Krzysztof Drozdowski, Adam Pelikant
Wyższa Szkoła Informatyki, Katedra Inżynierskich
Zastosowań Informatyki, 93-008 Łódź, ul Rzgowska 17a
email: dzord@klub.chip.pl,
apelikan@wsinf.edu.pl

TWORZENIE CHATTERBOTA CHRIS

Streszczenie – Rozwój metod sztucznej inteligencji i poszukiwania nowych oraz usprawnianie istniejących kanałów kontaktów z klientami zaowocowały zainteresowaniem programami symulującymi kompetencje komunikacyjne, w tym językowe człowieka. Obecnie programy tego typu rozwinęły się na tyle, że znalazły komercyjne zastosowania również w Polsce np. Adam, Hubert itd. W pracy przedstawiono chatterbota Chris opartego na metodzie wyszukiwania wzorców w wypowiedzi użytkownika. Prezentowane rozwiązanie jest oparte na mechanizmach wypracowanych w ramach projektu Alice fundacji A.L.I.C.E. AI. Wybór istniejącego szablonu rozwiązania pozwala skupić się na tym co w chatterbocie najważniejsze, czyli na jego bazie wiedzy. W ramach pracy przedstawiono wnioski i wskazówki, które pomogą zainteresowanym osobom tworzyć bazy wiedzy dla swoich wersji chatterbotów. Chatterbot Chris jest oparty na servletach Java, a jego wydajność jest na tyle duża, że nie stanowi elementu krytycznego.

1 Informacje wstępne

Systemy z implementacją funkcji konwersacyjnych często noszą nazwy: chatterbotów, lingubotów, agentów, wirtualnych asystentów, awatarów. Pełnione prze nie funkcje mogą być znacznie szersze niż sam proces komunikacji, chociaż zawsze będzie on stanowić ich główny, lub jeden z głównych modułów.

Samo porozumiewanie się, ze względu na łatwość implementacji ogranicza się do postaci tekstu pisanego. Proces komunikacji przypomina rozmowy prowadzone w ramach grup, forów dyskusyjnych, systemów typu IRC.

Wiele z takich programów zostaje wyposażonych dodatkowo w złożony graficzny interfejs. Prowadząc rozmowę widzimy na ekranie obraz człowieka, albo animację postaci niekoniecznie humanoidalnej. Obrazy zmieniają się, ich wygląd zostaje dostosowywany do treści rozmowy. Dzięki temu są one w stanie wyrażać emocję. Człowiekowi łatwiej jest wówczas rozmawiać. Nasze zmysły łatwiej dokonują personalizacji takiego programu i często jest on traktowany, odbierany, jak człowiek.

Można spotkać implementacje korzystające przy rozmowie z programów zamiany tekstu na głos. Część algorytmów syntezy mowy działa bardzo sprawnie, inne mogą raczej utrudnić konwersację. Są to elementy, które w głównej mierze mają uatrakcyjnić rozwiązanie.

Możliwości zastosowania takich programów wydają się znaczące i ciągle zyskują na popularności. Ich zastosowanie to często sposób uatrakcyjnienia strony firmy. Im bardziej wyposaża się je w dodatkowe cechy, poza zdolnością komunikacji w języku naturalnym, tym stają się bardziej przydatne. Znane są implementacje chatterbotów pełniących funkcje doradców, konsultantów, specjalistów np. duński bot Marloes udziela porad z zakresu ekonomii [25], Nicole pełni funkcje pracownika biura obsługi klienta [26], itp.

Bot nie męczy się, może pracować 24 godziny na dobę, nie pobiera pensji, nie choruje, nie bierze urlopów. Jego zastosowanie staje się opłacalne ekonomicznie.

Jednym z elementów nowej gospodarki związanej z wykorzystaniem WWW jest indywidualne podejście do klienta, tworzenie ofert dostosowanych do indywidualnych potrzeb, a nie sztywnych ofert kierowanych do większości. W spełnieniu tych celów może pomóc bot (w tym znaczeniu częściej używa się pojęcia agent). Pozwoli użytkownikowi zapoznać się z ofertą firmy, która go interesuje, doradzi, ułatwi nawigację na stronie, przyjmie zamówienie itd. Klient ma wrażenie, że bot zajmuje się tylko nim, pomimo tego, że w tym samym czasie może obsługiwać wiele innych osób. Zaimplementowana wiedza agenta na pewno jest skończona, a więc pojawią się pytania, na które nie będzie w stanie odpowiedzieć. Wówczas może przyjąć zapytanie, na które odpowiedź zostanie udzielona później, np. mailowo lub telefonicznie, albo połączyć z ludzkim przedstawicielem, jeśli jest to możliwe.

Do tej pory zaimplementowano wiele programów do prowadzenia dialogów. Pierwszy znany program (Eliza) opierał się na wyszukiwania słów kluczowych i prostych regułach przekształcania zdań użytkownika. Idea ta znalazła rozwinięcie w kolejnej generacji opartej na wzorcach ujętych w postaci reguł (Alice). W wypowiedzi rozmówcy są poszukiwane wzorce zapisane w bazie wiedzy. Po znalezieniu wzorca, program wyświetla odpowiedź towarzyszącą wzorcowi. Programy tego typu nie „rozumieją” treści, ale ich wydajność i skuteczność są w pełni wystarczające dla większości zastosowań. Kolejny etap stanowią programy dokonujące formalnej analizy zdania. Skala analizy jest różna, a najbardziej rozbudowane algorytmy dążą do utworzenia modelu semantycznego. Ten etap wynika z postępów w badaniach nad formalnymi sposobami opisu gramatyki języków naturalnych, które do tej pory jeszcze nie są pełne. Większość stosowanych opisów gramatyk, jest gramatykami bezkontekstowymi np Chomsky i skupiają się na języku angielskim.

Również dla języka polskiego w ramach prac badawczych zostały podjęte próby stworzenia opisów formalnych gramatyki. Tworząc reguły bazowano na różnym formalizmie zapisu: DCG (Definite Clause Grammar), HPSG (Head-driven Phrase Structure Grammar), FROG (Free Order Grammar). Większość z nich bazuje na regułach zapisywanych w Prologu. W większości chatterbotów reguły gramatyczne, jeśli już występują, to są dopasowywane do potrzeb konkretnej implementacji i nie pretendują do pełnego opisu gramatyki języka polskiego.

2 Projekt ALICEBOT

Twórcą Alice [10 – 16] jest dr Richard S. Wallace. Pierwsza edycja chatterbota powstała w 1995. Nazwa Alice jest tłumaczona, jako: Artificial Linguistic Internet Computer Entity. Projekt jest dostępny na zasadzie licencji (GNU General Public License), tak jak np.: Linux, lub programy open source tworzone w ramach The Free Software Foundation. Wyniki projektu mogą być używane w dowolnych zastosowania niekomercyjny i komercyjnych.

„Mózg" Alice zawiera ponad 41 tysięcy elementów nazywanych kategoriami. Każda kategoria jest kombinacją pytania i odpowiedzi, bodźca i reakcji, które są reprezentowane przez znaczniki „pattern" i „template". Oprogramowanie obsługujące AIML (standard ogólnej specyfikacji XML) przechowuje w pamięci zawartość elementów „pattern" w postaci struktury drzewa. Drzewo to jest zarządzane przez obiekt Graphmaster, który ma zaimplementowane mechanizmy składowania informacji i dopasowywania odpowiedzi.

Na popularność Alice ma wpływ również jej szybkość działania, osiąganie zakładanych celów oraz koszty wdrożenia i eksploatacji. Stosowanie standardu AIML umożliwia rozszerzanie istniejących aplikacji o umiejętności konwersacji.

Tabela. 1. Zajmowane miejsca w konkursie „The chatterbox challenge".

Rok	Miejsce
2001	2
2002	3
2003	5
2004	1
2005	3

Chatterbot Alice bierze corocznie udział w konkursach Loebnera i The Chatterbox Challenge. W konkursie Loebnera [8] program zdobywał nagrody w latach: 2000, 2001, 2004.W konkursie The Chatterbox Challenge [9] program zyskał złoty medal w 2004.

Tabela. 2. Wyróżnienia chatterbota Alice w ramach dodatkowych kategorii.

Rok	Dziedzina	Miejsce
2004	Wiedza	1
2004	Popularność	3
2005	Charakter	2

Zwycięstwa w konkursach wskazują na ciągły rozwój chatterbota, jak i na poziom tego rozwoju.

Sami twórcy jednak wskazują, że język stosowany przez sędziów przy dokonywaniu oceny programów do konwersacji odbiega od obserwowanego przy rozmowach zwyczajnych użytkowników z programem. Wskazują, że zdolności programu są znacznie większe. Pokrywają obserwowany zakres poruszanych zagadnień.

3 Dedykowany język zapisu reguł - AIML

AIML [11] jest językiem opartym na XML i podlega takim samym regułom, jak dokumenty XML. Obecnie zatwierdzona wersja standardu nosi numer 1.0.1.

Cele stawiane przed AIML:

- Łatwość nauki dla ludzi;
- Stosowanie koncepcji modelowania systemu wiedzy na regule bodziec – reakcja;
- Zgodność ze specyfikacją XML;
- Zapewnienie łatwości napisania programów do przetwarzania dokumentów AIML;
- Zrozumienie i czytelność obiektów AIML dla ludzi;
- Projekt AIML powinien być prosty i zwięzły;
- Nie może być zależny od innych języków.

4 Mechanizm przetwarzania wypowiedzi rozmówcy i udzielania odpowiedzi

4.1 Graphmaster

Wiedza" bota jest przechowywana w pamięci w postaci drzewa. Gałęzie pomiędzy węzłami reprezentują pojedyncze słowa z znaczników pattern i znaki specjalne „*" i „_". Ostatnie węzły w gałęziach drzewa noszą nazwę liści i zawierają zawartość znaczników template.

Załóżmy, że „n" jest węzłem w grafie, a „w" jest wyrazem, zaś „G(n, w)" zwraca następny węzeł „m", lub nic (null). Zbiór S_n = {w: $ m | G(n, w) = m} jest zbiorem wyrazów formatujących gałęzie od węzła „n". Jeżeli

„r” jest korzeniem (głównym, pierwszym węzłem) wówczas S_r jest kolekcją wszystkich pierwszych wyrazów z znaczników pattern.

Graphmaster przechowywuje zawartość wzorców pattern w gałęziach wzdłuż ścieżki od węzła głównego „r” do liścia zawierającego wartość znacznika template. Ścieżka ta jest uporządkowanym ciągiem wyrazów: w1, …, wk, gdzie k jest ilością wyrazów, lub tokenów ze znacznika pattern. Aby wstawić kolejną zawartość znacznika pattern do drzewa, Graphmaster najpierw weryfikuje, czy istnieje m = G(r, w1). Jeśli istnieje m, to następnie jest weryfikowane istnie kolejnego następnego węzła G(m, w2), itd., aż do k. Jeżeli dla i-tego wyrazu wi następny węzeł nie istnieje, czyli $n | G(n, wi) zwrócił null, wówczas jest tworzony nowy węzeł m = G(n, wi). Postępując według podanego algorytmu tworzymy drzewo wiedzy (bazę wiedzy – „mózg” chatterbota).

Gałęzie są sortowane według znaków z następującymi uwagami:

- gałąź „_” jest pierwsza w kolejności, jeśli występuje;
- kolejność gałęzi wynikająca z sortowania wyrazów, które ją reprezentują;
- gałąź „*”, jeśli występuje.

4.2 Normalizacja

Wypowiedź użytkownika zanim zostanie zastosowana do przygotowania odpowiedzi podlega normalizacji. Proces normalizacji możemy podzielić na kilka etapów: podstawienia, dzielenie na pojedyncze zdania, modyfikacje.

Normalizacja podstawieniowa polega na zastępowaniu ciągów znaków innymi znakami. Na tym etapie rozwija się skróty do ich pełnych form, zastępuje się znaki, które w dalszej normalizacji mogłyby być wzięte, jako znaki rozdzielające zdania, np.: kropki ze skrótów, adresów, rozszerzeń plików, poprawia najczęściej występujące literówki. W używanym programie normalizacja podstawieniowa jest realizowana na podstawie zawartości plików substitutions.xml umieszczonym standardowo w podkatalogu conf katalogu instalacyjnego programu.

W kolejnym etapie wypowiedź użytkownika jest dzielona na poszczególne zdania. Pomimo, że wypowiedzi (pytania) wielozdaniowe nie występują zbyt często, to konieczne jest uwzględnienie tego etapu. Po dokonaniu podziału, każde zdanie jest analizowane oddzielnie. Znaki, które służą, jako oznaczenia końca zdań, są umieszczone w pliku: sentencesplitters.xml w podkatalogu conf.

Ostatni etap normalizacji powoduje zamianę małych liter na duże i zamianę wszystkich znaków, które nie są zaliczane do znaków normalnych na spacje. Do znaków normalnych są zaliczane: litery, cyfry.

4.3 Ścieżka wyszukiwania

Po przeprowadzeniu procesu normalizacji, wypowiedź użytkownika staje się uporządkowaną listą wartości (słów): w1, ..., wk, gdzie: k oznacza liczbę słów. Proces wyszukiwania wzorca w regułowej bazie wiedzy przebiega rekursywnie. W pierwszej kolejności wywoływana jest funkcja Match(r, 1), gdzie: r oznacza węzeł główny (root), a 1 jest indeksem pierwszego wyrazu wypowiedzi. Funkcja ta zwraca wartość logiczną: prawda lub fałsz, w zależności od tego, czy istnieje gałąź od węzła r reprezentująca znak specjalny „_", słowo w1, lub drugi znak specjalny: „*".

Główne elementy procesu wyszukiwania polegają na:

1. Jeżeli od węzła istnieje gałąź „_", to węzeł, do którego prowadzi staje się węzłem głównym podgrafu analizowanego w dalszym kroku dla kolejnych słów.
2. Jeśli pierwszy warunek nie został spełniony, to weryfikowane jest istnienie gałęzi ze słowem odpowiadającym dopasowywanemu słowu z wypowiedzi użytkownika. Jeśli taka gałąź istnieje, to węzeł, do którego prowadzi, staje się węzłem głównym podgrafu analizowanego dla dalszych słów.
3. Jeśli warunek drugi nie został spełniony, to weryfikowane jest istnienie gałęzi „*". Jeśli istnieje, to węzeł, do którego prowadzi, staje się węzłem głównym podgrafu analizowanego do dalszych słów.

Proces wyszukiwania bierze pod uwagę nie tylko znormalizowaną wypowiedź użytkownika, ale również elementy zawarte w znacznikach that i topic. Graphmaster traktuje ich zawartość tak jak pattern. Proces wyszukiwania pracuje więc na trójce atrybutów: PATTERN, THAT, TOPIC. Przeszukiwana zawartość THAT i TOPIC wynika z wcześniejszego przeszukiwania. Jeżeli znacznik „category", który zawiera wyszukane wartości „template", nie posiada znaczników THAT i TOPIC, to w kolejnym wyszukaniu są one traktowane jako „*".

5 Tworzenie własnej bazy wiedzy

Jeśli posortujemy wyrazy według częstości ich występowania, a następnie narysujemy wykres, w którym na jednej osi umieścimy częstość występowania wyrazów, a na drugiej ilość wyrazów, wówczas otrzymamy charakterystykę tekstu. Jej kształt nosi nazwę krzywej Zipfa, lub prawa Zipfa. Jeśli na osiach zastosujemy skalę logarytmiczną wówczas otrzymamy linię prostą o nachyleniu –1.

Z prawa Zipfa wynika, że pewna liczbą wyrazów, zwrotów, zdań występuje najczęściej w procesie komunikacji. Wystarczy dla tych zwrotów, zdań opracować szablony odpowiedzi, aby pokryć większą część możliwych wypowiedzi, pytań.

Według analizy przeprowadzonej przez A.L.I.C.E. AI Foundation ok. 6000 wzorców wyszukiwania pokrywa 95% wszystkich spotykanych wypowiedzi [16].

Można stosować wiele podejść do strategii tworzenia regułowej bazy wiedzy. Zawsze jednak w pierwszej kolejności trzeba odpowiedzieć sobie na pytanie, jakich dziedzin będą dotyczyć prowadzone dialogi. Tworząc chatterbota specjalizującego się w jakiejś dziedzinie, trzeba skupić się nad zagadnieniami szczegółowymi. W dialogach ze znawcami zagadnienia będą pojawiać się pytania dotyczące konkretnych zagadnień specjalistycznych. U chatterbotów tzw. „pogaduszkowych" przygotowanie takich szczegółowych szablonów nie jest wymagane. Nie oznacza to, że przygotowanie chatterbota ogólnego przeznaczenia jest łatwiejsze. Wręcz przeciwnie, sprecyzowanie poruszanych zagadnień ułatwia tworzenie reguł. W chatterbotach dziedzinowych mamy większe możliwości skierowania tematu rozmowy na zagadnienia, w których nastąpiła specjalizacja.

Nie istnieją żadne wytyczne, jak tworzyć reguły (wzorce wyszukiwania) i szablony odpowiedzi. Ogólnie można stwierdzić, że warto uogólniać reguły tak, aby pasowały do wielu wypowiedzi oraz stosować wywołania rekurencyjne w ich analizie. W języku polskim, ze względu na złożoną fleksję oraz możliwość budowania poprawnych zdań posiadających różną strukturę, tworzenie bazy wiedzy nabiera jeszcze większego znaczenia i podnosi poziom trudności. Każdy botmaster musi wypracować sobie własny, najlepiej mu odpowiadający styl.

Tworząc reguły zespół: botmaster wraz z osobami zainteresowanymi projektem, powinien próbować wymyślić wszystkie możliwe zapytania, jakie rozmówcy mogą skierować do systemu. Najlepiej w takich sesjach stosować metodę burzy mózgów: zapisywać wszystkie pomysły, a dopiero później dokonywać ich weryfikacji. Mając taką listę dobieramy reguły i szablony wypowiedzi, tak aby wszystkie zostały uwzględnione. Po tym etapie sprawdzamy zgodność udzielanych odpowiedzi na pytania z listy z odpowiedziami, jakich sami oczekiwalibyśmy. Proces ten należy powtarzać kilkakrotnie, zwłaszcza przed uruchomieniem chatterbota, jak i regularnie podczas pracy bota.

Własną pomysłowość należy wspierać udostępnianymi w Internecie logami z zapisem dialogów prowadzonych w innych projektach chatterbotów i konkursach. Problem z bezpośrednim zastosowaniem pytań zadawanych w konkursach polega na tym, że sędziowie zadają pytania, które w normalnej pracy systemu nie zdarzają się. Np.: w konkursie Loebnera zadaniem sędziów jest rozróżnienie programu od człowieka, zadawane pytania częsta testują zdolność myślenia, posiadanie ogólnej wiedzy i tzw. „zdrowy rozsądek".

Część projektów chatterbotów jest udostępniana w Internecie z przykładowymi bazami wiedzy. Warto zapoznać się z nimi, gdyż zastoso-

wane tam sposoby rekurencyjnych wywołań wzorców, strategie i szablony odpowiedzi, mogę natchnąć nas nowymi pomysłami. Nie można ich dołączać bezkrytycznie, gdyż ich twórcy mogą prezentować zupełnie odmienne od naszych poglądy, a niespójność osobowości naszego chatterbota będzie odstręczać rozmówców.

Powyższe źródła wskazówek mają również i tę wadę, że najczęściej dotyczą języka angielskiego i często nie da się ich bezpośrednio przenieść na grunt języka polskiego.

Po uruchomieniu chatterbota skarbnicą umożliwiającą rozbudowę i modyfikację wiedzy programu, są logi przeprowadzonych dialogów. Otrzymujemy w nich wzorce zachowań użytkowników systemu, ich sposoby wyrażania się, formułowania pytań, poruszanych tematów, etc.

Tworząc szablony odpowiedzi warto nadawać im określony styl, który w dialogach będzie odbierany, jako osobowość chatterbota. Jest to rozpoznawalne, rozróżnialne dla odbiorcy i uatrakcyjnia rozmowę.

6 Zapewnienie kontekstowości

Zapewnieniu wrażenia kontekstowości sprzyja już sam mechanizm wyszukania szablonu odpowiedzi dla pytania użytkownika. Dopasowywaniu podlega bowiem trójka atrybutów: PATTERN, THAT, TOPIC (wypowiedź użytkownika, poprzednia wypowiedź chatterbota, temat).

Proces dialogu nie polega tylko na odpowiadaniu na pytania. Chatterbot sam może zadawać pytania pozornie sterując rozmową. Aby było to możliwe, musi być w stanie powiązać odpowiedź użytkownika z wcześniejszą swoją wypowiedzią i od tego uzależnić swoją następną wypowiedź. W tym celu standard AIML wprowadza znacznik <that>. Stosując znaczniki that i topic można tworzyć szablony dyskursu.

Im więcej opracowanych zostanie możliwych toków dyskusji, tym chatterbot będzie prowadził bardziej naturalną i płynną dyskusję.

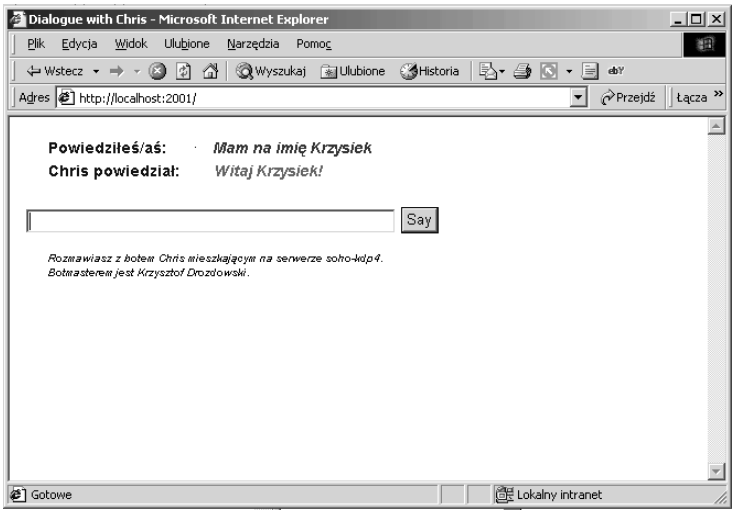
Taki mechanizm zachowania kontekstu w dominującej liczbie przypadków będzie skuteczny. Biorąc pod uwagę, że zastosowanie innego podejścia wymagałoby wprowadzenia znacznego obciążenia pamięci, co odbiłoby się na wymaganiach sprzętowych, ten sposób realizacji wydaje się być wystarczający.

7 Przykładowe dialogi

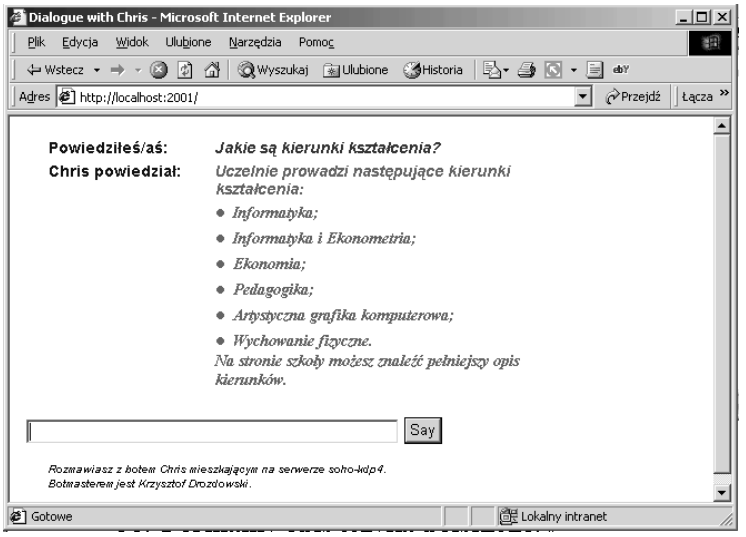
Przy powitaniu chatterbot Chris „pyta” o imię rozmówcy. Po przedstawieniu się, jego imię zostaje zapamiętane w zmiennej name, którą można wykorzystywać przy wypowiedziach programu.

W szablonach odpowiedzi można stosować znaczniki nie będące znacznikami AIML. Są one ignorowane przez interpreter AIML, ale mogą

być wykorzystywane np. przez przeglądarkę do formatowania sposobu wyświetlania.



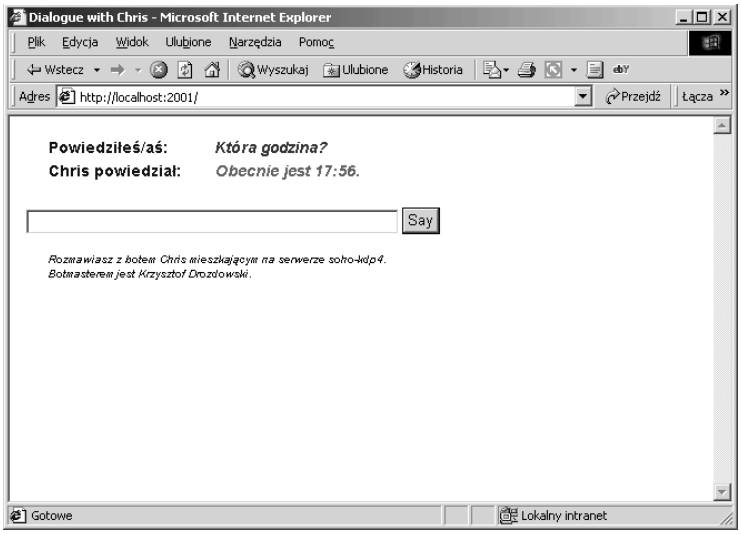
Rys. 1. Powitanie.



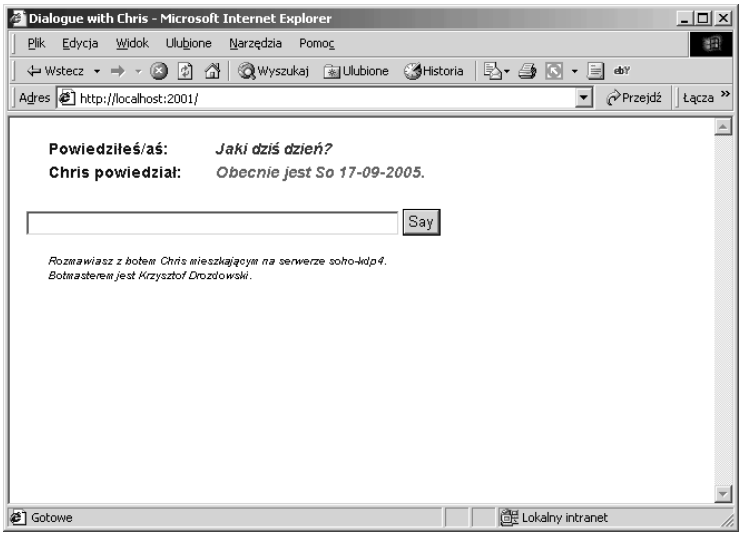
Rys. 2. Powitanie.

Stosowanie ogólnych znaków * i _ w regułach dopasowania daje w większości przypadków wrażenie odporności chatterbota na literówki ze

strony użytkowników. Warto więc tworząc reguły dopasowywania uwzględniać najczęściej popełniane przez użytkowników błędy.



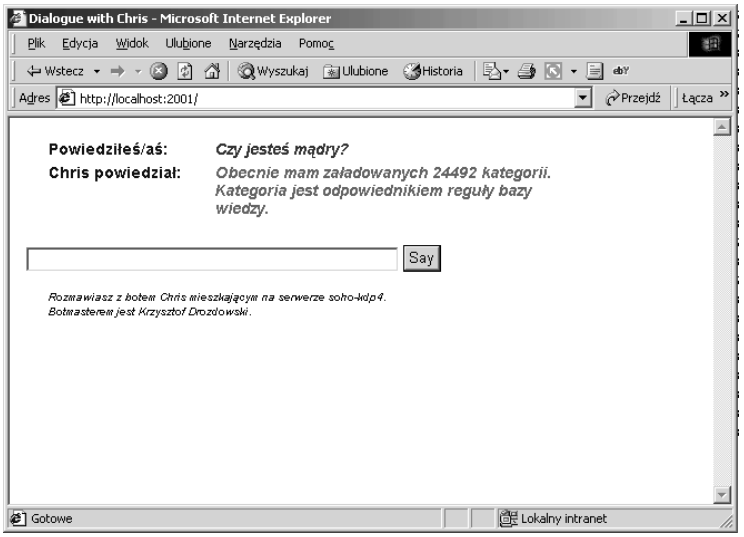
Rys. 3. Chatterbot podaje aktualną godzinę.



Rys. 4. Chatterbot podaje aktualną datę.

Chatterbot Chris został wyposażony w umiejętność podawania aktualnego czasu i daty. Osiągnięto to poprzez zastosowanie znacznika

<system>. Znacznik ten daje duże możliwości, ale również może być niebezpieczny przy niewłaściwym jego użyciu. Chatterbot jest również w stanie podać ilość załadowanych w danej chwili kategorii.



Rys. 5. Wielkość wiedzy chatterbota Chris.

We wszystkich działających projektach dotyczących chatterbotów podnoszony jest problem wulgaryzmów i specyficznego języka stosowanego przez rozmówców. Na ciągłe nawiązywanie do seksu i obelżywe uwagi należy przygotować również reguły chatterbota. Problem ten był podnoszony zarówno przez osoby biorące udział w projekcie Alice [16], jak i wypowiadał się na ten temat autor TriBota [18].

Bazując na doświadczeniach projektu Alice, dr Richard Wallace dokonał podziały rozmówców na trzy kategorie: A, B i C. Kategorię A stanowi 10 do 20 procent rozmówców, są to osoby, których języka jest wulgarny, obelżywy, wręcz pornograficzny. Kategoria B (60 do 80 procent) to rozmówcy przeciętni. Pozostały procent stanowią osoby zaliczane do kategorii C, są oto osoby określające się mianem komputerowych ekspertów, które są bardzo krytyczne w stosunku do możliwości chatterbota. Taki podział jest dodatkową informacją, z czym może spotkać się chatterbot i do czego należałoby go przygotować.

Praca botmastera to ciągłe śledzenie logów oraz praca nad poprawą i rozbudową bazy wiedzy.

8 Podsumowanie

Chatterboty oparte na regułach są w stanie realizować zadania, które są przed nimi stawiane. Wydajność i szybkość działania rekompensuje świadomość, że tak naprawdę, nie rozumieją wypowiadanych tekstów. Odpowiednio przygotowana baza wiedzy jest w stanie wytworzyć w rozmówcy wrażenie rozumienia wypowiedzi i inteligencji programu. Brak pierwszej nagrody w konkursie Loebnera świadczy jednak o tym, że do pełnej imitacji inteligentnego dialogu jest jeszcze daleko. Kiegy nie wiemy, że rozmawiamy z programem, zadajemy pytania i reagujemy inaczej niż wiedząc, że nie rozmawiamy z człowiekiem. Brak świadomości rozmowy z programem wielokrotnie już był przyczyną wielu zabawnych pomyłek.

Dalszy rozwój opisów gramatyk i dołączanie ich do regułowych chatterbotów, powinien poprawić zdolność imitowania rozumienia treści. Sądzę, że w najbliższym czasie będzie to zasadniczy kierunek rozwoju komercyjnych zastosowań systemów dialogowych. Interfejsy konwersacyjne zapewnią również nową jakość w komunikacji człowiek – maszyna. A ich zastosowanie w konkretnych przypadkach nie będzie rzadkością, a standardowym wzorcem postępowania.

Kierunki rozwoju Internetu wyznaczane przez rekomendacje World Wide Web Consortium (W3C) zmierzają do przekształcenia tej sieci w sieć semantyczną. Główny zysk będzie widoczny, gdy programy, serwisy usługowe, agenci, wirtualni asystencji, boty itd. będą w stanie się ze sobą komunikować i wzajemnie świadczyć sobie usługi. Aby zapewnić możliwość komunikacji dowolnych nieznanych sobie wcześniej programów trzeba zapewnić nie tylko określony standard komunikacji, ale także to, aby posługiwały się one tymi samymi pojęciami. Potrzebne są do tego możliwości wykorzystywania wspólnych ontologii. Umiejętność korzystania z ontologii będzie wymaganym elementem każdego chatterbota. Wizja globalnej sieci semantycznej zwiększa zapotrzebowania na rozwój metod symulacji kompetencji językowych człowieka przez systemy informatyczne.

Postępujące badania nad funkcjonowaniem ludzkiego mózgu i wzorowanie się na działaniu systemu neuronowego w zastosowaniach informatycznych wskazuje kolejny ważny kierunek rozwoju. Prace nad tymi zagadnieniami są prowadzone od wielu lat, ale uzyskane do tej pory efekty związane z próbami implementacji zdolności językowych, jak na razie nie są zbyt imponujące. Być może wkrótce nastąpi przełomowe odkrycie i naśladowanie biologicznych rozwiązań przyniesie nową jakość w dziedzinie komunikacji człowiek – maszyna.

Literatura

[1] Przepiórkowski A., Kupść A., Marciniak M., Mykowiecka A.: Formalny opis języka polskiego. Teoria i implementacja. Exit 2002

[2] Vetulani Z.: Komunikacja człowieka z maszyną. Komputerowe modelowanie kompetencji językowych. Exit 2004

[3] Meller A.: Dialog z maszyna. Software 2.0: narzędzia, programy, sieci; Luty 2002

[4] Kandefer T., Meller A.: Czeka nas era chatterbotów. Software 2.0: narzędzia, programy, sieci; Luty 2004

[5] Jessa S.: Czy chatterboty nas rozumieją? Software 2.0 Extra!; Nr 10 2004

[6] Meller A.: Wystaw bota na zawody. Software 2.0 Extra!; Nr 10 2004;

[7] Karolewska W.: Bot - wirtualny rozmówca i przyjaciel. - Młody Technik; Październik 2004

[8] http://www.loebner.net/Prizet/loebner-prize.html - Strona główna konkursu Loebnera

[9] http://www.chatterboxchallenge.com - Strona konkursy „The Chatterbox Challenge", podczas którego są wyłaniane najlepsze programy do konwersacji

[10] http://www.alicebot.org - Projekt Alice

[11] http://alicebot.org/TR/2005/WD-aiml - Opis AIML

[12] http://www.alicebot.org/articles - Artykuły dotyczące projektu Alice

[13] http://www.alicebot.org/articles/matching.html - Opis sposobu udzielania odpowiedzi przez Alice

[14] http://www.alicebot.org/articles/wallace/zipflaw.html - Artykuł: "Zipf's Law"

[15] http://www.alicebot.org/documentation/ - Dokumentacja systemu Alice

[16] http://www.alicebot.org/documents/Anatomy.doc - Opis filozofii, założeń, głównych elementów projektu, fundacji

[17] http://www.chatbot.pl - Chatterbot TriBot

[18] http://www.cybercore.republika.pl - Cybercore nr 7 str. 9 – 10

[19] http://www.lingubot.pl/cgi-bin/fido/Fido.cgi; http://www.fido.pl – Lingubot Fido

[20] http://www.sterprojekt.com.pl - Lingubot Adam

[21] http://www.hestia.pl - Lingubot Hubert

[22] http://www.denise.gnu.pl; http://www.denise.union.pl - Projekt Denise

[23] http://www.ipipan.waw.pl/mmgroup/HPSG/-"The Linguistic Engineering Group: Polish in HPSG"

[24] http://main.amu.edu.pl/~vetulani; http://www.amu.edu.pl/~vetulani - Strona prof. Zygmunta Vetulani.

[25] http://www.marloes.nl – bot Marloes

[26] http://www.nativemainds.com – bot Nicole

CONSTRUCTION OF CHATTERBOT CHRIS

Summary – Development of artificial intelligence methods and looking for Improvement of existing contacts channels with clients are main reasons to increase of interests of programs for simulation human communication competences including spelling. Now this kind of programs are developed so far that have commerce application also in Poland (Adam, Hubert etc). In this paper chatterbot Chris based on looking for the pattern in user statement is presented. Presented solution is based on technologies created at the Alice project (A.L.I.C.E. foundation). The creation of the adequate knowledge base is the main part of programmers work. Some conclusions and advices which can be helpful to crate knowledge base adequate to programmers needs are presented. Presented bot is based on Java servlets, so its efficiency is satisfactory.