, które udało się poprawnie dodać, zaznaczone są pogrubioną czcionką. Jeśli URL istnieje już w bazie wówczas nie zostanie dodany, co sygnalizuje normalny format czcionki.



Rysunek 5.4: Definiowanie nowego raportu.

Rysunek 5.5: Przeglądanie zdefiniowanych raportów.



Rysunek 5.6: Dodawanie wyników wyszukiwania z systemu Google(R).

5.2.6. Usługi

WSDL opisujący usługę jest dołączony jako załącznik A.

Plik konfiguracyjny ma nazwę *Service.properties* i jest plikiem, składającym się z par klucz=wartość. Należy ustawić następujące parametry:

- *connection.hostConfig*: ciąg opisujący połączenie do bazy danych przy pomocy sterownika JDBC.
- *connection.user*: użytkownik bazy danych.
- *connection.password*: hasło użytkownika.
- Wpisy konfiguracyjne log4.

Należy zwrócić uwagę, że plik ten musi być umieszczony w katalogu głównym aplikacji Tomcat.

System zaczyna przeglądanie Internetu od stron, które zostaną zapisane w bazie. Najlepiej, jeśli będą to strony(strona) o tematyce przetargowej. Wówczas system szybciej zacznie zbierać właściwe informacje. Pająk odwiedzając stronę, wstępnie ocenia czy jej treść jest istotna i czy ma zostać poddana dalszej analizie. Jak zostało pokazane w rozdziale 4, ogłoszenia przetargowe można wstępnie odfiltrować.

System potrafi przeglądać zawartość stron z dość dużą prędkością. W ciągu doby jest w stanie zgromadzić bazę rzędu 100 000 ogłoszeń oraz 300 000 unikalnych adresów URL. Oczywiście, niektóre ogłoszenia mogą być powielone, ponieważ niekiedy mogą wystąpić na stronach BIP oraz portalach przetargowych. Niektóre z wyników są fałszywe, gdyż system ma raczej tendencję do "fałszywych" pozytywów niż negatywów. Bardzo trudno jest określić stopień "zaśmiecenia" bazy ze względu na jej duży rozmiar. Z tego względu przedstawione w następnej sekcji wyniki są tylko pewnym przybliżeniem.

5.3. Przykład

W tej sekcji zaprezentowane zostanie działanie systemu dla przykładowej strony z ogłoszeniem. Ma to na celu zaprezentowanie, w bardziej obrazowej formie, sposobu działania przeglądarki.

System trafia jedną z podstron Wojewódzkiego Szpitala Podkarpackiego (*http://www.krosno.med.pl/przetargi/przetargi.php?id=21*). Fragment tej strony znajduje się na rysunku 5.7. Pająk sprawdza, czy na serwerze znajduje się plik "robots.txt". Nie odnajduje go, więc wstępnie uznaje, że stronę można pobrać. Po wykonaniu tej czynności sprawdzane są jeszcze nagłówki HTML. Również w nich nie ma ograniczeń dla pająka, dlatego możliwa jest dalsza analiza. Następuje wyodrębnienie tekstu (rysunek 5.8 i jego ocena.

Program odnajduje frazę "przetarg" i dodaje do wyniku jej wagę. Ponadto na stronie znajdują się również inne frazy określone w słowniku. Ostateczna ocena wynosi aż 16. Dokument jest klasyfiko-

wany jako ogłoszenie i zostanie przekazany do dalszej analizy. Widać tu trudność poprawnej klasyfikacji dokumentów. W następnej kolejności wyodrębnione zostają odnośniki. Ponieważ strona jest sklasyfikowana jako przetarg, wszystkie mają wagę 1. Wśród linków znajduje się wskazujący na stronę *http://www.krosno.med.pl/przetargi/przetarg_detal.php?id=278* (rysunek 5.9). Zawiera on w sobie frazę "przetarg" więc jego waga będzie wynosić 1.

Po ocenie wszystkich odnośników system doda do kolejki Analizatora badaną stronę oraz wszystkie odnośniki do dalszego przeglądania. Po pewnym czasie dotrze do wyróżnionego w tekście ogłoszenia i postąpi podobnie jak zostało już opisane.

Analizator zażąda od usługi sieciowej stron do analizy. W odpowiedzi zwrócony zostanie dokument z ogłoszeniem przetargowym 5.3 (rysunek 5.10). Postać tekstowa tego ogłoszenia znajduje się na rysunku 5.11. Pierwszym krokiem jest wykonanie kategoryzacji za pomocą klasyfikatora Bayesa. Wynik jest pozytywny więc waga dokumentu zostaje zwiększona o 1. Kolejnym krokiem jest porównanie histogramu słów (po stemmingu) ze słownikami branż. W tym wypadku wybrane zostało "Budownictwo" (6 trafień). Po ustaleniu branży program wyszukuje w tekście wzorców dat. Pierwszą jest "2009-03-10". Przeglądane jest jej otoczenie *NZ/215/23/2009 Termomodernizacja budynków Wojewódzkiego Szpitala Podkarpackiego im. Jana Pawła II w Krośnie Numer: NZ/215/23/2009 Data składania:* . Ciąg znaków jest dzielony (separatorem jest biały znak) i wykonywany jest stemming. Tablica ta jest następnie porównywana za pomocą metryki Q-Gram z listą słów kluczowych. Nie znaleziono żadnego podobnego słowa, więc analizowana jest następna data. Jej otoczenie to: *2009-03-10 Data otwarcia:*. W tym wypadku pasuje słowo "otwarcia", więc program klasyfikuje tę datę jako otwarcia.

Ostatecznie w dokumencie zostają znalezione następujące daty:

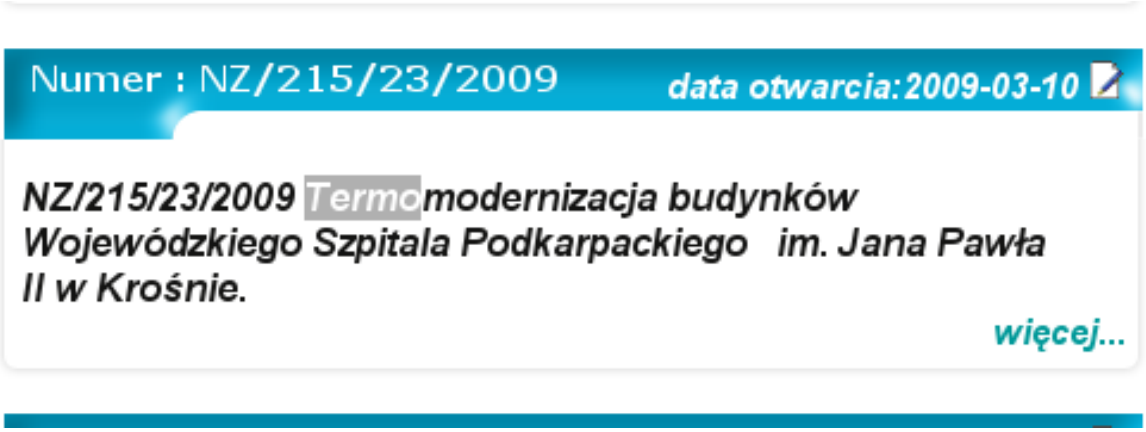
- otwarcia: 2009-03-10 ,
- publikacji: 2009-02-13,
- realizacji: 2010-06-30.

Ranga dokumentu zostaje ustalona na 4.0 (odnalezione daty, Bayes, branża). Analizator zapisuje dokument oraz wszystkie zebrane informacje do bazy i analizuje kolejne ogłoszenie.

Rysunek 5.7: Fragment strony, na której znajdują się odnośniki do przetargów.

```
wojewódzki szpital podkarpacki im. jana pawła ii w krośnie
szpital
iso9001
mikrobiologia
akredytacja
zporr
przetargi archiwalne
zawiera przetargi których data otwarcia już upłynęła
numer : konkurs ofertdata otwarcia:2008-01-11
konkurs ofert w celu udzielenia zamówienia na wykonywanie zadań publicznego zakładu opieki
zdrowotnej podmiotom, o których mowa w art. 35 ustawy o z.o.z.
numer : konkurs ofert 2data otwarcia:2008-01-14
konkurs ofert w celu udzielenia zamówienia na wykonywanie zadań publicznego zakładu opieki
zdrowotnej
numer : nz/215/4/2008data otwarcia:2008-01-17
nz/215/4/2008 zakup i dostawa materiałów zuzywalnych do badań urodynamicznych.
numer : nz/214/1/2008data otwarcia:2008-01-17
nz/214/1/2008 zapytanie ofertowe
numer : nz/215/1/2008data otwarcia:2008-01-23
nz/215/1/2008 zakup i dostwa leków objętych programem lekowym
numer : nz/215/3/2008data otwarcia:2008-01-23
nz/215/3/2008 zakup i dostawa leków
numer : nz/215/2/2008data otwarcia:2008-01-25
nz/215/2/2008 zakup i dostawa materiałów opatrunkowych.
numer : nz/215/9/2008data otwarcia:2008-01-29
zakup i dostawa defibrylatora
numer : nz/215/6/2008data otwarcia:2008-02-05
nz /215/6/2008 zakup i dostawa elektrod do litotryptera i cewników do hemodializy
numer : nz/215/7/2008data otwarcia:2008-02-13
nz/215/7/2008 zakup i dostawa sprzętu ratowniczo-medycznego
numer : nz/215/11/2008data otwarcia:2008-02-22
nz/215/11/2007 zakup i dostawa odczynników do badań pilnych i hormonów płciowych do aparatu mini
vidas oraz usługę serwisową aparatu
numer : nz/215/13/2008data otwarcia:2008-02-22
nz/215/13/2008 zakup i dostawa jednorazowych światłowodów do lasera zielonego greenlight.
numer : nz/215/15/2008data otwarcia:2008-02-22
nz/215/8/2008 zakup i dostawa defibrylatora, respiratora, kardiomonitora i pulsoksymetru
numer : nz/215/10/2008data otwarcia:2008-02-22
przetarg nieograniczony na usługę konserwacji wind- unieważniony
numer : nz/215/16/2008data otwarcia:2008-02-28
nz/215/16/2008 dostawa aparatu rtg z torem wizyjnym - system cyfrowy. unieważniony
numer : nz/215/17/2008data otwarcia:2008-02-28
```

Rysunek 5.8: Fragment strony, na której znajdują się odnośniki do przetargów (po wyodrębnieniu tekstu).



Rysunek 5.9: Jeden z odnalezionych odnośników.

Rysunek 5.10: Fragment strony z ogłoszeniem przetargowym.

```
Wojewódzki Szpital Podkarpacki im. Jana Pawła II w Krośnie
Szpital
ISO9001
Mikrobiologia
Akredytacja
ZPORR
Informacja skrócona
NZ/215/23/2009 Termomodernizacja budynków Wojewódzkiego Szpitala Podkarpackiego im. Jana Pawła II w
Krośnie.<!
Numer : NZ/215/23/2009
Data składania: 2009-03-10
Data otwarcia: 2009-03-10
Krosno: Nz/215/23/2009 Termomodernizacja budynków Wojewódzkiego Szpitala Podkarpackiego im. Jana
Pawła II W Krośnie
Numer ogłoszenia: 37769 - 2009; data zamieszczenia: 13.02.2009
OGŁOSZENIE O ZAMÓWIENIU - roboty budowlane
Zamieszczanie ogłoszenia: obowiązkowe.
Ogłoszenie dotyczy: zamówienia publicznego.
SEKCJA I: ZAMAWIAJĄCY
I. 1) NAZWA I ADRES: Wojewódzki Szpital Podkarpacki im. Jana Pawła II , ul. Korczyńska 57, 38-400
Krosno, woj. podkarpackie, tel. 013 4378497, 4378215, faks 013 4378497, 4378215.
Adres strony internetowej zamawiającego: www.krosno.med.pl
I. 2) RODZAJ ZAMAWIAJĄCEGO: Podmiot prawa publicznego.
SEKCJA II: PRZEDMIOT ZAMÓWIENIA
II.1) OKREŚLENIE PRZEDMIOTU ZAMÓWIENIA
II.1.1) Nazwa nadana zamówieniu przez zamawiającego: Nz/215/23/2009 Termomodernizacja budynków
Wojewódzkiego Szpitala Podkarpackiego im. Jana Pawła II W Krośnie.
II.1.2) Rodzaj zamówienia: roboty budowlane.
II.1.3) Określenie przedmiotu oraz wielkości lub zakresu zamówienia: Przedmiotem zamówienia są
roboty budowlane w zakresie docieplenia ścian zewnętrznych i stropodachów oraz wymiany stolarki
okiennej i drzwi zewnętrznych oraz bram garażowych określone przez: specyfikację techniczną
wykonania i odbioru robót (załącznik 4 ) projekt budowlany i projekty wykonawcze (załącznik 8)
przedmiary robót (załączniki 5a, 6a, 7a), dokumentację projektową do docieplenia ścian zewnętrznych
i stropodachów, wymiany stolarki okiennej i drzwi zewnętrznych oraz bram garażowych ( załącznik nr
9) oraz kosztorysy budowlane ślepe (załączniki 5b, 6b, 7b).
II.1.4) Wspólny Słownik Zamówień (CPV): 45.00.00.00-7, 44.22.11.00-6.
II.1.5) Czy dopuszcza się złożenie oferty częściowej: nie.
II.1.6) Czy dopuszcza się złożenie oferty wariantowej: nie.
II.1.7) Czy przewiduje się udzielenie zamówień uzupełniających: nie.
II.2) CZAS TRWANIA ZAMÓWIENIA LUB TERMIN WYKONANIA: Zakończenie: 30.06.2010.
(...)
```

Rysunek 5.11: Fragment strony z ogłoszeniem przetargowym (postać tekstowa).

5.4. Testy

System podczas testu, trwającego około tygodnia zgromadził niemal 679 620 dokumentów sklasyfikowanych jako ogłoszenia przetargowe. Wobec takiej ilości, trudno jest ocenić ogólną skuteczność systemu.

Wyniki dla poszczególnych branż zdefiniowanych w systemie są zaprezentowane w tabeli 5.1.

Kategoria	**Ilość ogłoszeń**
Art.Spoż	2530
Automaty	11 150
Budownictwo	42590
Energetyka	36780
Medycyna	23770
Nieruchomości	42490
Vending	110
Pozostałe	520 200

Tablica 5.1: Ilość dokumentów z podziałem na kategorie.

Zaobserwowano, że bardzo dużo ogłoszeń nie ma przypisanej kategorii. Aby ocenić skuteczność działania systemu została sprawdzona poprawność klasyfikacji dla pierwszych 10 wyników z każdej kategorii (tabela 5.2).

Kategoria	**Błędnych**
Art.Spoż	5
Automaty	5
Budownictwo	0
Energetyka	3
Medycyna	1
Nieruchomości	5
Vending	10*

Tablica 5.2: Ilość dokumentów błędnie sklasyfikowanych (na pierwszych 10 odnalezionych)

Trudność klasyfikacji dokumentów polega na tym, że ogłoszenia przetargowe obejmują czasami swoim zakresem wiele branż. Gwiazdka przy "Vending" oznacza, że ogłoszeń na dostawę maszyn vendignowych nie znaleziono. System odnalazł przetargi na przeprowadzenie szkoleń. Jednym z wymagań przetargu było zapewnienie uczestnikom szkolenia dostępu do ekspresów do kawy. Wobec braku katego-

rii “Szkolenia” czy “Edukacja” najbliższą kategorią był właśnie “Vending”. W pozostałych wypadkach, zwłaszcza ”Energetyki”, błędne wyniki spowodowane były odnalezieniem ogłoszeń z branży budowlanej, gdzie istotnym elementem zamówienia było wykonanie/modernizacja instalacji elektrycznej.

System bezbłędnie odnalazł ogłoszenia z zakresu budownictwa i medycyny. W przypadku budownictwa wpływ ma bardzo duża łączna ilość ogłoszeń tej kategorii. W przypadku medycyny istotna jest najprawdopodobniej unikalność słów kluczowych oraz fakt, że nie są wykorzystywane w innych branżach.

W wyszukiwarce Google ciężko jest odnaleźć ogłoszenia dla całej branży. Po wpisaniu np. ”przetarg energetyka“ wśród pierwszych 10 wyników nie odnajdziemy aktualnych ogłoszeń o przetargach w tej kategorii. Dopiero podanie konkretnych słów kluczowych pozwala znaleźć istotne dokumenty. Po wpisaniu ”przetarg dostawa energii elektrycznej“ odnajdziemy 6 ogłoszeń przetargowych, 1 link sponsorowany oraz informację o procedurze ogłaszania przetargów na dostawę energii elektrycznej. Wyszukiwarka, wykonana w ramach pracy magisterskiej, znalazła około 150 ogłoszeń. Pierwsze 10 było prawidłowymi ogłoszeniami.

W tabeli 5.3 przedstawione są wyniki wyszukiwania trzech haseł w wykonanej przeglądarce, wyszukiwarce Google oraz przy użyciu portalu e-gospodarka.pl oraz money.pl (wartości przybliżone). Sprawdzowa została również poprawność pierwszych 10 wyników (tabela 5.4).

Hasło	Rozw. własne	egospodarka	Google	money.pl**
automat napój	50	0	22 100	0
dostawa leków	7290	8 179	77 600	ponad 1 500
dostawa energii elektrycznej	3960	2 709	57 000	ponad 1 500
marchewka ziemniaki	100	67	6	45

Tablica 5.3: Wyniki wyszukania dla poszczególnych haseł.

Hasło	Rozw. własne	egospodarka	Google	money.pl
automat napój	7	nd.	5	nd.
dostawa leków	9*	10	10	8
dostawa energii elektrycznej	10	10	6	7
marchewka ziemniaki	10	10	7	10

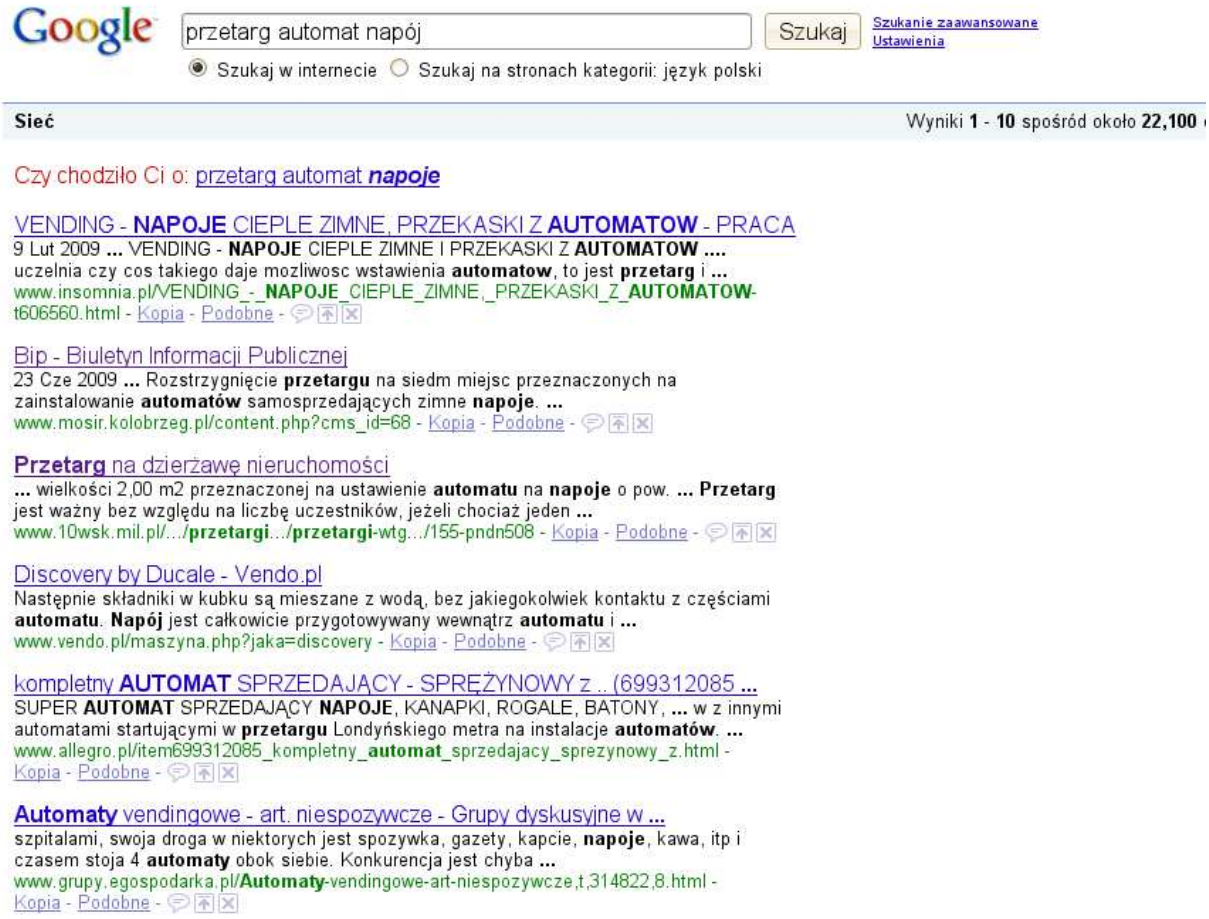
Tablica 5.4: Poprawność pierwszych 10 wyników.

** - jedno ogłoszenie było stroną zawierająca listę wszystkich ogłoszonych przetargów*

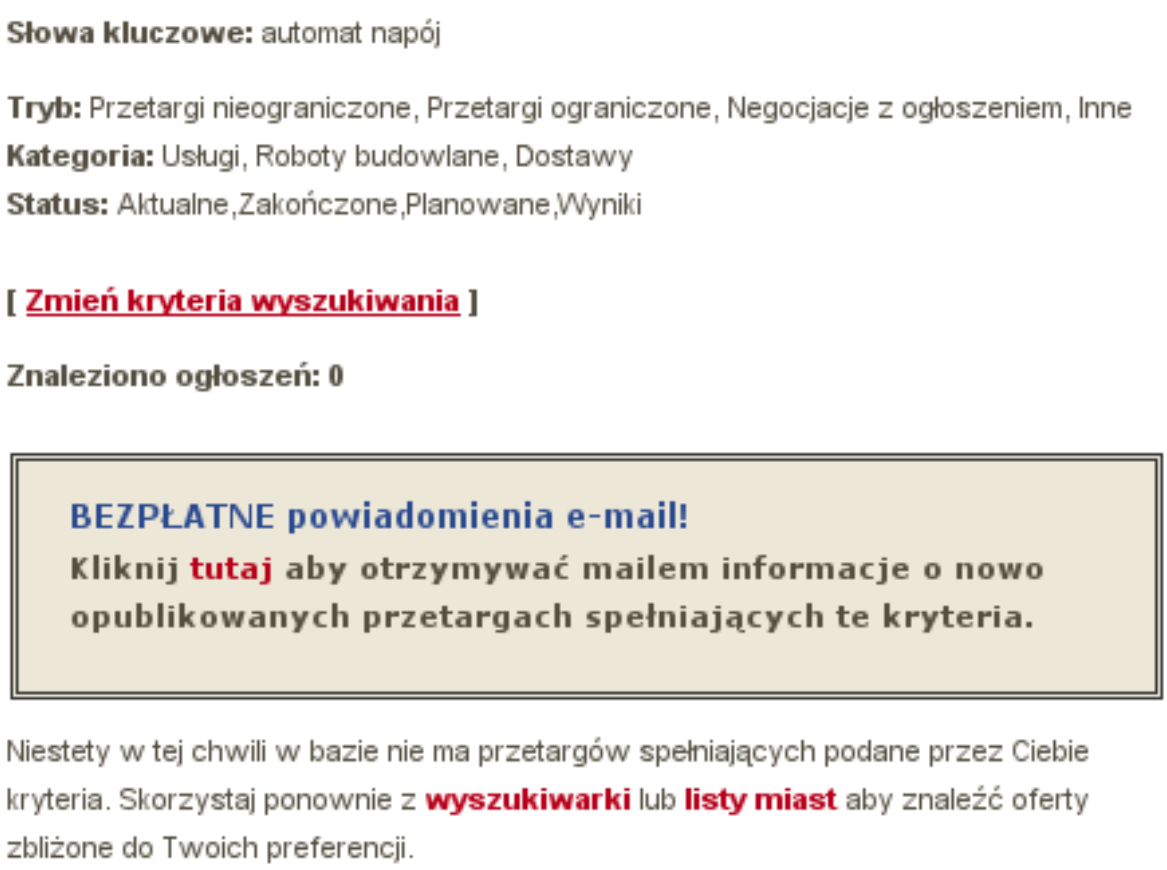
*** - wyszukiwarka money.pl nie podaje liczby odnalezionych ogłoszeń, więc sprawdzono tylko czy jest ponad 100 stron z ogłoszeniami(na każdej jest ich 15)*

Dokonywanie sprawdzenia poprawności tylko pierwszych 10 wyników ma sens, ponieważ zazwyczaj użytkownicy takich systemów mają tendencję do oglądania tylko pierwszej strony. Wytrwalsi docierają do 2-3 podstron. Dla wyszukiwarki Google do każdego hasła było dodawane słowo ”przetarg“ w celu odnalezienia tylko stron o tematyce przetargowej.

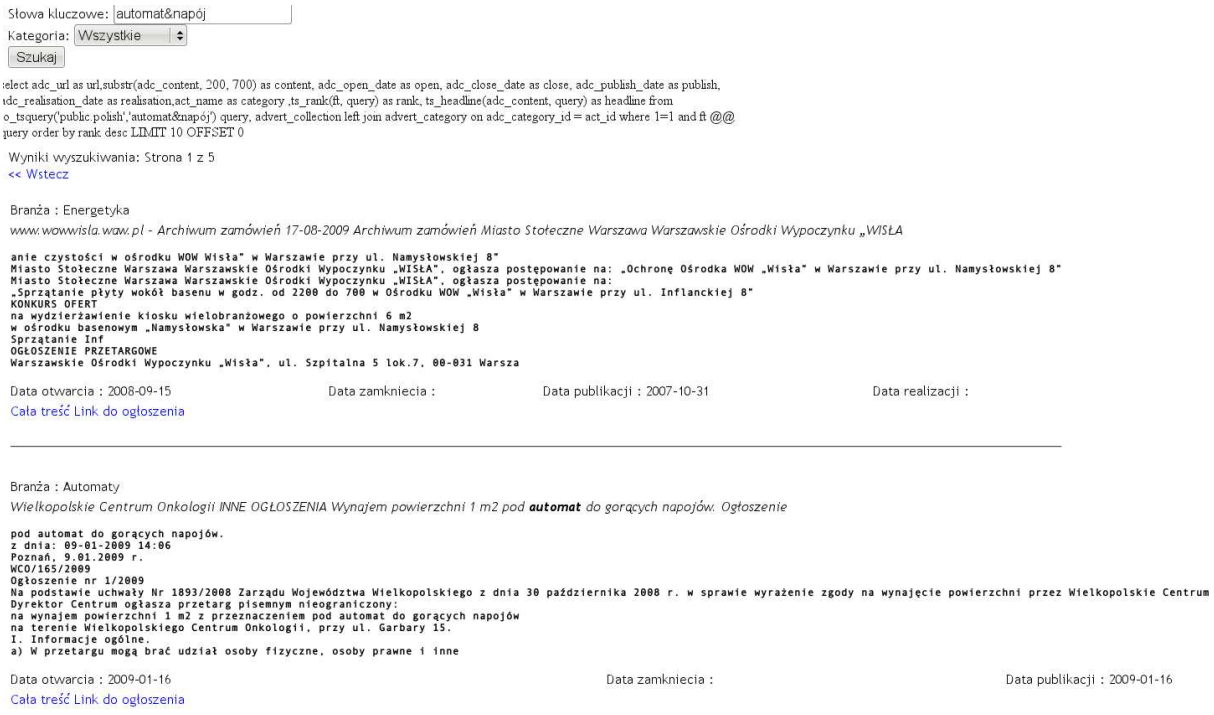
Poniżej wykonano przykładowe zrzuty ekranu z analizowanych systemów, prezentujące wyniki wyszukiwania.



Rysunek 5.12: Wynik wyszukiwania hasła ”przetarg automat napój“ w wyszukiwarce Google.



Rysunek 5.13: Wynik wyszukiwania hasła ”automat napój“ w wyszukiwarce egospodarka.pl.



Rysunek 5.14: Wynik wyszukiwania hasła ”automat&napój“ w wyszukiwarce własnej.

6. Wnioski

Celem tej pracy było przeanalizowanie możliwości automatycznego poszukiwania w Internecie informacji o wszelkiego rodzaju ogłoszeniach przetargowych. Zadanie to jest bardzo skomplikowane i porusza wiele zagadnień z tematyki informatyki, jak i lingwistyki. Postawiony problem jest dość specyficzny, stąd konieczność poszukiwań wielu rozwiązań na własną rękę. Ostatecznie wykonano złożony system składający się z kilku funkcjonalnych części:

- pająk,
- analizator,
- serwer (baza danych, usługi sieciowe, interfejsy WWW, aplikacje pomocnicze).

Jest to kompletny system do wyszukiwania informacji o ogłoszeniach przetargowych. Pozwala on skutecznie odnajdywać ogłoszenia przetargowe, z wielu branż. Dodatkowo, ma praktycznie nieograniczony zakres działania, gdyż nie koncentruje się na określonych z góry serwisach/stronach lecz swobodnie przegląda zasoby Internetu. Pomimo stosunkowo prostych algorytmów i technik jego skuteczność można nazwać zadowalającą.

Stworzone rozwiązanie łączy zalety przeglądarek ogólnych oraz wyspecjalizowanych serwisów. Dzięki zasilaniu wynikami z wyszukiwarek o dużym zasięgu można docierać do mało popularnych ogłoszeń. Dzięki temu można uzyskać przewagę nad płatnymi portalami, w których można nie znaleźć interesujących ogłoszeń. Podczas testów zrealizowany program odnalazł ogłoszenia z zakresu "Vendingu", których jest stosunkowo mało w porównaniu z innymi branżami (np. budownictwem). Ponadto dla słowa kluczowego "automat napój" wyszukiwarki portali egospodarka i money.pl nie zwróciły żadnego wyniku. Największa ilość dokumentów została znaleziona w systemie Google. Jednak już na pierwsze 10 wyników, 5 było niewłaściwych. Widać więc, że wymaga to od użytkownika poświęcenia czasu na filtrację stron. Połączenie zalet Google oraz systemu dedykowanego daje ciekawe rezultaty w postaci automatycznej klasyfikacji i oceny oraz szerszego zakresu działania.

System działa automatycznie, wymagając jedynie pierwszej strony, od której rozpoczęte zostanie przeglądanie Internetu. Dzięki temu zbiera dane o przetargach praktycznie bez ingerencji użytkownika

(wyjątkiem jest dodawanie stron do indeksowania z wyników wyszukiwarki Google). Trudno jest jednak osiągnąć pełną kompletność zebranych informacji, ze względu na nieprzewidywalne wejście programu. Pod tym względem system ustępuje komercyjnym portalom. Zawarte w nich informacje są zawsze dokładne i pełne. Wynika to z faktu, że analizują z góry określone strony, co ułatwia parsowanie.

Podstawowym zastosowaniem wykonanego projektu jest wyszukiwanie zamówień. Można również odszukiwać w nim strony, na których są ciekawe ogłoszenia. Jest to szczególnie interesujące jeśli chce się stworzyć bądź rozwinąć system do gromadzenia informacji o przetargach. W takim wypadku wykonany w ramach tej pracy program może wskazywać źródła informacji o zamówieniach oraz pojedyncze, ale interesujące przetargi.

W chwili obecnej system ma jeszcze parę niezaimplementowanych funkcji, pod które przygotowana jest struktura bazy danych. Można go rozbudować o wyciąganie informacji o typie przetargu (ograniczony/nieograniczony) oraz jego wartości.

Jednocześnie okazało się, że trudno jest bezbłędnie odróżnić ogłoszenie od strony związanej z tematyką przetargową. W tym wypadku należałoby dokonać analizy semantycznej dokumentu, czyli zrozumieć jego sens. Można również rozważyć możliwość dodawania stron do “czarnej listy” tzn, stron/serwisów, które nie powinny być przeglądane. Ponadto usługa rozdzielająca zadania dla pająków może dokonywać rozpraszania ruchu, czyli nie koncentrować się na jednym portalu. W chwili obecnej może się tak zdarzyć jeśli pająk wejdzie na portal z ogłoszeniami przetargowymi. Są to jednak już dodatkowe możliwości, wychodzące poza podstawę tej pracy.

A. Dodatek A

```
1 <?xml version="1.0" encoding="UTF-8"?>
  <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:ns1="http://
      org.apache.axis2/xsd" xmlns:ns="http://server.tuxpowered.net" xmlns:wsaw="http:
      //www.w3.org/2006/05/addressing/wsdl" xmlns:http="http://schemas.xmlsoap.org/
      wsdl/http/" xmlns:ax21="http://structures.tuxpowered.net/xsd" xmlns:xs="http://
      www.w3.org/2001/XMLSchema" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://schemas.
      xmlsoap.org/wsdl/soap12/" targetNamespace="http://server.tuxpowered.net">
3     <wsdl:documentation>
      Serwis odpowiedzialny za crawling
5   </wsdl:documentation>
      <wsdl:types>
7         <xs:schema xmlns:ax22="http://structures.tuxpowered.net/xsd"
              attributeFormDefault="qualified" elementFormDefault="qualified"
              targetNamespace="http://server.tuxpowered.net">
              <xs:import namespace="http://structures.tuxpowered.net/xsd"/>
9             <xs:element name="RetrieveContentResponse">
                  <xs:complexType>
11                    <xs:sequence>
                          <xs:element maxOccurs="unbounded" minOccurs="0" name="return
                              " nillable="true" type="xs:string"/>
13                    </xs:sequence>
                  </xs:complexType>
15            </xs:element>
              <xs:element name="PutContent">
17                <xs:complexType>
                      <xs:sequence>
19                        <xs:element maxOccurs="unbounded" minOccurs="0" name="
                              a_arrContent" nillable="true" type="ax22:SiteContent"/>
                      </xs:sequence>
21                </xs:complexType>
              </xs:element>
23            <xs:element name="PutContentResponse">
```

```
            <xs:complexType>
25              <xs:sequence>
                    <xs:element minOccurs="0" name="return" nillable="true" type
                        ="xs:string"/>
27              </xs:sequence>
            </xs:complexType>
29      </xs:element>
        <xs:element name="PutAdvertContent">
31          <xs:complexType>
                <xs:sequence>
33                  <xs:element maxOccurs="unbounded" minOccurs="0" name="
                        adverts" nillable="true" type="ax22:AdvertContent"/>
                </xs:sequence>
35          </xs:complexType>
        </xs:element>
37      <xs:element name="PutAdvertContentResponse">
            <xs:complexType>
39              <xs:sequence>
                    <xs:element minOccurs="0" name="return" nillable="true" type
                        ="xs:string"/>
41              </xs:sequence>
            </xs:complexType>
43      </xs:element>
        <xs:element name="GetSitesToAnalyzeResponse">
45          <xs:complexType>
                <xs:sequence>
47                  <xs:element maxOccurs="unbounded" minOccurs="0" name="return
                        " nillable="true" type="ax22:SiteContent"/>
                </xs:sequence>
49          </xs:complexType>
        </xs:element>
51  </xs:schema>
    <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified"
        targetNamespace="http://structures.tuxpowered.net/xsd">
53      <xs:complexType name="SiteContent">
            <xs:sequence>
55              <xs:element minOccurs="0" name="URL" nillable="true" type="
                    xs:string"/>
                <xs:element minOccurs="0" name="content" nillable="true" type="
                    xs:string"/>
57              <xs:element maxOccurs="unbounded" minOccurs="0" name="links"
                    nillable="true" type="ax21:SiteLink"/>
```

```
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="SiteLink">
            <xs:sequence>
                <xs:element minOccurs="0" name="link" nillable="true" type="
                    xs:string"/>
                <xs:element minOccurs="0" name="priority" nillable="true" type="
                    xs:int"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="AdvertContent">
            <xs:sequence>
                <xs:element minOccurs="0" name="URL" nillable="true" type="
                    xs:string"/>
                <xs:element minOccurs="0" name="amount" type="xs:double"/>
                <xs:element minOccurs="0" name="category" type="xs:int"/>
                <xs:element minOccurs="0" name="closeDate" nillable="true" type=
                    "xs:string"/>
                <xs:element minOccurs="0" name="content" nillable="true" type="
                    xs:string"/>
                <xs:element minOccurs="0" name="openDate" nillable="true" type="
                    xs:string"/>
                <xs:element minOccurs="0" name="publishDate" nillable="true"
                    type="xs:string"/>
                <xs:element minOccurs="0" name="realisationDate" nillable="true"
                     type="xs:string"/>
                <xs:element minOccurs="0" name="title" nillable="true" type="
                    xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:schema>
</wsdl:types>
<wsdl:message name="GetSitesToAnalyzeRequest"/>
<wsdl:message name="GetSitesToAnalyzeResponse">
    <wsdl:part name="parameters" element="ns:GetSitesToAnalyzeResponse"/>
</wsdl:message>
<wsdl:message name="PutContentRequest">
    <wsdl:part name="parameters" element="ns:PutContent"/>
</wsdl:message>
<wsdl:message name="PutContentResponse">
    <wsdl:part name="parameters" element="ns:PutContentResponse"/>
</wsdl:message>
```

```
91      <wsdl:message name="RetrieveContentRequest"/>
        <wsdl:message name="RetrieveContentResponse">
93          <wsdl:part name="parameters" element="ns:RetrieveContentResponse"/>
        </wsdl:message>
95      <wsdl:message name="PutAdvertContentRequest">
            <wsdl:part name="parameters" element="ns:PutAdvertContent"/>
97      </wsdl:message>
        <wsdl:message name="PutAdvertContentResponse">
99          <wsdl:part name="parameters" element="ns:PutAdvertContentResponse"/>
        </wsdl:message>
101     <wsdl:portType name="ManagerServicePortType">
            <wsdl:operation name="GetSitesToAnalyze">
103             <wsdl:input message="ns:GetSitesToAnalyzeRequest" wsaw:Action="
                    urn:GetSitesToAnalyze"/>
                <wsdl:output message="ns:GetSitesToAnalyzeResponse" wsaw:Action="
                    urn:GetSitesToAnalyzeResponse"/>
105         </wsdl:operation>
            <wsdl:operation name="PutContent">
107             <wsdl:input message="ns:PutContentRequest" wsaw:Action="urn:PutContent"/
                    >
                <wsdl:output message="ns:PutContentResponse" wsaw:Action="
                    urn:PutContentResponse"/>
109         </wsdl:operation>
            <wsdl:operation name="RetrieveContent">
111             <wsdl:input message="ns:RetrieveContentRequest" wsaw:Action="
                    urn:RetrieveContent"/>
                <wsdl:output message="ns:RetrieveContentResponse" wsaw:Action="
                    urn:RetrieveContentResponse"/>
113         </wsdl:operation>
            <wsdl:operation name="PutAdvertContent">
115             <wsdl:input message="ns:PutAdvertContentRequest" wsaw:Action="
                    urn:PutAdvertContent"/>
                <wsdl:output message="ns:PutAdvertContentResponse" wsaw:Action="
                    urn:PutAdvertContentResponse"/>
117         </wsdl:operation>
        </wsdl:portType>
119     <wsdl:binding name="ManagerServiceSoap11Binding" type="ns:ManagerServicePortType
            ">
            <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="
                document"/>
121         <wsdl:operation name="GetSitesToAnalyze">
                <soap:operation soapAction="urn:GetSitesToAnalyze" style="document"/>
```

```
123             <wsdl:input>
                    <soap:body use="literal"/>
125             </wsdl:input>
                <wsdl:output>
127                 <soap:body use="literal"/>
                </wsdl:output>
129         </wsdl:operation>
            <wsdl:operation name="PutContent">
131             <soap:operation soapAction="urn:PutContent" style="document"/>
                <wsdl:input>
133                 <soap:body use="literal"/>
                </wsdl:input>
135             <wsdl:output>
                    <soap:body use="literal"/>
137             </wsdl:output>
            </wsdl:operation>
139         <wsdl:operation name="RetrieveContent">
                <soap:operation soapAction="urn:RetrieveContent" style="document"/>
141             <wsdl:input>
                    <soap:body use="literal"/>
143             </wsdl:input>
                <wsdl:output>
145                 <soap:body use="literal"/>
                </wsdl:output>
147         </wsdl:operation>
            <wsdl:operation name="PutAdvertContent">
149             <soap:operation soapAction="urn:PutAdvertContent" style="document"/>
                <wsdl:input>
151                 <soap:body use="literal"/>
                </wsdl:input>
153             <wsdl:output>
                    <soap:body use="literal"/>
155             </wsdl:output>
            </wsdl:operation>
157     </wsdl:binding>
        <wsdl:binding name="ManagerServiceSoap12Binding" type="ns:ManagerServicePortType
            ">
159         <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="
                document"/>
            <wsdl:operation name="GetSitesToAnalyze">
161             <soap12:operation soapAction="urn:GetSitesToAnalyze" style="document"/>
                <wsdl:input>
```

```
163                 <soap12:body use="literal"/>
                </wsdl:input>
165             <wsdl:output>
                    <soap12:body use="literal"/>
167             </wsdl:output>
            </wsdl:operation>
169         <wsdl:operation name="PutContent">
                <soap12:operation soapAction="urn:PutContent" style="document"/>
171             <wsdl:input>
                    <soap12:body use="literal"/>
173             </wsdl:input>
                <wsdl:output>
175                 <soap12:body use="literal"/>
                </wsdl:output>
177         </wsdl:operation>
            <wsdl:operation name="RetrieveContent">
179             <soap12:operation soapAction="urn:RetrieveContent" style="document"/>
                <wsdl:input>
181                 <soap12:body use="literal"/>
                </wsdl:input>
183             <wsdl:output>
                    <soap12:body use="literal"/>
185             </wsdl:output>
            </wsdl:operation>
187         <wsdl:operation name="PutAdvertContent">
                <soap12:operation soapAction="urn:PutAdvertContent" style="document"/>
189             <wsdl:input>
                    <soap12:body use="literal"/>
191             </wsdl:input>
                <wsdl:output>
193                 <soap12:body use="literal"/>
                </wsdl:output>
195         </wsdl:operation>
    </wsdl:binding>
197 <wsdl:binding name="ManagerServiceHttpBinding" type="ns:ManagerServicePortType">
        <http:binding verb="POST"/>
199     <wsdl:operation name="GetSitesToAnalyze">
            <http:operation location="ManagerService/GetSitesToAnalyze"/>
201         <wsdl:input>
                <mime:content type="text/xml" part="GetSitesToAnalyze"/>
203         </wsdl:input>
            <wsdl:output>
```

```
205                 <mime:content type="text/xml" part="GetSitesToAnalyze"/>
                </wsdl:output>
207         </wsdl:operation>
            <wsdl:operation name="PutContent">
209             <http:operation location="ManagerService/PutContent"/>
                <wsdl:input>
211                 <mime:content type="text/xml" part="PutContent"/>
                </wsdl:input>
213             <wsdl:output>
                    <mime:content type="text/xml" part="PutContent"/>
215             </wsdl:output>
            </wsdl:operation>
217         <wsdl:operation name="RetrieveContent">
                <http:operation location="ManagerService/RetrieveContent"/>
219             <wsdl:input>
                    <mime:content type="text/xml" part="RetrieveContent"/>
221             </wsdl:input>
                <wsdl:output>
223                 <mime:content type="text/xml" part="RetrieveContent"/>
                </wsdl:output>
225         </wsdl:operation>
            <wsdl:operation name="PutAdvertContent">
227             <http:operation location="ManagerService/PutAdvertContent"/>
                <wsdl:input>
229                 <mime:content type="text/xml" part="PutAdvertContent"/>
                </wsdl:input>
231             <wsdl:output>
                    <mime:content type="text/xml" part="PutAdvertContent"/>
233             </wsdl:output>
            </wsdl:operation>
235     </wsdl:binding>
        <wsdl:service name="ManagerService">
237         <wsdl:port name="ManagerServiceHttpSoap11Endpoint" binding="
                ns:ManagerServiceSoap11Binding">
                <soap:address location="http://callforbid.tuxpowered.net:80/axis2/
                    services/ManagerService.ManagerServiceHttpSoap11Endpoint/"/>
239         </wsdl:port>
            <wsdl:port name="ManagerServiceHttpSoap12Endpoint" binding="
                ns:ManagerServiceSoap12Binding">
241             <soap12:address location="http://callforbid.tuxpowered.net:80/axis2/
                    services/ManagerService.ManagerServiceHttpSoap12Endpoint/"/>
            </wsdl:port>
```

```
243         <wsdl:port name="ManagerServiceHttpEndpoint" binding="
                ns:ManagerServiceHttpBinding">
                <http:address location="http://callforbid.tuxpowered.net:80/axis2/
                    services/ManagerService.ManagerServiceHttpEndpoint/"/>
245         </wsdl:port>
        </wsdl:service>
247 </wsdl:definitions>
```

B. Dodatek B

Zawartość płyty CD:

- Kody źródłowe systemu (strona serwera i kliencka),
- Użyte biblioteki,
- Słowniki zdefiniowanych w pracy branż,
- Strony, na których uczono klasyfikator Bayesa,
- Przykładowe strony HTML zawierające ogłoszenia przetargowe oraz histogramy słów dla każdego dokumentu,

Bibliografia

[1] S. Chapman. Simmetrics (http://www.dcs.shef.ac.uk/ sam/simmetrics.html). Technical report, University of Sheffield, 2009.

[2] Creamsoft. Teoria naiwnego klasyfikatora bayesa (http://forum.creamsoft.com.pl/static_html/smieciarek/bayes_theory.html). Technical report, Creamsoft, 2009.

[3] H. Fischer. Soundex phonetic vs. string-matching (http://www.soundex.com/alternative_qgram.htm). Technical report, Wikipedia, 2009.

[4] A. S. Foundation. Apache lucene overview (http://lucene.apache.org/java/docs/). Technical report, Apache Software Foundation, 2009.

[5] A. S. Foundation. Solr (http://lucene.apache.org/solr/). Technical report, Apache Software Foundation, 2009.

[6] J. A. . N. Haja. Google official blog (http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html). Technical report, Google Inc., 2009.

[7] K. Hemenway. *100 sposobów na tworzenie robotów sieciowych*. Helion, Gliwice, 2004.

[8] http://www.przetargi.info/Detailinfos.htm. Szczegółowe informacje o przetargi.info. Technical report, przetargi.info, 2009.

[9] http://www.przetargi.pl/. Szczegółowe informacje o przetargi.pl. Technical report, przetargi.pl, 2009.

[10] M. Kosmulski. *Reprezentacja dokumentów tekstowych w modelu przestrzeni wektorowej (Praca Magisterska)*. Politechnika Warszawska, Warszawa, 2005.

[11] K. I. K. P.Cz. Sieci neuronowe(http://www.kik.pcz.czest.pl/nn/index.php). Technical report, Katedra Inżynierii Komputerowej P.Cz., 2004.

[12] PostgreSQL. Postgresql 8.3 documentation (http://www.postgresql.org/docs/8.3/static/textsearch-indexes.html). Technical report, PostgreSQL, 2009.

[13] S. Seung. Multilayer perceptrons and backpropagation learning. *http://hebb.mit.edu/courses/9.641/2002/lectures/lecture04.pdf*, 1(1):1–6, 2002.

[14] S. Wiki. Spamassassin wiki (http://wiki.apache.org/spamassassin/perceptron). Technical report, Spamassassin, 2004.

[15] Wikipedia. Algorytm soundex (http://pl.wikipedia.org/wiki/soundex). Technical report, Wikipedia, 2009.

[16] Wikipedia. Temat wyrazu (http://pl.wikipedia.org/wiki/temat_wyrazu). Technical report, Wikipedia, 2009.

[17] Wikipedia. Wyrażenia regularne (http://pl.wikipedia.org/wiki/wyra Technical report, Wikipedia, 2009.

[18] Łukasz Łażewski Mariusz Pikuła Adam Siemion Michał Szklarzewski. *Klasyfikacja dokumentów tekstowych (Praca Magisterska)*. Politechnika Warszawska, Warszawa, 2005.