

Пловдивски университет „Паисий Хилендарски“



Факултет по математика и информатика

ДИПЛОМНА РАБОТА

на тема

Автоматизирано създаване на семантични карти

Дипломант:

Траян Момков
ф.н. 0901407002

специалност: Софтуерни технологии

Ръководител:

проф. д.м.н. Георги Тотков

Пловдив
2010

Благодаря на свободната енциклопедия Wikipedia
и на движението за свободен софтуер,
без които тази дипломна работа
не би могла да се осъществи

Специални благодарности и към
Катерина Ташева

Съдържание

Списък на съкращенията.....	3
Резюме.....	4
1. Увод.....	5
1.1. Идея за проекта.....	5
1.2. Цел на дипломната работа.....	5
1.3. Задачи за постигане на целите.....	5
2. Проучване.....	6
2.1. Семантични мрежи.....	6
2.1.1. Примери за семантични мрежи.....	7
2.1.2. Софтуерни системи за семантични мрежи.....	8
2.1.3. Тематични карти.....	9
2.1.4. Понятийни карти.....	9
2.1.5. Мисловни карти.....	10
2.1.6. Семантични карти.....	11
2.2. Класификация на Блум.....	11
2.3. Бъбромати.....	16
3. Описание на програмата.....	17
3.1. Клиентска част.....	17
3.1.1. Описание на използваните технологии.....	17
3.1.2. Описание на клиентската част.....	20
3.2. Сървърна част.....	21
3.2.1. Perl – общ преглед.....	23
3.2.2. Структури от данни в Perl.....	23
3.2.3. Структури от данни използвани в бота.....	24
3.2.4. Регулярни изрази в Perl.....	27
3.2.5. Регулярни изрази използвани в бота.....	28
3.2.6. ImageMagick – общ преглед.....	30
3.2.7. Генериране на семантична карта.....	31
4. Резултати.....	33
5. Заключение.....	34
5.1. Какво е постигнато.....	34
5.2. Проблеми в процеса на работа.....	34
5.3. Области от информатиката, които са засегнати.....	35
5.4. Бъдещо развитие.....	35
Литература.....	36
Списък на изображенията и таблиците.....	37
Приложение 1 – пример.....	38
Разговор.....	38
Извлечена от разговора информация.....	43
Автоматично създадената семантична карта.....	47
Приложение 2 – резултати от експерименти.....	48

Списък на съкращенията

AJAX	Asynchronous JavaScript and XML (асинхронен JavaScript и XML)
CSS	Cascading Style Sheets (език за описание на стилове)
DHTML	Dynamic HTML (динамичен HTML)
DOM	Document Object Model (обектен модел на документа)
GUI	Graphical User Interface (графичен потребителски интерфейс)
HTML	HyperText Markup Language (език за маркиране на хипертекст)
HTTP	Hypertext Transfer Protocol (протокол за трансфер на хипертекст)
LAMP	Linux, Apache, MySQL и Perl/PHP/Python
MIT	Massachusetts Institute of Technology (Масачузетският технологичен институт)
MSL	Magick Scripting Language (скриптов език Magick)
MultiNet	Multilayered extended semantic networks (многослойна разширена семантична мрежа)
NASM	Netwide Assembler
OWL	Web Ontology Language (език за уеб онтологии)
RDF	Resource Description Framework (рамка за описание на ресурси)
SnePS	The Semantic Network Processing System (система за обработка на семантични мрежи)
SNOMED CT	Systematized Nomenclature of Medicine – Clinical Terms (систематизирана номенклатура на медицината – клинични термини)
UML	Unified Modeling Language (единен език за моделиране)
WWW	World Wide Web (световната мрежа)
XHR	XMLHttpRequest
XML	Extensible Markup Language (разширен език за маркиране)
Бисквитки	Cookies
Бот	Робот за разговори – бърбот (chatbot)
Браузър	Browser
ОО	Обектно-ориентиран
ПО	Предметна област
Сайт	Website
Уеб	Web

Резюме

Настоящата дипломна работа се състои от увод, изложение и заключение, разработени в общо 52 страници. В началото е поместен **списък на съкращенията**, а в края - списък на изображенията и таблиците. Препратки в текста сочат към списък с използваната **литература**. Налични са и **2 приложения**.

В увода са посочени идеята (обосновката), **целта на дипломната работа**, и задачите, които трябва да се изпълнят за постигането на целта.

Изложението е разделено на 3 части.

В първата от тях е направено **проучване**, описана е същността на семантичните мрежи, класификацията на Блум и роботите за разговори. Представени са примери за конкретни семантични мрежи и системи за работа с тях, както и един от първите ботове.

Втората част описва проектирането и създаването на системата. Представени са **използваните технологии**. Описана е клиентската част и потребителският интерфейс. Чрез **диаграма** нагледно е показан начинът на работа на сървърната част. В две таблици са обяснени **използваните структури от данни** и регулярни изрази. Направен е преглед на графична библиотека, чрез която се изобразяват картите, и евристиките, които се използват за това. Форматът, в който се запазва семантичната карта е представен в **приложение 1. Инструкции за използване** на системата или по-точно за провеждане на разговор се дават преди самото му започване в страницата на бота: <http://trekto.info/semantic-bot>

В третата част от изложението е описан **проведеният експеримент** със студенти от Факултета по математика и информатика към Пловдивския университет „Паисий Хилендарски“. Резултати от него са поместени в **приложение 2**.

В заключението е посочено **какво е постигнато**, какви **проблеми са възникнали** в процеса на работа, както и кои **области на информатиката** са засегнати. Представени са възможни направления за **бъдещо развитие**.

Приносът на автора се изразява най-вече в написването на въпросите, с помощта на които да се събира информация за дадена предметна област от потребителите, както и в създаването на системата. Новото в системата е, че тя създава семантична карта чрез обикновен разговор с потребителя.

Дипломната работа е достъпна онлайн на адрес <http://trekto.info/semantic-maps>

1. Увод

1.1. Идея за проекта

Компютърното моделиране на информацията и автоматизирането на информационните дейности предполагат въвеждането, изучаването и използването на различни модели на информацията за обектите и за дейностите, в които те участват. Информационните обекти и информационните процеси са абстрактни модели на информацията и информационните дейности, наречени абстрактни (концептуални) информационни модели. В компютърната информатика абстрактните информационни модели се проектират и създават под формата на алгоритми и структури от данни. [\[Тотков\]](#)

Идеята е може ли създаването на такива модели за различни предметни области (ПО) да бъде автоматизирано?

1.2. Цел на дипломната работа

Автоматизирано създаване на описание (модел) на предметна област.

1.3. Задачи за постигане на целите

Нужно е да се създаде система, която чрез диалог с човек да създава семантична карта по предварително зададена тема. Приема се, че човекът, с който се води диалогът не е запознат с понятието семантична карта и не знае как се прави такава. Създадената карта трябва да се съхранява в удобен за по-нататъшна работа вид.

2. Проучване

2.1. Семантични мрежи

Семантичната мрежа е графична система за означаване (notation) представяща знания чрез свързани помежду си върхове и дъги. Компютърна реализация на семантична мрежа най-напред е била разработена за **изкуствения интелект** и **машинния превод**, но след това дълго време се използва и във **философията, психологията и лингвистиката**.

Общото между всички семантични мрежи е декларативното графично представяне, което може да се ползва, както да **представя знания**, така и да подпомага системите за **автоматично аргументиране** (reasoning).

Шестте най-срещани типа семантични мрежи са следните:

1. Дефиниционните мрежи подчертават подтиповите връзки от вида „е“ (is-a) между тип и дефинирани негови подтипове. Такива мрежи се наричат още обобщаващи йерархии и поддържат правилото за наследяване на свойства на типа от всички негови подтипове. Доколкото определенията са верни по подразбиране, информацията в тези мрежи често се приема, че е непременно вярна.

2. Декларативните мрежи (assertional networks) са предназначени да представят твърдения. За разлика от дефиниционните мрежи, информацията в декларативните мрежи се приема за условно вярна, освен ако изрично не е отбелязана с модален оператор. Някои декларативни мрежи се приемат, като модели на концептуалните структури, върху които е изградена семантиката на естествените езици.

3. Импликационните мрежи използват импликацията, като основна връзка между отделните върхове. Могат да се използват, за да представят убеждения (beliefs), причинно-следствени връзки или изводи.

4. Изпълнимите мрежи включват механизми, като предаване на маркер или прикачени (attached) процедури, които могат да правят логически изводи чрез дедукция, да предават съобщения или да търсят шаблони и асоциации.

5. Изучаващите мрежи (learning networks) се построяват или разширяват чрез придобиване на знания от примери. Новите знания могат да променят старата мрежа чрез добавяне или изтриване на върхове и дъги,

или чрез промяна на числени стойности наречени тегла, свързани с върховете и връзките.

6. Хибридните мрежи са комбинация на две или повече от горните, както в една единствена мрежа така и в няколко отделни, но взаимодействащи помежду си мрежи. [\[Sowa\]](#)

Семантичните мрежи са разработени за първи път за компютри от Richard H. Richens през 1956 година, в отдела за изследване на езици към университета в Кеймбридж, като междинен език използван за машинен превод на естествени езици. [\[Wikipedia 4\]](#)

2.1.1. Примери за семантични мрежи

Пример за семантична мрежа е **WordNet**. Тя е лексикална база данни на английския език. Групира английските думи в множества от синоними наречени *synsets*, предоставя кратка, обща дефиниция на думите и различни семантични връзки между тези семантични множества. Някои от най-общите семантични връзки са дефинирани като:

- мероними (*meronymy*) – А е част от Б, Б притежава А, като част от себе си;
- холоними (*holonymy*) – Б е част от а, А притежава Б, като част от себе си;
- хипоними (*hyponymy*) или тропоними (*troponymy*) – А е подчинен на Б, А е вид Б;
- хиперними (*hypernymy*) – на А е подчинен Б;
- синоними – А означава същото, като Б;
- антоними – А означава обратното на Б.

Също така е възможно да се представят логически описания използвайки семантични мрежи, като екзистенциални графи на Пиърс (Charles Sanders Peirce) или свързани концептуални графи на Сова (John F. Sowa). Това е толкова полезно и дори повече отколкото стандартната предикатна логика от първи ред. За разлика от WordNet, семантична мрежа използваща такова представяне може да бъде използвана за правене на надеждна и автоматична, логическа дедукция. Някои системите за автоматично аргументиране използват тези предимства на семантичните мрежи.

Други примери за семантични мрежи са **моделите на Гелиш (Gellish)**. Гелиш-английският заедно с речника за Гелиш-английски е формален език, който е дефиниран, като мрежа от връзки между понятия и имена на понятия. Той е формално подмножество на английския език, както Гелиш-немският е формално подмножество на немския език, като идеята е, че много езици имат общи концепции. Други познавателни и информационни модели на Гелиш са

обяснени с езика на Гелиш. Мрежа на Гелиш е мрежа от двойки връзки между неща. Всяка връзка в мрежата представя факт, класифициран от тип на връзката. Всеки тип на връзката от друга страна е понятие, което е дефинирано в речника на Гелиш езика. Всяко свързано нещо е или понятие или индивидуално нещо, класифицирано от понятие. Дефинициите на понятията са създадени във форма на дефиниционни модели, които заедно формират речника на Гелиш. Мрежа на Гелиш може да бъде документирана в база данни на Гелиш и може да се интерпретира от компютър. [\[Wikipedia 4\]](#)

Още примери за семантични мрежи:

- **Lexipedia** – онлайн, визуална, семантична мрежа с речник и енциклопедия. Представя думите заедно с техните семантични връзки изобразени като анимирана паяжина. [\[Wikipedia 7\]](#)
- **SNOMED CT** (Systematized Nomenclature of Medicine – Clinical Terms – систематизирана номенклатура на медицината – клинични термини) – систематично организиран, компютърно обработван архив от медицински термини обхващащ повечето области от клиничната информация, като болести, процедури, микроорганизми фармацевтични продукти и други. Предоставя начин за организиране, съхраняване, извличане и обобщаване на клинични данни. Също така помага за организиране на съдържанието на медицински текстове, намалявайки неяснотите при събирането, кодирането и използването на данни при клинично лечение на пациенти и проучвания. [\[Wikipedia 8\]](#)
- **Cognition** е семантична мрежа на английския език, за която авторите твърдят, че е най-голямата комерсиална система за обработка на естествен език. [\[Cognition\]](#)

2.1.2. Софтуерни системи за семантични мрежи

Като мрежово базирана система **SnePS** (The Semantic Network Processing System – Система за обработка на семантични мрежи) е декларативна, семантична мрежа. [\[Wikipedia 9\]](#)

MultiNet (Multilayered extended semantic networks – многослойна разширена семантична мрежа) е едновременно парадигма за представяне на знания и език за представяне на изрази от естествения език. Твърди се, че тя е една от най-обширните и подробно описани системи за представяне на знания. Притежава понятийна структура от значения на около 140 предефинирани връзки и функции, които са систематично охарактеризирани и подкрепени от формален аксиоматичен апарат. Отделно от техните връзки, понятията са вградени в многоразмерно пространство от свойства и техните стойности.

MultiNet е използван в практически системи за обработка на естествен език, като интерфейс на естествен език към Интернет и системи отговарящи на въпроси над обширни, коментирани, семантични корпуси съставени от милиони изречения.

Някои инструменти на MultiNet са семантичният интерпретатор **WOCADI**, който превежда изрази от естествен език (фрази, изречения, текст) към формални изрази за MultiNet, **MWR+** за знанияни инженери (knowledge engineer) (съдържащ модули за автоматично придобиване на знания и аргументиране) и **LIA+** за компютърни лексикографи, подпомагащ изграждането на големи семантично базирани речници.

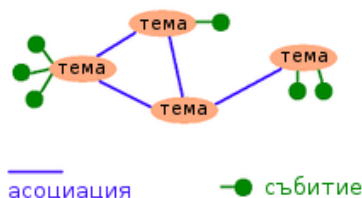
2.1.3. Тематични карти

Тематичните карти (topic maps) са **стандарт** за представяне и обмен на знания, който набляга на леснотата при откриване на информация (findability). ISO стандартът е **ISO/IEC 13250:2003**.

Тематичните карти представят информация използвайки:

- теми (topics), представящи понятия, като хора, държави, организации, програмни модули, събития и т.н.;
- връзки (associations) между темите;
- събития (occurrences), представящи информационни източници свързани с конкретна тема.

Темите, връзките и събитията могат да бъдат типизирани, като всеки тип трябва да бъде дефиниран от създателя на тематичната карта. Дефиницията на позволените типове е известна, като онтология на картата. [\[Wikipedia 5\]](#)



Илюстрация 1:
Тематична карта

2.1.4. Понятийни карти

Понятийните карти (concept map) представляват диаграми показващи връзките между понятия. Те са графични инструменти за **организиране** и **представяне на знания**.

Понятията обикновено са представени като правоъгълници или елипси и са свързани със стрелки с надписи в йерархична структура от тип отгоре-надолу. Някои възможни връзки между понятията са „необходимо за“, „допринася за“, „представлява“, „е“ и други.

2.1.6. Семантични карти

Семантичните карти са средство за представяне на знания. Те описват различни понятия и връзките между тях. Тематичните, понятийните и мисловните карти се припокриват до известна степен и представляват **частни случаи** на семантични карти. От друга страна семантичната карта е много малка семантична мрежа или по-скоро семантичната мрежа може да бъде съставена от много свързани помежду си карти.

Понятието семантична карта има още едно значение: географска карта, върху която е разположена и друга полезна информация свързана с географското място. Повече информация за това може да се намери на адрес: http://www.mediawiki.org/wiki/Extension:Semantic_Maps

2.2. Класификация на Блум

Чрез класификацията на Блум се определя какво очакваме или възнамеряваме учащите да научат, като резултат от обучението им. Бенжамин С. Блум, работещ в Чикагския университет, подема идеята за **класифициране на знанието** с надеждата, че това ще улесни тежката задача по съставяне на годишните общообразователните изпити в САЩ. За да успее в своето начинание той съставя група от специалисти по изпитване, много от които редовно са принудени да се справят с подобни проблеми. Тази група се събира два пъти годишно от 1949 до 1956 година, за да обсъжда напредъка, прави оценка на свършената работа и да планира следващите задачи. Техният краен проект е публикуван през 1956 под заглавието Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain - **Класификация на образователните цели**. Ръководство I: Познавателна област. (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956). Това е оригиналната класификация. Нейн преработен вариант е публикуван 45 години по-късно: Anderson, L.W. (Ed.), Krathwohl, D.R. (Ed.), Airasian, P.W., Cruikshank, K.A., Mayer, R.E., Pintrich, P.R., Raths, J., & Wittrock, M.C. (2001) – A taxonomy for learning, teaching, and assessing: A revision of Bloom's Taxonomy of Educational Objectives (Класификация за учене, обучение и оценяване: **Преработена класификация на Блум**).

За Блум оригиналната класификация, е нещо повече от средство за оценяване. Тя би могла да служи като:

- Основа за създаване на индивидуален курс или учебен план, които да покриват по-широк кръг от образователни цели;

- Начин за съгласуване на образователните цели, оценяването и самоинициативността. Обединяването им в едно цяло – един курс или учебна програма;
- Обща картина на образователните възможности, която е в контраст с ограничения обхват и слабата задълбоченост на всеки отделен образователен курс или програма.

Оригиналната класификация представя йерархична подредба на шестте главни категории в познавателната област. Те са подредени от по-прости към по-сложни, от конкретни към абстрактни. Категориите са **знания, разбиране, приложение, анализ, синтез, оценка**. Като изключим приложението, всяка една от тези категории е разделена на подкатегории. Освен това е прието оригиналната класификация да бъде представена във възходяща йерархия. Усвояването на всяка по-проста категория е необходимо условие, за да се усвои следващата – по-сложна. По времето, когато е въведен, терминът класификация е непознат за образователната област. Потенциалните ползватели не знаят какво означава той, поради което първоначално на оригиналната класификация не се отделя достатъчно внимание. Но след като читателите разбират за възможностите ѝ, нейната структура става широко известна и похвалите не закъсняват. В последствие тя е преведена на 22 езика.

Най-често оригиналната класификация се ползва, за да се категоризират образователните цели и средствата за проверка на знанията. Те от своя страна показват постигнатия напредък и развитие. Почти винаги направените анализи, показват съсредоточаване върху целите, изискващи само разпознаване или възпроизвеждане на информация. Те попадат в категорията знания. Обикновено обаче най-важните цели, които си поставя образованието, са свързани с разбирането и използването на знанията. Те могат да се класифицират в категориите разбиране, приложение, анализ и синтез. По тази причина такива анализи спомагат за периодичното обновяване на учебната програма и начините за изпитване. Така се поставят все по-високи цели, които попадат в по-сложните категории. [\[Krathwohl\]](#)

Оригиналната класификация на Блум, преведена на български език може да се види в следващата таблица.

Когнитивна област	Формулировка на общите цели	Глаголи за формулиране на конкретни цели
Знания Паметта за научен вече материал. Отнася се от факти до теории. Основното е способността да се възпроизведе необходимата в момента информация.	Познава конкретните факти. Знае общата терминология, основните концепции, функциите на...	описва, изброява, възпроизвежда, формулира, посочва, назовава, избира, подчертава
Разбиране Способността да се схване смисълът на изучаваната материя. Проявява се трансформиране на знанията чрез обяснение, обобщение, оценяване на бъдещи тенденции. Използва се информацията от вече изучавани области.	Разбира факти, принципи, обяснява методи и процедури, оценява последствия.	преобразува, защитава, различава, оценява, обяснява, разширява, обобщава, дава примери, перифразира, прогнозира, преразказва, дефинира, обсъжда, сравнява
Приложение Способност да се използва наученото в нови и конкретни ситуации. Свързва се с прилагането на принципи, правила, концепции, методи, теории. Знанията и разбиранията се прилагат в съвсем нови, неизучавани области и ситуации.	Прилага концепции в нови ситуации. Демонстрира правила, употреба на нов метод или процедура.	използва, доказва, открива, решава, променя, разработва
Анализ Способността за разделянето на дадена материя на съставните ѝ части, за да се разбере и изследва структурата ѝ. Свързан е с определянето на компонентите, анализ на връзките им, изследване на принципите за организиране на съответната материя. По-високото ниво за схващане на смисъла се свързва освен със съдържанието и със структурата на материала.	Разпознава неявно формулирани идеи и предположения. Разпознава логически парадокси и софизми в разсъжденията. Различава факти от изводи. Оценява приложимостта на данните. Анализира структурата на план, проект, експеримент. Разбива материала на съставните му части.	разделя на компоненти, представя графично, различава, диференцира, определя, илюстрира, прави заключения и изводи, обобщава, посочва, свързва, избира, разделя, подразделя
Синтез Обединяване на новите части, за да се получи ново цяло. Свързва се с разработката на нов метод за комуникация, план за действие при изследователска работа, проект, система от абстрактни отношения, например класифициране на информация. Изисква се творческо поведение с акцент върху разработката на нови модели и структури.	Развива нови идеи. Организира добре изложението в дадена писмена тема. Има добре развита и организирана реч. Пише разкази. Разработва план за проект. Интегрира наученото в други области в план за решаване на даден проблем.	категоризира, комбинира, събира, съставя, създава, изобретява, проектира, обяснява, генерира, модифицира, преподава, реконструира, разпознава, ревизира
Оценка Способността да се окачествява стойността / ценността на дадена материя / стих, доклад, план и др./ с дадено предназначение. Оценките се базират на точно определени критерии. Резултатите от учебния процес изискват най-високо ниво в когнитивната област, защото съдържат елементи от всички други категории плюс съзнателно оценяване на стойности.	Оценява логическата съгласуваност на даден писмен материал, адекватност на данните, подкрепящи заключенията. Оценява стойността на работата, като използва външни / независими/ стандарти за качество. Оценява стойността на работата като използва вътрешни критерии.	оценява, категоризира, сравнява, прави заключения и изводи, критикува, описва, обяснява, диференцира, доказва, интерпретира, свързва, обобщава, посочва, оспорва, защитава.

Таблица 1: Оригинална класификация на Блум. [\[Анонимен\]](#)

Преработената класификация на Блум представлява двуизмерна рамка представяща познанието и мисловните процеси. В едното измерение са категориите от оригиналната класификация, а в другото следните четири: факти, понятия, процедури и метазнание. Шестте категории от оригиналната версия са променени както следва:

- Знание – Помнене;
- Разбиране – Разбиране;
- Приложение – Прилагане;
- Анализ – Анализирание;
- Синтез – Създаване;
- Оценка – Оценяване.

Също са подредени йерархично, но не толкова строго и последните две категории са разменени: оценяването е преди създаването.

За целите на дипломната работа бяха подбрани части от преработената класификацията на Блум и допълнени с нови идеи, което доведе до създаването на таблицата по-долу. Чрез нея бяха уточнени въпросите, които да задава ботът.

	Факти	Понятия	Процедури	Метазнание
Поменне	<p>Избройте фактите/новите думи, които сте запомнили.</p> <p>Какво е ...? за обект</p> <p>Как става ...? за действие</p> <p>Къде (в пространството) ...?</p> <p>Какви примери са дадени.</p> <p>Опишете ги.</p> <p>Исторически факти</p> <p>Избройте личностите, които участват.</p> <p>Опишете ги.</p> <p>С какво друго се занимават тези личности?</p> <p>Избройте местата, на които се случват действията.</p> <p>Кога се случват действията, в продължение на какъв период?</p>	<p>Кои са главните идеи, понятия?</p> <p>Дайте значението им.</p> <p>Какви примери са дадени.</p> <p>Опишете ги.</p>	<p>Избройте процедурите, които сте запомнили.</p> <p>Опишете процедурите стъпка по стъпка.</p>	
Разбиране	<p>Дайте синоними/обяснете с други думи фактите.</p> <p>Подредете фактите в групи/категории.</p> <p>Какви са причинно-следствените връзки между фактите?</p> <p>Какво доказва всеки един от фактите?</p>	<p>Каква е областта?</p> <p>Обяснете с други думи понятията.</p> <p>Подредете понятията в групи/категории.</p> <p>За какви други понятия свързани с изброените се досещате?</p> <p>Дайте конкретни примери за всяко понятие, като посочите връзката между понятието и примера.</p>	<p>Избройте проблемите, които са решени и чрез кои процедури се решават.</p> <p>Подредете процедурите в групи/категории.</p> <p>Какви са условията необходими за прилагането на дадена процедура?</p> <p>Какви са последиствията от прилагането на дадена процедура.</p>	<p>Каква е целта?</p>
Прилагане	<p>В какви ситуации, къде могат да се приложат фактите? Дайте примери.</p> <p>Използвайте фактите, за да докажете дадена теза.</p> <p>Как се отразява това на всекидневния живот?</p>	<p>Как може ... да бъде използвано за ...?</p> <p>Как се отразява това на всекидневния живот?</p>	<p>По какъв начин наученото може да се приложи на практика?</p> <p>Демонстрирайте, дайте пример за прилагане на процедурите.</p> <p>Използвайте процедура, за да докажете дадена теза.</p> <p>Как би се отразило това на всекидневния живот?</p>	<p>Как се отразява това на всекидневния живот?</p>
Анализиране		<p>Какви са йерархичните зависимости между категориите понятия?</p> <p>Каква е връзката между всяко понятие и неговата категория?</p> <p>Какви свойства (качество, характеристики) притежава понятието?</p> <p>Какви са връзките между понятието и своята категория?</p> <p>Какви други връзки има между понятията?</p> <p>Ако е възможно заменете връзките между понятията с други.</p> <p>Какви са разликите между ... и ...?</p> <p>По какво си приличат ... и ...?</p>	<p>Какви са времевите връзки между процедурите? Подредете ги по ред на изпълнение.</p> <p>Какви други връзки има между процедурите? Сравнете отделните процедури помежду си.</p>	<p>Сравнете идеите и понятията от областта с такива от други области.</p> <p>Какви прилики и разлики откривате?</p> <p>Може ли да се счита, че идеите и понятията са изоморфни (сходни) на такива от друга област?</p> <p>Опитайте се да си представите какви последиствия би имало ако хората от друга епоха са притежавали тези знания.</p> <p>Опитайте се да проектирате развитието на нещата в бъдеще.</p> <p>Сравнете получените знания с тези, които сте получили последния път (когато сте получавали знания).</p>
Оценяване	<p>Кои са най-важните факти? Подредете ги по значимост.</p> <p>Има ли грешни твърдения?</p>	<p>Подредете по значимост понятията.</p> <p>Ако трябва да оставите само едно понятие, една идея и да изхвърлите останалите, кое ще бъде то?</p> <p>Знанията в правилната област ли са? Не трябвали да бъдат в друга област?</p>	<p>Оценете процедурите от различни гледни точки, спрямо различни крайни цели.</p> <p>Кой от изброените начини за справяне с проблемите бихте приложили вие и защо?</p> <p>Има ли грешки в процедурите и какви са те? Подредете по значимост процедурите.</p>	<p>Какви са ползите от наученото/от тези знания? За кого ще бъдат полезни тези знания?</p> <p>Има ли смислови/логически грешки в предоставените знания?</p>
Създаване	<p>Какви факти (за какво) липсват или са малко?</p> <p>Какви факти бихте добавили вие?</p> <p>Как ще проверите фактите?</p> <p>Къде ще потърсите още факти?</p>	<p>Кои понятия е нужно да се доизяснят и развият?</p> <p>Как бихте го направили вие?</p> <p>На какви въпроси още трябва да се отговорят/да бъдат обсъдени?</p> <p>Как бихте обяснили дадено понятие на друг човек или дете?</p>	<p>Кои процедури имат нужда от промяна?</p> <p>Как бихте ги променили вие?</p> <p>Предложете нови процедури</p> <p>Има ли други възможни начини за решаване на даден проблем?</p> <p>Как ще проверите дали процедурите доведат до желания резултат?</p> <p>Къде ще потърсите нови процедури?</p> <p>Как бихте обяснили дадена процедура на друг човек или дете?</p>	<p>Преразкажете наученото от гледна точка на някоя друга област, като наблегнете на важните за областта неща.</p> <p>Комбинирайте придобитите познания с такива от други области и създайте нови приложения или знания.</p> <p>Предскажете какво би станало ако тези знания се приложат на практика.</p> <p>Какво още трябва да се изясни/какви теми да бъдат засегнати?</p>

Таблица 2: Преведена и допълнена, преобразована класификация на Блум

2.3. Бъбромати

Бъброматът представлява компютърна програма, проектирана да имитира интелигентен разговор с един или няколко човека, посредством гласови или най-често текстови съобщения. По принцип целта на тази имитация е да се заблудят потребителите, че съобщенията от програмата са написани от човек (тестът на Тюринг). Подобни програми понякога са наричани **изкуствени единици за разговори**, роботи (**ботове**) за разговор или **разговорни кутии**. В последно време бъбромати започват да се използват за практически цели, като онлайн помощници, персонализирани услуги или събиране на информация, в които случаи програмата работи, като разговорен агент. Това, което различава бъбромата от по-сложните системи за обработка на естествени езици е простотата на използваните алгоритми. Въпреки че много бъбромати изглежда сякаш тълкуват човешките съобщения интелигентно, те просто претърсват списък с ключови думи и изпращат отговор с най-близката до входното съобщение или генерират отговор по разпознат шаблон от база данни с шаблони.

Терминът „ChatterBot“ е въведен от Michael Mauldin (създател на първия вербален бот Джулия) през 1994 година. [\[Wikipedia 10\]](#)

Българският превод на този термин – „Бъбромат“ е въведен от Траян Момков през 2008 година, за назоваване на неговия, подобен на ELIZA бъбромат, който се намира на адрес <http://trekto.info/bot>

ELIZA е един от първите бъбромати. Една от най-известните ѝ роли е тази на психотерапевт. Без да използва почти никаква информация за мислите и емоциите на човека, ELIZA постига изумяващи резултати. Тя е написана в MIT (Massachusetts Institute of Technology – Масачузетският технологичен институт) от Joseph Weizenbaum между 1964 и 1966 година.

Когато пациентът не предоставя почти никаква информация, ELIZA отговаря възможно най-общо. Например на съобщението „Боли ме главата“ се отговаря с „Защо казваш, че те боли главата?“. Отговор на съобщението „Моята майка ме мрази“ може да бъде „Кой друг от семейството ти те мрази?“. ELIZA е реализирана чрез разпознаване на шаблони, и все пак е взета на сериозно от няколко нейни събеседника, въпреки че Weizenbaum им е обяснил как работи. [\[Wikipedia 11\]](#)

3. Описание на програмата

За постигане на целите е създадена система работеща в уеб среда. Следва нейното описание.

Системата представлява робот за разговори (**бот**), който разпитва човека за лекцията или урока, на който е присъствал или прочел и се опитва да извлече систематизирани знания от разговора. Знанията се запазват в удобна форма (представена в [приложение 1](#)), която представлява семантична карта, след което се визуализира. **Въпросите**, които задава ботът са **динамични** - променят се до известна степен спрямо предишните отговори на потребителя. Така например ако потребителят е изброил две процедури ботът ще зададе няколко въпроса за тях, докато ако не са изброени никакви процедури въобще няма да се пита за тях. При въпроса за връзките между понятията, примерни такива се прочитат от файл и се предлагат на потребителя. Той може да използва някои от тях или да въведе собствена връзка, която след това се добавя към файла и се предлага като възможна на следващия потребител. Добра представа за целия процес на разпитване може да се придобие ако се разгледа примерният разговор между човек и бота, който се намира в [приложение 1](#).

Програмата се състои от клиентска и сървърна част.

3.1. Клиентска част

3.1.1. Описание на използваните технологии

HTML, съкращение от HyperText Markup Language (език за маркиране на хипертекст), е най-разпространеният и използван език за описание на уеб страници. Той позволява да се създават структурирани документи изградени от заглавия, параграфи, списъци, връзки, цитати и други. Също така е възможно да се вграждат изображения и обекти, като форми за обратна връзка и така да се създава интерактивно съдържание. Могат да се вграждат и скриптове написани на езици като JavaScript, което се отразява на „поведението“ на HTML страницата.

Чрез CSS, съкращение от Cascading Style Sheets (език за описание на стилове) може да се зададе по какъв начин да се изобразява HTML съдържанието.

[\[Wikipedia 1\]](#)

HTTP, съкращение от Hypertext Transfer Protocol (протокол за пренасяне на хипертекст) е протокол от приложния слой използван в разпределени,

хипермедийни информационни системи, най-вече в световната мрежа WWW (World Wide Web). HTTP е протокол от типа заявка-отговор, който е стандартен за клиент-сървър приложенията. При HTTP, браузърът най-често представлява клиента, а приложението работещо на машината, на която е сайтът представлява сървъра. Клиентът изпраща HTTP заявка към сървъра, който отговаря връщайки дадена страница или друг ресурс, като изображения например. [\[Wikipedia 2\]](#)

HTTP се нарича протокол **без запазване на състоянието**, защото всяка заявка се обработва независимо, без да е необходима информация за предишната. Това е главната причина, поради която е трудно да се направи уеб сайт, който да реагира интелигентно на потребителските входни данни. Този недостатък се решава чрез технологии като JavaScript и бисквитки (cookies). [\[Answers\]](#)

JavaScript е скриптов език разработен от Netscape, за да позволи създаването на интерактивни уеб сайтове. Въпреки че по възможности и структура прилича много на езика Java той е разработен независимо от него и има основни различия. JavaScript предоставя достъп и позволява да се **променя изходният код** на дадена страница, което прави възможно изграждането на динамично съдържание. Кодът се интерпретира от уеб браузъра, което означава, че се изпълнява на потребителската машина, а не на сървъра.

JavaScript се подкрепя от много софтуерни компании и е отворен език, който всеки може да използва без да е нужно закупуването на специален лиценз. Той се поддържа от повечето съвременни браузъри, като изключение прави този на Microsoft, който поддържа само подмножество на езика [\[Webopedia\]](#). Това е причина много ръководства за JavaScript да имат отделни секции за това как даден код се изпълнява от този браузър.

AJAX, съкращение от Asynchronous JavaScript and XML (асинхронен JavaScript и XML) представлява съвкупност от техники използвани при клиента (client-side) за създаване на **интерактивни** уеб приложения. С AJAX, браузърът може да изисква данни от сървъра асинхронно и незабележимо за потребителя без това да се отразява на изгледа и поведението на вече заредената страница. Използването на AJAX техниките води до увеличаване на интерактивността или динамичността на уеб страницата. Данните обикновено се получават, като се използва XMLHttpRequest (XHR) обект.

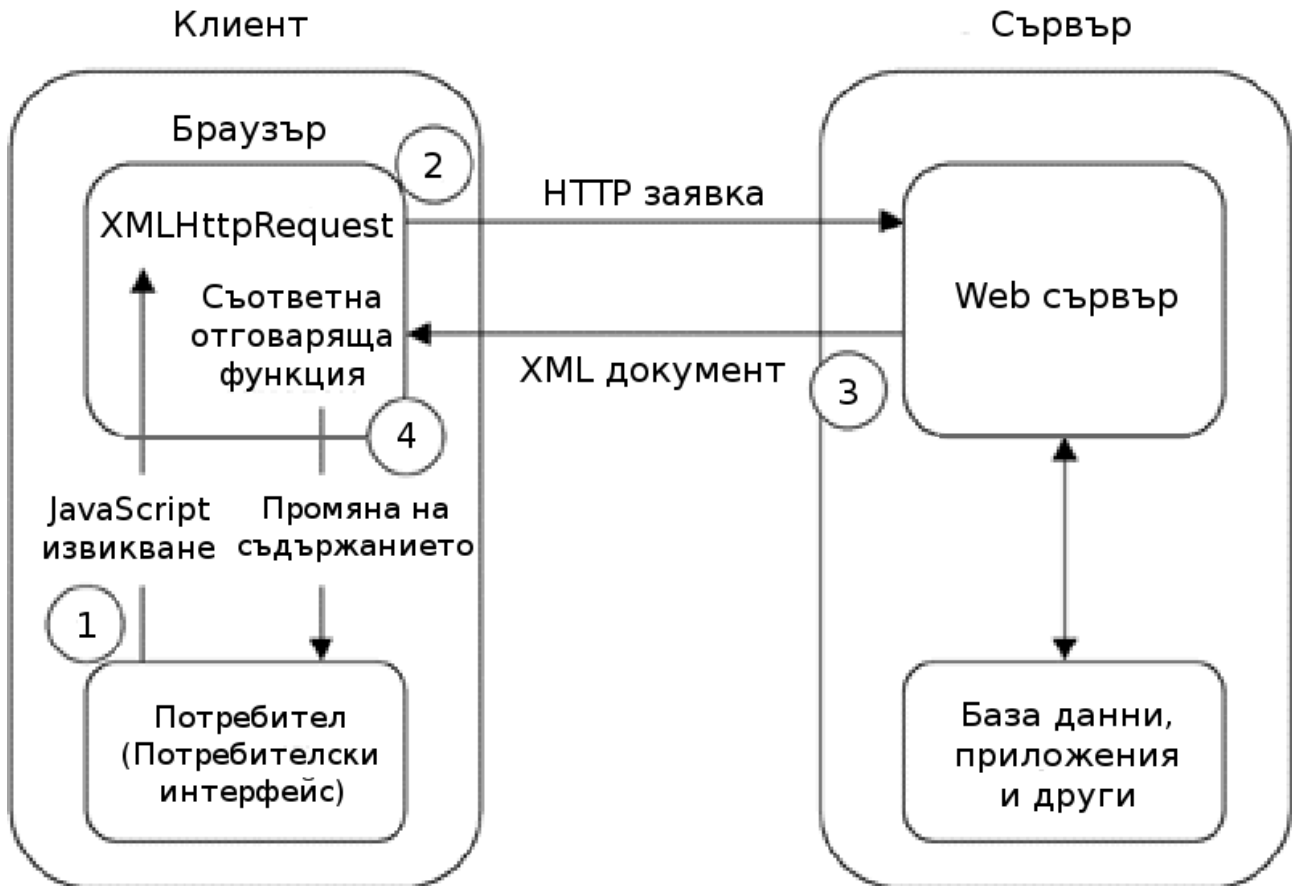
Въпреки името, използването на XML (Extensible Markup Language – разширен език за маркиране) всъщност не е нужно заявките да бъдат асинхронни.

Както DHTML (Dynamic HTML – динамичен HTML) и LAMP (Linux, Apache, MySQL и Perl/PHP/Python), AJAX не е технология сама по себе си, а група от технологии. Тя използва комбинация от HTML и CSS, за да маркира и форматира информацията. DOM (Document Object Model – обектен модел на документа), достъпен чрез JavaScript, се използва за динамично представяне на съдържанието и позволява на потребителя да взаимодейства с него. AJAX и

XMLHttpRequest обектът предоставят начин за обмяна на данни асинхронно между браузъра и сървъра без да се налага презареждане на цялата страница.

[\[Wikipedia 3\]](#)

На следващата схема е показано как работи AJAX.



Илюстрация 3: Как работи AJAX

3.1.2. Описание на клиентската част

Клиентската част е написана на HTML и JavaScript и представлява интерфейс за потребителите към сървърната част. Състои се от поле за въвеждане на текст и поле за извеждане на текст от сървъра. Използвана е технологията AJAX за трансфер на данните между сървъра и брауъра на потребителя. Това позволява да се прехвърлят само нужните данни без да се презарежда цялата страница, което води до по-бързото получаване на отговор от сървъра.

Прочетете упътването по-долу и отговорете на въпроса:
Каква е темата?

Изберете си някаква лекция, на която сте били или урок, който сте прочели скоро и отговаряйте на въпросите за него.
Ако не можете да отговорите на някой въпрос или да посочите това, което се изисква напишете „не знам“.
Въпросите са около 25 (зависи как ще се развие разговорът).
Пишете българските думи на кирилица. Спазвайте точния правопис. И бъдете толерантни към бота, малко е глупав.
Накрая той ще ви покаже семантична карта на част от вашите знания.
Ако искате да започнете отначало щракнете тук -> [започни отначало](#)

За да работи ботът трябва да ползвате нормален браузър. Примерно [Firefox](#).

Илюстрация 4: Потребителски интерфейс

Един разговор между двама души протича **последователно**, като се използва информацията обменена при предишните реплики. За да се реализира това в уеб среда, като се използва HTTP протоколът, който, както е описано по-горе не запазва предишни състояния е нужно използването на бисквитки. Чрез тях при започването на нов разговор се започва **сесия**, като в бисквитката се записва уникалният номер на сесията. При всяко следващо обръщение на клиентската част, към сървърната се предава номерът на сесията, по който сървърът може да разбере от кой потребител идва заявката. Всички номера на сесии се пазят на сървъра за определено време, след което сесията се смята за изтекла и трябва да се започне нова. Когато това става, сървърът създава нов уникален номер и го връща на клиенти. Всички заявки към сървъра, обработването на отговорите, както и записването и прочитането на бисквитки става посредством **JavaScript**. Началната страница се зарежда само веднъж и заедно с нея се зарежда и започва изпълнението на скрипта.

При натискане на бутона Enter, от страна на потребителя, входният текст заедно с номерът на сесията се изпращат към сървъра и се изобразява анимация, показваща, че се очаква отговор. Сървърът отговаря с „wait“ ако е приел

правилно заявката или с „ERROR“ ако е възникнала грешка. Ако отговорът на сървъра не може да бъде разбран от клиентския скрипт се изписва съобщение „BAD RESPONSE“.

При отговор „wait“ се преминава в състояние на очакване на отговор. През определен **интервал от време** се отправя заявка към сървъра със запитване дали е готов с отговора. Ако не е, той отговаря отново с „wait“. Това може да се повтори определен брой пъти, след което се изписва съобщение „Сървърът не отговаря. Опитайте пак по-късно.“.

Когато сървърът е готов с отговора той изпраща или новия въпрос, или „Expected time: <секунди>“, или „end“. Ако е изпратил новия въпрос, клиентският скрипт задава въпроса към потребителя. Ако е изпратил „Expected time: <секунди>“, това означава, че в момента се създава семантична карта от разговора и се очаква това да стане за време <секунди> секунди. В този случай на потребителя се показва съобщение:

Генерира се семантичната карта. Това може да отнеме повече от <секунди> секунди.

Моля изчакайте...

Ако изпратеният отговор е „end“ това означава, че семантичната карта е създадена. Клиентският скрипт, чрез номера на сесията, генерира пътя до картата (изображението), ако е необходимо го преоразмерява, за да се побере на екрана на потребителя и му го показва. С това разговорът приключва.

3.2. Сървърна част

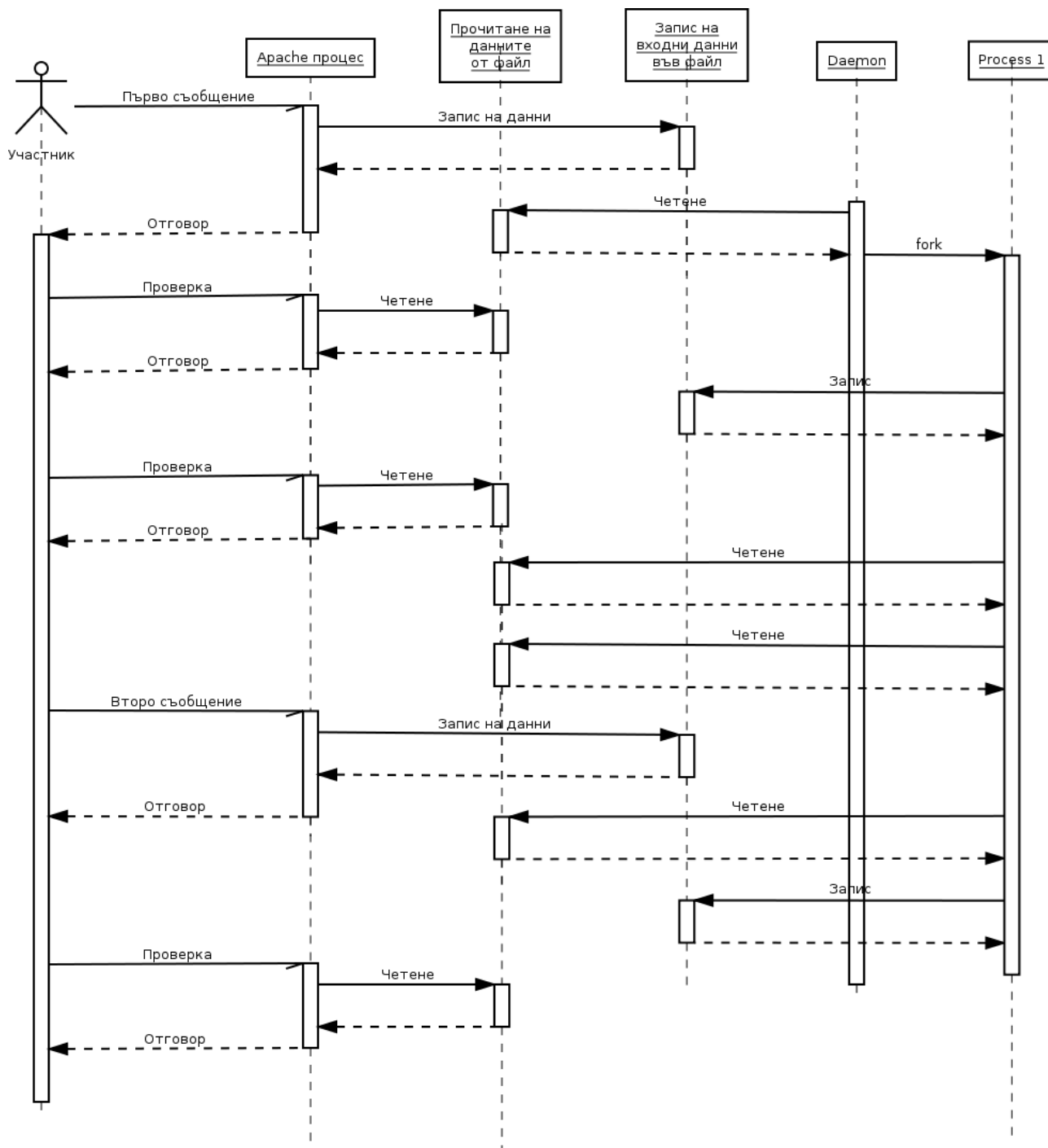
Сървърната част се състои от **два модула**. Първият **обработва клиентските заявки** и връща отговорите за тях. Той се активира от уеб сървъра, когато бъде получена заявка. Вторият модул е напълно независим от първия и представлява демон, работещ във фонов режим. В него е заложена **логиката на бота**. Двата модула комуникират посредством файлове.

Когато пристигне заявка от клиента, модулът обработващ заявките записва входните данни във файл или прочита изходните данни (от друг файл, преди това записани там от демона) и ги връща на клиента. В имената на файловете се съдържа номерът на клиентската сесия, което ги прави уникални. Демонът следи за нови файлове, и когато такъв бъде записан се разклонява **нов процес** (fork), който обработва получаваните от клиента данни. Той периодично прочита файла, в който се очакват те и когато се получат, изпълнението на процеса продължава. Запазва се времето на последната активност на клиента и ако от тогава е изминал по-голям от определен период от време, процесът се **самопрекратява**.

Благодарение на това кодът на въпросите, които задава ботът е последователен и се избягва разпокъсаността им.

Тази архитектура решава нелесната задача за последователност на разговора провеждан в уеб среда през HTTP протокола, който не запазва предишните състояния.

Описаното по-горе се вижда нагледно в диаграма 1.



Диаграма 1: Диаграма на събитията при провеждане на разговор

3.2.1. Perl – общ преглед

Perl е програмен език за обща употреба, първоначално разработен за **обработка на текст**, но използван в днешно време за широк кръг от задачи, като: системно администриране, уеб разработки, мрежово програмиране, разработка на графичен потребителски интерфейс (GUI – Graphical User Interface) и други. Езикът е замислен така, че да бъде повече **практичен** (лесен за употреба, ефективен, пълен), отколкото да бъде красив (строен и елегантен). Главните му предимства са леснотата при използване, поддръжката едновременно на **процедурно, обектно-ориентирано (ОО) и функционално** програмиране, вградените мощни инструменти за обработка на текст и една от най-големите колекции от модули в света. [\[Perl\]](#)

Perl е известен с това, че позволява да се пишат много кратки и стегнати програми, които решават големи и сложни проблеми.

3.2.2. Структури от данни в Perl

В Perl има три основни типа данни: скаларен, масив и хеш.

Скаларният представлява единична стойност:

```
my $animal = "камила";  
my $answer = 28;
```

Скаларен тип са низовете, целите числа или тези с плаваща запетая и Perl прави автоматично преобразуване между тях ако се налага. Няма нужда да декларирате променливите преди тяхната инициализация. Има редица магически скаларни променливи, като например променливата по подразбиране – `$_`. Тя се използва, като аргумент по подразбиране на редица функции в Perl и се инициализира неявно в циклични конструкции, като **foreach**.

Масивите представляват списък от стойности:

```
my @animals = ("камила", "мечка", "гарга");  
my @numbers = (6, 28, 496);  
my @mixed   = ("камила", 28, 1.23);
```

Специалната променлива `$#array` връща индекса на последния елемент от масива `array`:

```
print $mixed[$#mixed];    # отпечатва последния елемент 1.23
```


Ако се използва масив, там където Perl очаква скаларен тип данни (в скаларен контекст) ще бъде върнат броят на елементите на масива:

```
if (@animals < 5) { ... }
```

За достъп до част от масива могат да се използват следните записи:

```
@animals[0,1];           # връща ("камила", "мечка");  
@animals[0..2];         # връща ("камила", "мечка", "гарга");
```

Масивът може лесно да се сортира:

```
my @sorted = sort @animals;
```

Хешът представлява множество от ключ/стойност двойки или иначе казано асоциативен масив:

```
my %fruit_color = (  
    ябълка => "червена",  
    банан  => "жълт"  
);
```

Елемент се достъпва по следния начин:

```
$fruit_color{"ябълка"};           # връща "червена"
```

Могат да се вземат всички ключове или стойности от хеша по следния начин:

```
my @fruits = keys %fruit_colors;  
my @colors = values %fruit_colors;
```

Хешът няма вътрешна подредба.

По-сложни структури от данни могат да се конструират, като се използват указатели [\[Perl\]](#). Този подход се използва в бота при временно съхраняване на данните от разговора. Това е обяснено подробно по-долу.

3.2.3. Структури от данни използвани в бота

Както бе споменато по-горе, разговорът между двама души протича последователно, като се използва информацията обменена при предишните реплики. Ботът помни тази информация във временни структури от данни, и я използва за генериране на следващи въпроси. Едновременно с това събраните от разговора данни се записват в XML файл, като специалните символи

предварително се заменят с техния еквивалент в XML:

<	<
>	>
&	&
'	'
"	"

Примерен XML файл е този от [приложение 1](#).

Ето списък на структурите и за какво се използва всяка от тях:

@concepts	Масив с изброените от потребителя главни понятия
%concept_explanation	Хеш с ключ понятие и стойност обяснение на понятието – запазва обясненията на потребителя за всяко понятие
%concept_example	Хеш с ключ понятие и стойност пример за понятието – запазва дадените от потребителя примери за понятието
@procedures	Масив с изброените процедури
%procedure_steps	Хеш с ключ процедура и стойност друг хеш с ключ номер на стъпка и стойност обяснение на стъпката – запазва обяснените стъпка по стъпка процедури
@word_therefore	Масив от хешове, на които ключът е понятие, от което следва друго понятие – запазва причинно-следствените връзки между понятията
@relatedConcepts	Масив от понятия свързани с вече изброените
%concept_synonim	Хеш с ключ понятие и стойност синоним на понятието
%concept_category	Хеш с ключ понятие и стойност категорията на понятието
%procedureSolveProblems	Хеш с ключ процедура и стойност описание какви проблеми решава процедурата
%procedure_category	Хеш с ключ процедура и стойност категорията на процедурата
%procedureCircumstances	Хеш с ключ процедура и стойност масив от предусловията необходими за изпълнението на процедурата
%procedureEffects	Хеш с ключ процедура и стойност масив от последствията от изпълнението на процедурата
\$mainGoal	Низ с главната цел
%concept_application	Хеш с ключ понятие и стойност какви приложения има понятието
%procedure_application	Хеш с ключ процедура и стойност какви приложения има процедурата
@categoryHierarhy	Масив от хешове, на които ключът е категория, която е подгрупа на групата стойност – запазва йерархичните зависимости между категориите

@procedure_after_procedure	Масив от хешове, на които ключът е процедура, която се изпълнява след процедурата стойност – запазва времевите зависимости между процедурите
%concept_mark	Хеш с ключ понятие и стойност оценка на понятието – запазва подредба на понятията по значимост
%procedure_mark	Хеш с ключ процедура и стойност оценка на процедурата – запазва подредба на процедурите по значимост
\$onlyOneConcept	Низ с най-важното понятие
@more_words	Масив от думи, които би добавил потребителят
\$showCheckWords	Низ с обяснение как потребителят ще провери фактите
\$findMoreWords	Низ с обяснение къде потребителят ще потърси още знания и факти
\$findMoreProcedures	Низ с обяснение къде потребителят ще потърси нови процедури
\$everydayLife	Низ с обяснение как според потребителя се отразяват тези знанията на всекидневния живот
\$isomorphism	Низ с обяснение дали идеите и понятията са сходни на такива от друга област
\$anotherAge	Низ с обяснение какви последствия би имало ако хората от друга епоха са притежавали тези знания
\$knowledgePractise	Низ с обяснение какво би станало ако тези знания се приложат на практика
\$progress	Низ с обяснение за развитието на нещата в бъдеще
\$more_words_about	Низ с обяснение за какво липсват понятия и факти
%concept_property_value	Хеш с ключ понятие и стойност друг хеш с ключ свойство и стойност стойност на свойството – запазва свойствата на понятията и техните стойности

Таблица 3: Структури от данни използвани в бота

3.2.4. Регулярни изрази в Perl

Регулярният израз е шаблон, който описва множество от низове. Тъй като Perl е създаден от лингвист за **обработка на текстове** е разбираемо защо той е един от езиците с най-развити регулярни изрази. Тук е представен накратко част от синтаксиса им.

.	Разпознава един символ
\s	Разпознава празно място (интервал, табулатор, нов ред)
\S	Разпознава непразно място
\d	Разпознава цифра (0-9)
\D	Разпознава символ, който не е цифра
\w	Разпознава символ от дума (a-z, A-Z, 0-9, _)
\W	Разпознава символ, който не е от дума
[aeiou]	Разпознава, който и да е символ от даденото множество от символи
[^aeiou]	Разпознава символ, който не е от даденото множество от символи
(foo bar baz)	Разпознава, който и да е от посочените низове
^	Разпознава началото на низа
\$	Разпознава края на низа
*	Разпознава нула или повече повторения на израза отляво
+	Разпознава едно или повече повторения на израза отляво
?	Разпознава нула или едно повторение на израза отляво
{n}	Разпознава точно n на брой повторения на израза в ляво
{n,m}	Разпознава от n до m на брой повторения на израза в ляво
{n,}	Разпознава n на брой или повече повторения на израза в ляво

Таблица 4: Регулярни изрази в Perl

Кръглите скоби освен да обозначават група от алтернативни низове се използват и за извличане на разпознатия низ с цел по-нататъшното му използване. Низовете се извличат в променливи \$1, \$2 и т.н. където номерът означава номер на група отляво на дясно. Например, за да се извлекат потребителското име и домейнът от даден e-mail адрес може да се използва следният код:

```
if ($email =~ /^[^@+)]@(.+)/) {  
    print "Потребителското име е $1\n";  
}
```

```

    print "Домейнът е $2\n";
}

```

Регулярните изрази в Perl поддържат още много възможности, като обръщение към група в същия регулярен израз (**backreference**), **рекурсивни изрази**, коментари и **изпълним код** в регулярния израз и други.

Авторът на тази дипломна работа успешно е разработвал компилатор от Pascal до асемблерен език за NASM (Netwide Assembler), синтактичният анализатор, в който е изграден изцяло от Perl-ски регулярни изрази.

3.2.5. Регулярни изрази използвани в бота

<code>/^кра[йи]\.?\$</code>	Разпознава край на изброяване от потребителя
<code>/^(не\s*(знам мога) кра[йи])\$</code>	Разпознава незнание на потребителя относно въпрос
<code>/^(чао довиждане)\$</code>	Разпознава желание за прекратяване на разговора
<code>s/^\s+ \s+\$//g</code>	Премахва празни места от началото и края на низ
<code>/(темата е)(.*)/</code>	Разпознава отговор на въпроса „Каква е темата?“
<code>/^(да точно така именно)\$</code>	Разпознава съгласие на потребителя
<code>/(\$concept (е означава))(.*)/</code>	Разпознава обяснение на дадено понятие
<code>/([0-9]+):\s?(.*)/</code>	Разпознава изброяване на стъпки на процедури
<code>/от (.+) следва (.+)/</code>	Разпознава причинно-следствени връзки
<code>/((concept)попада в(група категория)(та))(.+)/</code>	Групата на понятието
<code>/((procedure се)използва (се)за)(.*)/</code>	Разпознава за какво служи процедура
<code>/((procedure)попада в(група категория)(та))(.+)/</code>	Групата на процедура
<code>/(главната цел е целта е най-важното е)(.*)/</code>	Разпознава главната цел
<code>/(((concept се)използва(се)) за)(.*)/</code>	За какво се използва понятието
<code>/((на)пример(но))(.*)/</code>	Разпознава посочен от потребителя пример
<code>/(.*) е подгрупа на (.+)/</code>	Разпознава йерархичната зависимост между групите
<code>/(+)\s+-\s+(.+)/</code>	Разпознава двойки свойство - стойност
<code>/(+)\s+-\s+(+)\s+-\s+(.+)/</code>	Разпознава понятия/процедури и връзката между тях

<code>/(.+)</code> се изпълнява след <code>(.+)</code> /	Разпознава времевите връзки между процедурите
<code> /^[1-9] 10)\$/</code>	Разпознава число от 1 до 10 включително
<code> /^(да не)\$/</code>	Разпознава отговор с „да“ или „не“
<code></session_id>(.*?)</session_id><waiting>([01])</waiting>(.*?)</x</code>	
Разпознава входните данни от клиента	
<code> /Expected time: [\.\d]+/</code>	Разпознава очакваното време за създаване на карта
<code> /^\./</code>	Разпознава файлове започващи с „.“
<code> /^[0-9]+_server_write\$/</code>	Разпознава номера на сесията от името на файл
<code> /(.*)=(.*)/</code>	Разпознава двойки ключ-стойност във входните данни
<code> s/\+/ /g</code>	Заменя знакът „+“ с интервал
<code> s/(.+)/pack('c',hex(\$1))/eg</code>	Декодира входните данни
<code> s/%2F/\//g</code>	Заменя %2F с /
<code> s/%3F/\?/g</code>	Заменя %3F с ?
<code> s/%3D/=/g</code>	Заменя %3D с =
<code> s/%26/\&/g</code>	Заменя %26 с &
<code> s/%40/\@/g</code>	Заменя %40 с @

Таблица 5: Регулярни изрази използвани в бота

3.2.6. ImageMagick – общ преглед

ImageMagick е колекция от инструменти и библиотеки за създаване, обработка и аранжиране на **растерни изображения** в широк кръг от формати. Той позволява да се изрязват, преоразмеряват, завъртат, мащабират, изострят изображения, да се променят цветовете им или да се добавят ефекти, текст, прави линии, полигони, елипси или криви на Безие. Това може да се прави както с едно така и с поредица от изображения (batch processing – **пакетна обработка**). Могат да се създават и напълно нови изображения. Такъв е случаят с генерирането на семантични карти. Резултатът има възможност да бъде запазен в над сто файлови формата включително: DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PhotoCD, PNG, Postscript, SVG, и TIFF. ImageMagick обикновено се използва от командния ред, но също така има интерфейси за различни езици включително: Magick++ за C++, JMagick за Java, L-Magick за Lisp, IMagick за PHP и **PerlMagick** за Perl, който се използва при генерирането на семантичните карти в тази дипломна работа. [\[ImageMagick\]](#)

ImageMagick е свободен софтуер, предоставян, както компилиран и готов за използване, така и като изходен код, който може свободно да се използва, копира променя и разпространява в приложения с отворен и затворен код. Той е лицензиран под собствен лиценз. [\[ImageMagick License\]](#)

Главните инструменти на ImageMagick са следните:

- **Convert** – Преобразува изображението от един файлов формат в друг, като едновременно с това може да се приложат различни филтри или да се добавят примитиви, като линии, текст и други;
- **Import** – Прави снимка на екрана и я преобразува до файл. Могат да се зададат отделни прозорци, целият екран или правоъгълна част от него;
- **Animate** – Създава анимации;
- **Composite** – Позволява комбиниране на няколко отделни изображения;
- **Montage** – Подрежда група от изображения в едно, като може да приложи ефекти, като рамка и сянка към всяко от тях;
- **Mogrify** – Използва се главно за преобразувания, като мащабиране, завъртане, премахване на цветове и други. Главната разлика между Mogrify и другите инструменти е, че той записва резултата върху оригиналното изображение;
- **Conjure** – Предоставя възможност да се използва XML базираният

скриптов език Magick (Magick Scripting Language – MSL) вместо Perl, C++ и т.н. Желаните манипулации се написват на MSL и се извиква Conjure, за да ги изпълни;

- **Identify** – Извлича информация за изображението, като формата му, името и големината на файла, размера на изображението, цветовете и други. Може да установи дали изображението е повредено. [\[ImageMagick 2\]](#)

3.2.7. Генериране на семантична карта

PerlMagick е обектно ориентиран интерфейс в Perl към ImageMagick. С него могат да се използват по-голямата част от инструментите предлагани от ImageMagick.

Модулът, който визуализира семантичната карта, първо прочита XML файла създаден при разговора с потребителя, а после генерира самото изображение използвайки библиотеката PerlMagick. Прочетените данни се съхраняват в структури подобни на тези използвани от бота, които бяха описани по-горе.

При разполагането на обектите и връзките между тях се използват различни евристики, с цел разположението да бъде най-близко до оптималното. Можем да приемем семантичната карта за ориентиран, претеглен граф, в който теглата са низове свързващи обектите. На кратко нещата протичат по следния начин:

- Намира се върхът, от който излизат и влизат най-много ребра (сумата на входящата и изходящата степен (или просто степента на върха) е най-голяма) и той се поставя в центъра на картата;
- Върховете се сортират по степен. Половината от тях с по-голяма степен се поставят във вътрешен кръг, а другата половина с по-малка степен се поставят във външен кръг. Това се вижда добре от [илюстрация 6](#).
- Спрямо броят на върховете и дължината на низовете записани в тях се пресмята какъв трябва да бъде размерът на изображението;
- Изрисуват се върховете и за всеки от тях се запомнят координатите му. Върховете са елипси, размерът на които също е съобразен с дължината на низа записан в тях;
- Изрисуването на ребрата се осъществява чрез запомнените координати за всеки един от върховете. Изчислява се в каква посока сочи реброто, след

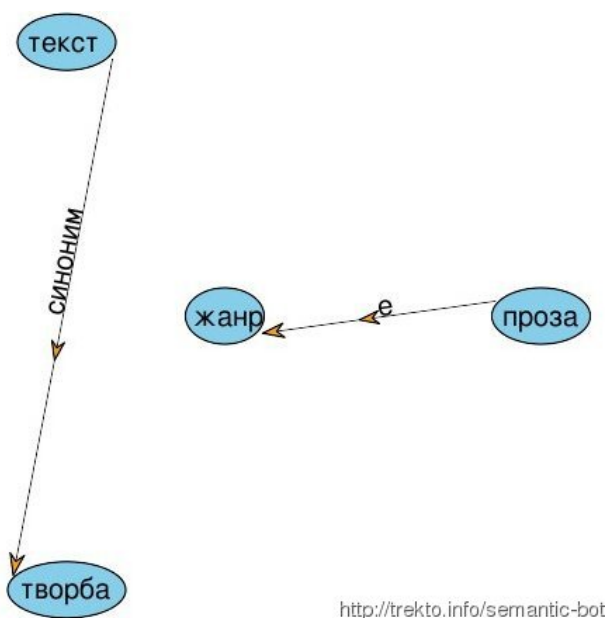
което стрелката и низът - тегло на реброто се завъртат по подходящ начин. Разбира се всичко това се реализира чрез използване на тригонометрични функции.

В зависимост от посоката, ребрата се преместват по подходящ начин, за да не закриват низа във върха.

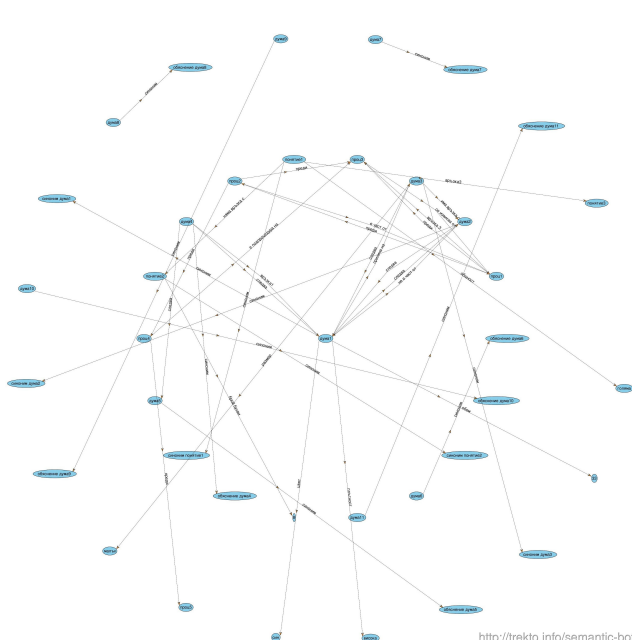
Ако между два върха има повече от една връзка линиите се начупват, така че да се заобикалят и да се вижда низът – тегло.

Поради действията извършвани по-горе е възможно в някои случаи обектите в генерираните изображения да не са ясно различими или даден текст да бъде труден за прочитане.

Също така се използва различно съотношение между броя на обектите и размера на изображението с цел по-ясно представяне. Това може да се види на следващите две изображения.



Илюстрация 5: Карта с няколко обекта



Илюстрация 6: Карта с много обекти

4. Резултати

Проведен бе експерименти със студенти от Факултета по математика и информатика към Пловдивския университет „Паисий Хилендарски“. Второкурсници, третокурсници и четвъртокурсници от специалност Информатика си поговориха с бота, за последните си лекции през учебната 2009-2010 година, съответно по дисциплините Операционни системи, Изкуствен интелект и Управление на проекти. Повечето от тях не завършиха целия разговор, вероятно отегчени от темите. С бота е разговаряно и на други теми. Някои от създадените автоматично семантични карти се виждат в [приложение 2](#).

5. Заключение

5.1. Какво е постигнато

Създаден е вариант на преработената класификация на Блум с някои нови идеи – [таблица 2](#). При нейната направа е наблегнато на въпросите, които могат да се използват за събиране на информация за дадена ПО.

Създаден е бот, който разпитва потребителя и по този начин събира информацията.

Той е достъпен онлайн на адрес: <http://trekto.info/semantic-bot>

Програмиран е модул, който създава изображение от семантична карта в XML формат. Примерна такава е поместена в [приложение 1](#).

Проведени са експерименти със студенти, като част от резултатите са поместени в [приложение 2](#).

5.2. Проблеми в процеса на работа

Първият главен проблем беше свързан с въпросите, които да се задават, за да се извлече информация за предметната област. Те трябваше да бъдат достатъчно общи и добре формулирани, за да са приложими за най-различни предметни области. Решението на този проблем се намира в [таблица 2](#).

Втората нелесна задача беше постигането на последователност на разговора провеждан в уеб среда през HTTP протокола, който не запазва предишните състояния. Това беше постигнато чрез добре обмислената архитектура на системата представена в дипломната работа.

Друг проблем беше генерирането на изображение на семантичната карта и по-конкретно разположението на обектите и връзките по подходящ начин. Този проблем се решава, но не напълно чрез някои евристики съобразени при анализ на разговори на бота с потребители.

По-лесно решими проблеми бяха: създаване на подходящ формат за извлечената информация, комуникацията между модулите на системата и други.

5.3. Области от информатиката, които са засегнати

- Компютърна лингвистика;
- Компютърна графика;
- Уеб програмиране;
- Паралелно програмиране;
- Софтуерно инженерство.

5.4. Бъдещо развитие

Някои възможности за подобряване на системата са следните:

- Предложение от няколко възможни отговора на даден въпрос, ако потребителят е отговорил с „не знам“. Предложенията могат да бъдат генерирани от отговорите на същия въпрос, за същата тема от други потребители. Отговори на въпрос могат да се предлагат и със самото му задаване, като само един от тях е верен, а останалите са генерирани от грешни отговори на предишни потребители;
- Създаване на преобразувател от семантична карта към тематична карта по стандарта ISO/IEC 13250:2003 или към такава описана чрез RDF (Resource Description Framework – Рамка за описание на ресурси) и OWL (Web Ontology Language – Език за уеб онтологии);
- Създаване на модул за сравняване близостта на две семантични карти.
- Превеждане на бота на други езици.

С настъпването на следващата учебна година те първа системата ще има възможност да се тества от ученици и студенти.

Литература

[**Костова**] Доц. Здравка Костова – *Как да учим успешно? (Иновации в обучението)*.
<http://xoomer.virgilio.it/pavelnik/Biblio/Pedagog/Kakdauczim/Kakdauczim10.htm> 28 юни 2010 год.

[**Тотков**] Георги Тотков – *Концептуално и компютърно моделиране на езикови структури и процеси (с приложения за българския език)*. Автореферат на дисертация за присъждане на научна степен „Доктор на математическите науки“, Пловдив 2004

[**Krathwohl**] David R. Krathwohl – *A Revision of Bloom's Taxonomy: An Overview*.

[**Sowa**] John F. Sowa – *Semantic Networks*. <http://www.jfsowa.com/pubs/semnet.htm> 28 юни 2010 год.

[**Уикипедия**] http://bg.wikipedia.org/wiki/Мисловна_карта 28 юни 2010 год.

[**Answers**] http://wiki.answers.com/Q/What_is_HTML_stateless_protocol 22 юни 2010 год.

[**Cognition**] <http://www.cognition.com/info/what.html> 3 юли 2010 год.

[**ImageMagick License**] <http://www.imagemagick.org/script/license.php> 25 юни 2010 год.

[**ImageMagick**] <http://www.imagemagick.org/script/index.php> 25 юни 2010 год.

[**ImageMagick 2**] <http://www.webhostingart.com/blog/imagemagick-features.html> 25 юни 2010 год.

[**Perl**] <http://perldoc.perl.org/perlintro.html> 23 юни 2010 год.

[**Webopedia**] <http://www.webopedia.com/TERM/J/JavaScript.html> 22 юни 2010 год.

[**Wikipedia 1**] <http://en.wikipedia.org/wiki/HTML> 22 юни 2010 год.

[**Wikipedia 2**] <http://en.wikipedia.org/wiki/Http> 22 юни 2010 год.

[**Wikipedia 3**] [http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)) 22 юни 2010 год.

[**Wikipedia 4**] http://en.wikipedia.org/wiki/Semantic_network 28 юни 2010 год.

[**Wikipedia 5**] http://en.wikipedia.org/wiki/Topic_map 28 юни 2010 год.

[**Wikipedia 6**] http://en.wikipedia.org/wiki/Mind_map 28 юни 2010 год.

[**Wikipedia 7**] <http://en.wikipedia.org/wiki/Lexipedia> 1 юли 2010 год.

[**Wikipedia 8**] http://en.wikipedia.org/wiki/SNOMED_CT 1 юли 2010 год.

[**Wikipedia 9**] http://en.wikipedia.org/wiki/SNePS#SNePS_as_a_Network-Based_System 1 юли 2010 год.

[**Wikipedia 10**] <http://en.wikipedia.org/wiki/Chatterbot> 2 юли 2010 год.

[**Wikipedia 11**] <http://en.wikipedia.org/wiki/ELIZA> 2 юли 2010 год.

[**Wikipedia 12**] http://en.wikipedia.org/wiki/Concept_maps 28 юни 2010 год.

[**Анонимен**] <http://bglog.net/ClientFiles/41c49fb0-bfe3-4301-9d2f-fe27f3630898/Таксономия%20на%20Блум%20за%20целите%20на%20образованието.doc> 1 юли 2010 год.

Списък на изображенията и таблиците

Указател на илюстрациите

Илюстрация 1: Тематична карта.....	9
Илюстрация 2: Мисловна карта.....	10
Илюстрация 3: Как работи AJAX.....	19
Илюстрация 4: Потребителски интерфейс.....	20
Диаграма 1: Диаграма на събитията при провеждане на разговор.....	22
Илюстрация 5: Карта с няколко обекта.....	32
Илюстрация 6: Карта с много обекти.....	32

Указател на таблиците

Таблица 1: Оригинална класификация на Блум.....	13
Таблица 2: Преведена и допълнена, преработена класификация на Блум.....	15
Таблица 3: Структури от данни използвани в бота.....	25
Таблица 4: Регулярни изрази в Perl.....	27
Таблица 5: Регулярни изрази използвани в бота.....	28

Приложение 1 – пример

Разговор

16:17:00: Бот: Каква е темата?

16:17:16: Човек: структура от данни стек

16:17:16: Бот: Значи темата е „структура от данни стек“?

16:17:19: Човек: да

16:17:19: Бот: Избройте главните понятия. Разделете ги със запетаи.

16:20:55: Човек: структура от данни, LIFO, съхранение

16:20:55: Бот: Какво означава „структура от данни“?

16:21:17: Човек: структура, в която се съхраняват някакви данни

16:21:17: Бот: Дайте пример за „структура от данни“.

16:21:32: Човек: стек

16:21:32: Бот: Какво означава „LIFO“?

16:22:03: Човек: Last In First Out - последен влязъл първи излязъл

16:22:03: Бот: Дайте пример за „LIFO“.

16:23:23: Човек: Влиза доц. Донева, влиза проф. Тотков, влиза доц. Шкуртов. Излиза доц. Шкуртов, излиза проф. Тотков, излиза доц. Донева.

16:23:23: Бот: Какво означава „съхранение“?

16:23:42: Човек: запазване за известно време на нещо

16:23:42: Бот: Дайте пример за „съхранение“.

16:24:06: Човек: запазване на временна променлива в стек

16:24:06: Бот: Избройте действията/процедурите, които сте запомнили. Разделете ги със запетаи.

Пример за процедура: „Намиране на най-голям общ делител на две числа“.

16:24:32: Човек: добавяне на елемент в стека, премахване на елемент от стека

16:24:32: Бот: Опишете процедурата „добавяне на елемент в стека“ стъпка по стъпка.

На първия ред запишете „1: описание на първа стъпка“, на втория „2: описание на втора стъпка“ и т.н. докато ' ' изброите всички стъпки.

Когато сте готови напишете „край“.

16:25:37: Човек: 1: насочваме временен указател към върха на стека

16:25:48: Човек: 2: заделяме памет за новия елемент и насочваме върха на стека към него

16:26:24: Човек: 3: съхраняваме данните в новия елемент

16:28:30: Човек: 4: задаваме следващия ел. на новия да е старият връх

16:29:37: Човек: край

16:29:37: Бот: Опишете процедурата „премахване на елемент от стека“ стъпка по стъпка.

На първия ред запишете „1: описание на първа стъпка“, на втория „2: описание на втора стъпка“ и т.н. докато ' ' изброите всички стъпки.

Когато сте готови напишете „край“.

16:30:18: Човек: 1: вземаме стойността на върха

16:30:53: Човек: 2: насочваме временен указател към върха

16:31:27: Човек: 3: насочваме върха към елемента след него

16:31:51: Човек: 4: изтриваме временния указател

16:31:54: Човек: край

16:31:54: Бот: Вие изброихте понятията: „структура от данни, LIFO, съхранение“
За какви други понятия свързани с изброените се сещате?
Разделете ги със запетаи.

16:33:36: Човек: алгоритъм, FIFO, компютър, информация

16:33:36: Бот: Вие обяснихте „структура от данни“ по следния начин:
структура, в която се съхраняват някакви данни
Дайте синоним или обяснете с други думи „структура от данни“.

16:39:15: Човек: формат за съхраняване на информация

16:39:15: Бот: Вие обяснихте „LIFO“ по следния начин:
Last In First Out - последен влязъл първи излязъл
Дайте синоним или обяснете с други думи „LIFO“.

16:39:27: Човек: FILO

16:39:27: Бот: Вие обяснихте „съхранение“ по следния начин:
запазване за известно време на нещо
Дайте синоним или обяснете с други думи „съхранение“.

16:40:18: Човек: консервиране

16:40:18: Бот: Обособете понятията в групи/категории.
В каква група попада „структура от данни“?

16:40:49: Човек: формат

16:40:49: Бот: Обособете понятията в групи/категории.
В каква група попада „LIFO“?

16:41:03: Човек: правило

16:41:03: Бот: Обособете понятията в групи/категории.
В каква група попада „съхранение“?

16:41:13: Човек: процедура

16:41:13: Бот: Какви проблеми решава процедурата „добавяне на елемент в стека“?

16:42:00: Човек: съхраняването на информация на върха на структура от данни

16:42:00: Бот: Какви проблеми решава процедурата „премахване на елемент от стека“?

16:42:39: Човек: достъп до и изтриването на елемент от структурата от данни

16:42:39: Бот: Обособете процедурите в групи/категории.
В каква група попада „добавяне на елемент в стека“?

16:43:32: Човек: трансформатори

16:43:32: Бот: Обособете процедурите в групи/категории.
В каква група попада „премахване на елемент от стека“?

16:43:36: Човек: трансформатори

16:43:36: Бот: Какви са условията необходими за прилагането на процедурата „добавяне на елемент в стека“, т.е. предусловията?

Избройте ги със запетаи помежду им.

16:44:44: Човек: стекът да съществува, стекът да не е препълнен

16:44:44: Бот: Какви са условията необходими за прилагането на процедурата „премахване на елемент от стека“, т.е. предусловията?

Избройте ги със запетаи помежду им.

16:45:03: Човек: стекът да съществува, стекът да не е празен

16:45:03: Бот: Какви са последствията от прилагането на процедурата „добавяне на елемент в стека“?
Избройте ги със запетай помежду им.

16:45:42: Човек: на върха на стека е запазен нов елемент, размерът на стека е увеличен с 1

16:45:42: Бот: Какви са последствията от прилагането на процедурата „премахване на елемент от стека“?
Избройте ги със запетай помежду им.

16:46:05: Човек: премахнат е елементът на върха на стека, размерът на стека е намален с 1

16:46:05: Бот: Кое е най-важното от темата? Каква е главната цел?

16:47:19: Човек: съхранение на данни и използването им по правилото LIFO

16:47:19: Бот: За какво може да се използва, какви приложения има понятието „структура от данни“?

16:51:41: Човек: за съхранение на информация, което пък може да се използва, като временно действие при различни алгоритми

16:51:41: Бот: За какво може да се използва, какви приложения има понятието „LIFO“?

16:52:41: Човек: за назоваване на определено правило на вкарване и изкарване на елементи от структура от данни и други обекти

16:52:41: Бот: За какво може да се използва, какви приложения има понятието „съхранение“?

16:53:14: Човек: ползва се за запомняне на информация, която по-късно ще бъде нужна

16:53:14: Бот: Демонстрирайте, дайте пример за прилагане на процедурата „добавяне на елемент в стека“?

16:55:31: Човек: добавяне на числата 2 и 3 в стек: `stack.push(2); stack.push(3);`

16:55:31: Бот: Демонстрирайте, дайте пример за прилагане на процедурата „премахване на елемент от стека“?

16:56:31: Човек: след изпълнение на `stack.pop(a); a = 3`, след следващо изпълнение на `stack.pop(a); a = 2`

16:56:31: Бот: Избройте следните групи: „формат, процедура, правило, трансформатори“
Избройте на нов ред тези групи, между които има йерархична зависимост по следния начин: група1 е подгрупа на група2.
Когато сте готови напишете „край“.

16:57:15: Човек: формат е подгрупа на правило

16:57:23: Човек: край

16:57:23: Бот: Избройте на нов ред свойствата на „структура от данни“, и тяхната стойност по следния начин: свойство - стойност. Примерно:
цвет - зелен
височина - 5 см.
Когато сте готови напишете „край“.

16:58:52: Човек: използваемост - висока

16:58:59: Човек: полезност - голяма

16:59:02: Човек: край

16:59:02: Бот: Избройте на нов ред свойствата на „LIFO“, и тяхната стойност по следния начин: свойство - стойност. Примерно:
цвет - зелен
височина - 5 см.
Когато сте готови напишете „край“.

17:00:43: Човек: подредба - обратна

17:00:46: Човек: край

17:00:46: Бот: Избройте на нов ред свойствата на „съхранение“, и тяхната стойност по следния начин: свойство - стойност. Примерно:

цвят - зелен
височина - 5 см.

Когато сте готови напишете „край“.

17:02:37: Човек: оценка - безценно

17:02:39: Човек: край

17:02:39: Бот: Вие изброихте следните понятия: „структура от данни, LIFO, съхранение“

Какви връзки съществуват между тях?

Избройте тези, между които има връзка по следния начин: понятие1 - връзка - понятие2.

Когато сте готови напишете „край“.

Някои възможни връзки са: „е“, „нужен за“, „е част от“. Примерно

праскова - е - плод

праскова - е част от - дърво

светлина - нужен за - растеж

17:03:13: Човек: структура от данни - нужен за - съхранение

17:10:15: Човек: LIFO - обратно на - FIFO

17:11:07: Човек: добавяне на елемент - метод на - клас Стек

17:11:15: Човек: премахване на елемент - метод на - клас Стек

17:11:48: Човек: клас Стек - реализация на - стек

17:19:15: Човек: компютър - изпълнява - алгоритъм

17:19:47: Човек: информация - се съхранява в - структура от данни

17:19:50: Човек: край

17:19:50: Бот: Вие изброихте следните процедури: „добавяне на елемент в стека, премахване на елемент от стека“

Какви са времевите връзки между тях? Избройте тези, между които има връзка по следния начин: процедура1 се изпълнява след процедура2.

Когато сте готови напишете „край“.

17:20:35: Човек: премахване на елемент се изпълнява след добавяне на елемент

17:20:35: Бот: Не сте посочили такава процедура „премахване на елемент“!

17:20:48: Човек: премахване на елемент от стека се изпълнява след добавяне на елемент в стека

17:20:55: Човек: край

17:20:55: Бот: Какви други връзки има между процедурите: „добавяне на елемент в стека, премахване на елемент от стека“

Някои възможни връзки са „е част от“, „се извиква от“

Избройте тези, между които има връзка по следния начин: процедура1 - връзка - процедура2.

Когато сте готови напишете „край“.

17:21:13: Човек: край

17:21:13: Бот: Подредете по значимост понятията, като им поставите числова оценка.

Оценете от 1 до 10 „структура от данни“

17:22:00: Човек: 9

17:22:00: Бот: Подредете по значимост понятията, като им поставите числова оценка.

Оценете от 1 до 10 „LIFO“

17:22:13: Човек: 8

17:22:13: Бот: Подредете по значимост понятията, като им поставите числова оценка.

Оценете от 1 до 10 „съхранение“

17:22:16: Човек: 10

17:22:16: Бот: Ако трябва да оставите само едно понятие, една идея и да изхвърлите останалите, кое ще бъде то?

17:22:39: Човек: LIFO

17:22:39: Бот: Подредете по значимост процедурите, като им поставите числова оценка.

Оценете от 1 до 10 „добавяне на елемент в стека“

17:22:45: Човек: 5

17:22:45: Бот: Подредете по значимост процедурите, като им поставите числова оценка.

Оценете от 1 до 10 „премахване на елемент от стека“

17:22:47: Човек: 5

17:22:47: Бот: Вие бихте ли използвали процедурата „добавяне на елемент в стека“ (отговорете с да или не)

17:22:55: Човек: да

17:22:55: Бот: Защо?

17:24:24: Човек: защото ми помага да се абстрахирам от реализацията на добавянето и да се съсредоточа върху същността на програмата

17:24:24: Бот: Вие бихте ли използвали процедурата „премахване на елемент от стека“ (отговорете с да или не)

17:24:26: Човек: да

17:24:26: Бот: Защо?

17:24:28: Човек: защото ми помага да се абстрахирам от реализацията на добавянето и да се съсредоточа върху същността на програмата

17:24:28: Бот: Какви термини и понятия (за какво) липсват или са малко?

17:25:46: Човек: за приложенията на стека

17:25:46: Бот: Какви термини и понятия бихте добавили вие? Разделете ги със запетаи.

17:26:24: Човек: приложение, конкретни примери, недостатъци

17:26:24: Бот: Как бихте проверили посочените факти?

17:26:43: Човек: с компилатор

17:26:43: Бот: Къде ще потърсите още знания и факти?

17:27:12: Човек: в свободната енциклопедия wikipedia.org

17:27:12: Бот: Къде ще потърсите нови процедури?

17:27:49: Човек: в учебник по Алгоритми и структури от данни

17:27:49: Бот: Как се отразяват тези знанията на всекидневния живот?

17:28:14: Човек: много дейности се автоматизират

17:28:14: Бот: Може ли да се счита, че идеите и понятията са изоморфни (сходни) на такива от други области? Кой?

17:30:19: Човек: начините на вкарване и изкарване на елементи (LIFO, FIFO) имат изобилие от примери в обществото и природата. Самата област, като че ли няма сходна.

17:30:19: Бот: Опитайте се да си представите и напишете какви последствия би имало ако хората от друга епоха са притежавали тези знания.

17:31:31: Човек: това би повлияло на организацията им на живот

17:31:31: Бот: Предскажете какво би станало ако тези знания се приложат на практика.

17:33:16: Човек: биха могли да се автоматизират различни дейности

17:33:16: Бот: Опитайте се да проектирате развитието на нещата в бъдеще.

17:34:23: Човек: много дейности ще се извършват от машини

Времетраене: 01:17:23

Извлечена от разговора информация

```
<semantic-map>
<starttime>21.юни.2010 - 16:17:00</starttime>
<theme>структура от данни стек</theme>
<structure name="понятия">
  <element>структура от данни</element>
  <element>LIFO</element>
  <element>съхранение</element>
</structure>

<structure name="понятие - обяснение">
  <pair><key>структура от данни</key>    <value>структура, в която се
съхраняват някакви данни</value></pair>
  <pair><key>съхранение</key>    <value>запазване за известно време на
нещо</value></pair>
  <pair><key>LIFO</key>    <value>Last In First Out - последен влязъл първи
излязъл</value></pair>
</structure>

<structure name="понятие - пример">
  <pair><key>структура от данни</key>    <value>стек</value></pair>
  <pair><key>съхранение</key>    <value>запазване на временна променлива
в стек</value></pair>
  <pair><key>LIFO</key>    <value>Влиза доц. Донева, влиза проф. Тотков,
влиза доц. Шкуртов. Излиза доц. Шкуртов, излиза проф. Тотков, излиза доц.
Донева.</value></pair>
</structure>

<structure name="процедури">
  <element>добавяне на елемент в стека</element>
  <element>премахване на елемент от стека</element>
</structure>

<structure name="процедура - стъпки">
  <procedure name="добавяне на елемент в стека">
    <pair><key>4</key>    <value>задаваме следващия ел. на новия
да е старият връх</value></pair>
    <pair><key>1</key>    <value>насочваме временен указател към
върха на стека</value></pair>
    <pair><key>3</key>    <value>съхраняваме данните в новия
елемент</value></pair>
    <pair><key>2</key>    <value>заделяме памет за новия елемент и
насочваме върха на стека към него</value></pair>
    <pair><key>5</key>    <value>насочваме върха към новия
елемент</value></pair>
  </procedure>

  <procedure name="премахване на елемент от стека">
    <pair><key>4</key>    <value>изтриваме временния
указател</value></pair>
    <pair><key>1</key>    <value>взимаме стойността на
върха</value></pair>
    <pair><key>3</key>    <value>насочваме върха към елемента след
него</value></pair>
    <pair><key>2</key>    <value>насочваме временен указател към
върха</value></pair>
  </procedure>
</structure>
```

```

    </procedure>
</structure>

<structure name="понятия свързани с вече изброените">
    <element>алгоритъм</element>
    <element>FIFO</element>
    <element>компютър</element>
    <element>информация</element>
</structure>

<structure name="понятие - синоним">
    <pair><key>структура от данни</key>    <value>формат за съхраняване на
информация</value></pair>
    <pair><key>съхранение</key>    <value>консервиране</value></pair>
    <pair><key>LIFO</key>    <value>FILO</value></pair>
</structure>

<structure name="понятие - категория">
    <pair><key>структура от данни</key>    <value>формат</value></pair>
    <pair><key>съхранение</key>    <value>процедура</value></pair>
    <pair><key>LIFO</key>    <value>правило</value></pair>
</structure>

<structure name="процедура - решава проблеми">
    <pair><key>добавяне на елемент в стека</key>    <value>съхраняването на
информация на върха на структура от данни</value></pair>
    <pair><key>премахване на елемент от стека</key> <value>достъп до и
изтриването на елемент от структурата от данни</value></pair>
</structure>

<structure name="процедура - категория">
    <pair><key>добавяне на елемент в стека</key>
<value>трансформатори</value></pair>
    <pair><key>премахване на елемент от стека</key>
<value>трансформатори</value></pair>
</structure>

<structure name="процедура - необходими условия">
    <structure name="добавяне на елемент в стека">
        <element>стекът да съществува</element>
        <element>стекът да не е препълнен</element>
    </structure>

    <structure name="премахване на елемент от стека">
        <element>стекът да съществува</element>
        <element>стекът да не е празен</element>
    </structure>
</structure>

<structure name="процедура - последствия">
    <structure name="добавяне на елемент в стека">
        <element>на върха на стека е запазен нов елемент</element>
        <element>размерът на стека е увеличен с 1</element>
    </structure>

    <structure name="премахване на елемент от стека">
        <element>премахнат е елементът на върха на стека</element>
        <element>размерът на стека е намален с 1</element>
    </structure>
</structure>

```

<mainGoal>съхранение на данни и използването им по правилото LIFO</mainGoal>

<structure name="понятие - приложение">

<pair><key>структура от данни</key> <value>за съхранение на информация, което пък може да се използва, като временно действие при различни алгоритми</value></pair>

<pair><key>съхранение</key> <value>ползва се за запомняне на информация, която по-късно ще бъде нужна</value></pair>

<pair><key>LIFO</key> <value>за назоваване на определено правило на вкарване и изкарване на елементи от структура от данни и други обекти</value></pair>

</structure>

<structure name="процедура - приложение">

<pair><key>добавяне на елемент в стека</key> <value>добавяне на числата 2 и 3 в стек: stack.push(2); stack.push(3);</value></pair>

<pair><key>премахване на елемент от стека</key> <value>след изпълнение на stack.pop(a); a = 3, след следващо изпълнение на stack.pop(a); a = 2</value></pair>

</structure>

<structure name="категория1 - подкатегория - категория2">

<pair><key>формат</key> <value>правило</value></pair>

</structure>

<structure name="понятие - свойство - стойност">

<structure name="структура от данни">

<pair><key>полезност</key> <value>голяма</value></pair>

<pair><key>използваемост</key> <value>висока</value></pair>

</structure>

<structure name="LIFO">

<pair><key>подредба</key> <value>обратна</value></pair>

</structure>

<structure name="съхранение">

<pair><key>оценка</key> <value>безценно</value></pair>

</structure>

</structure>

<structure name="понятие1 - понятие2 - връзка">

<triad><object1>структура от данни</object1>

<object2>съхранение</object2> <relation>нужен за</relation></triad>

<triad><object1>LIFO</object1> <object2>FIFO</object2>

<relation>обратно на</relation></triad>

<triad><object1>добавяне на елемент</object1> <object2>клас

Стек</object2> <relation>метод на</relation></triad>

<triad><object1>премахване на елемент</object1> <object2>клас

Стек</object2> <relation>метод на</relation></triad>

<triad><object1>клас Стек</object1> <object2>стек</object2>

<relation>реализация на</relation></triad>

<triad><object1>компютър</object1> <object2>алгоритъм</object2>

<relation>изпълнява</relation></triad>

<triad><object1>информация</object1> <object2>структура от

данни</object2> <relation>се съхранява в</relation></triad>

</structure>

<structure name="процедура1 след процедура2">

<pair><key>премахване на елемент от стека</key> <value>добавяне на

елемент в стека</value></pair>
</structure>

<structure name="процедура1 - процедура2 - връзка">
</structure>

<structure name="понятие - оценка">
 <pair><key>структура от данни</key> <value>9</value></pair>
 <pair><key>съхранение</key> <value>10</value></pair>
 <pair><key>LIFO</key> <value>8</value></pair>
</structure>

<onlyOneConcept>LIFO</onlyOneConcept>

<structure name="процедура - оценка">
 <pair><key>добавяне на елемент в стека</key> <value>5</value></pair>
 <pair><key>премахване на елемент от стека</key> <value>5</value></pair>
</structure>

<structure name="процедура - прилагане - защо">
 <triad><procedure>добавяне на елемент в стека</procedure>
<applying>да</applying> <why>защото ми помага да се абстрахирам от реализацията
на добавянето и да се съсредоточа върху същността на програмата</why></triad>
 <triad><procedure>премахване на елемент от стека</procedure>
<applying>да</applying> <why>защото ми помага да се абстрахирам от реализацията
на добавянето и да се съсредоточа върху същността на програмата</why></triad>
</structure>

<missing_words>за приложенията на стека</missing_words>

<structure name="още думи">
 <element>приложение</element>
 <element>конкретни примери</element>
 <element>недостатъци</element>
</structure>

<howCheckWords>с компилатор</howCheckWords>

<findMoreWords>в свободната енциклопедия wikipedia.org</findMoreWords>

<findMoreProcedures>в учебник по Алгоритми и структури от
данни</findMoreProcedures>

<everydayLife>много дейности се автоматизират</everydayLife>

<isomorphism>начините на вкарване и изкарване на елементи (LIFO, FIFO) имат
изобилие от примери в обществото и природата. Самата област, като че ли няма
сходна.</isomorphism>

<anotherAge>това би повлияло на организацията им на живот</anotherAge>

<knowledgePractise>биха могли да се автоматизират различни
дейности</knowledgePractise>

<progress>много дейности ще се извършват от машини</progress>

<endTime>21. юни. 2010 - 17:34:23</endTime>

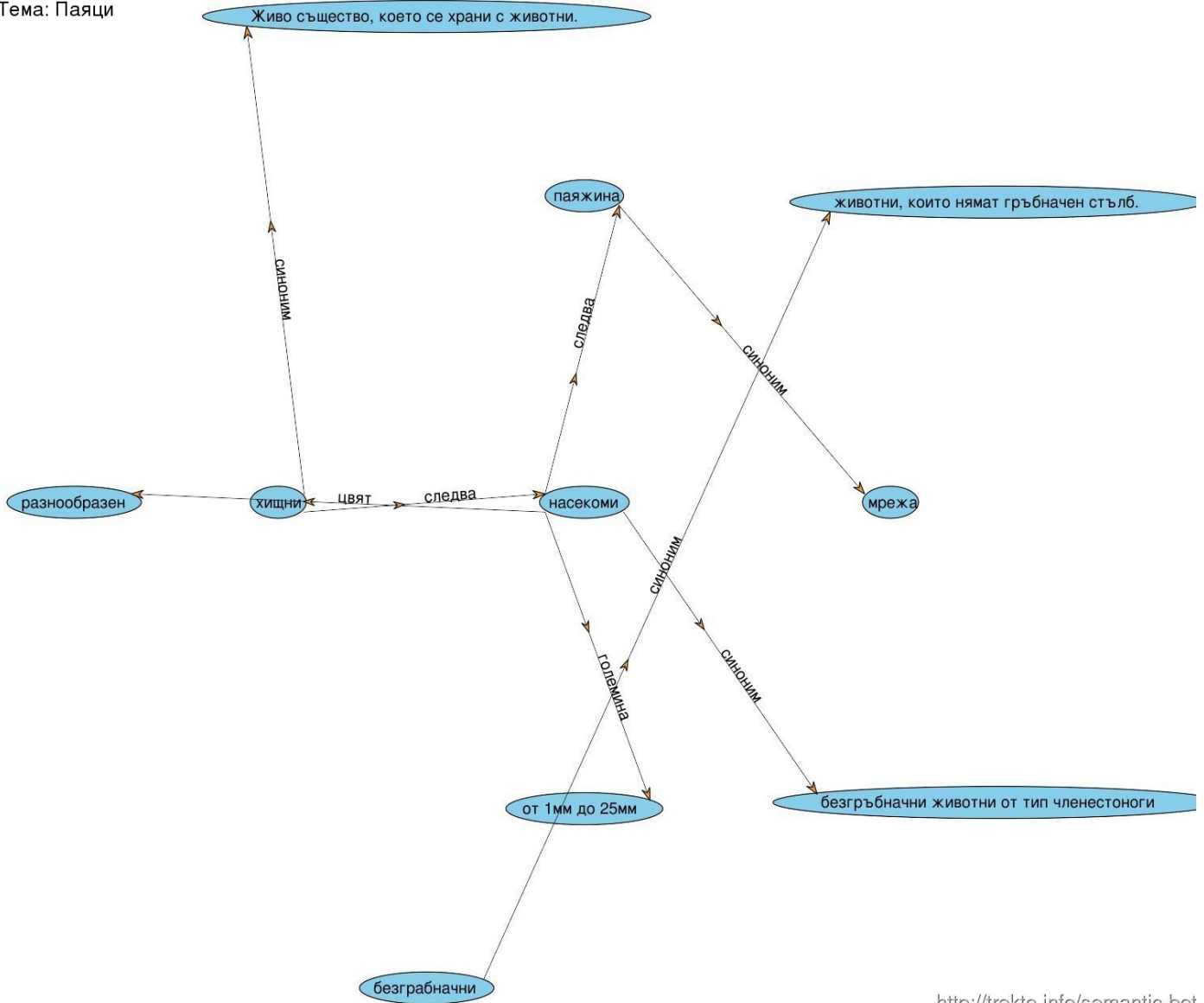
<duration>01:17:23</duration>

</semantic-map>

Приложение 2 – резултати от експерименти

Тема: Паяци

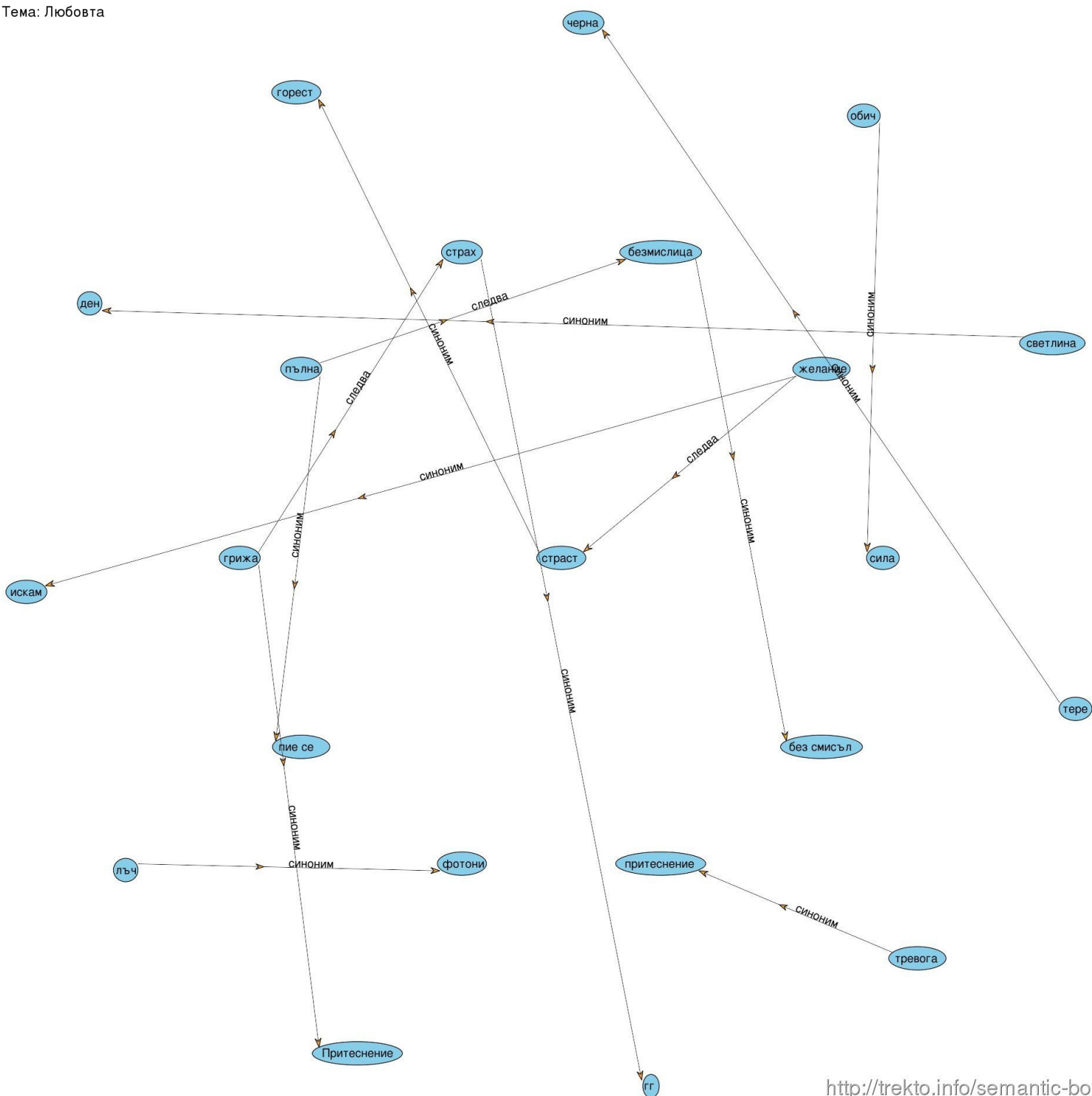
Тема: Паяци



<http://trekto.info/semantic-bot>

Тема: Любовта

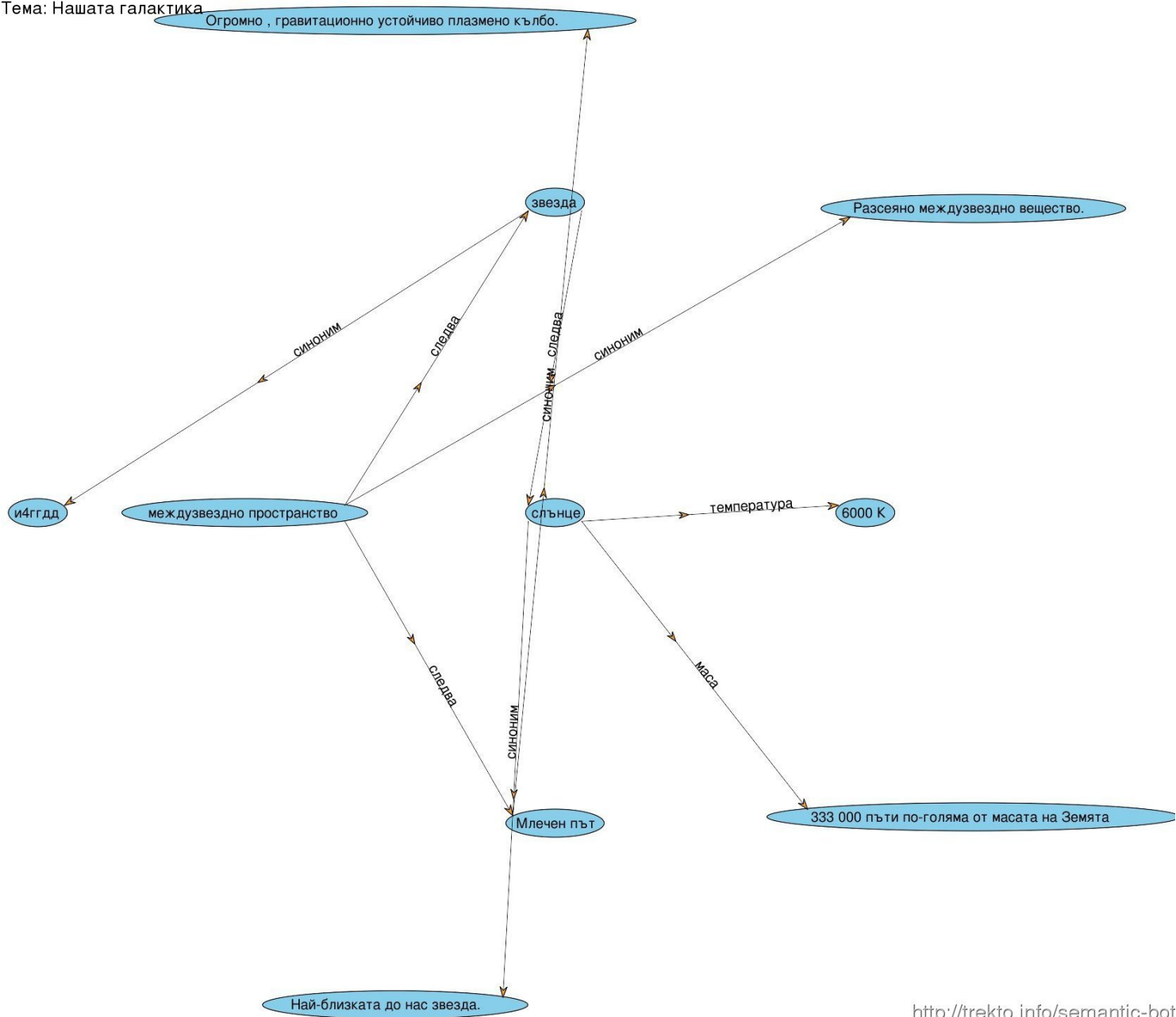
Тема: Любовта



<http://trekto.info/semantic-bot>

Тема: Нашата галактика

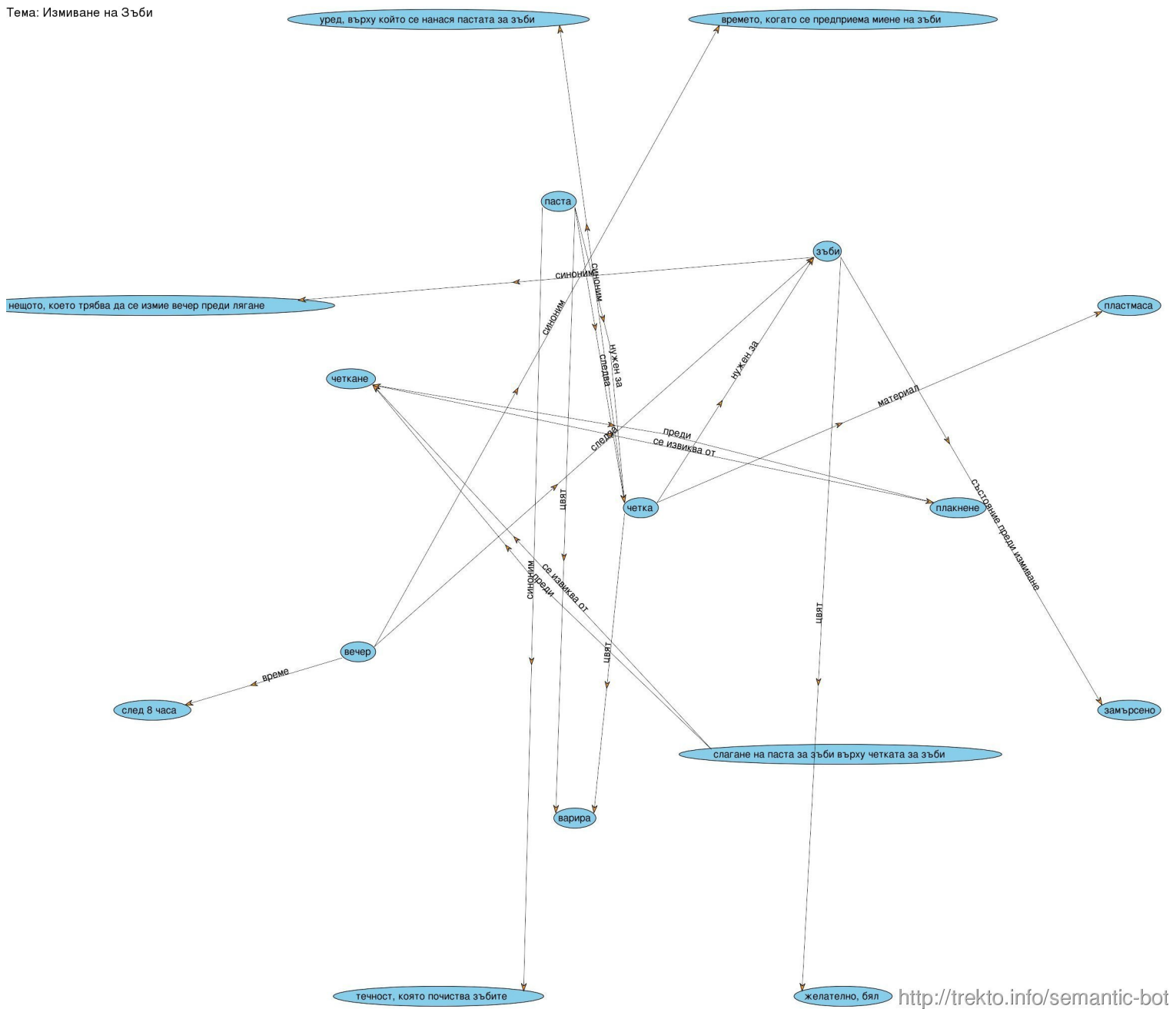
Тема: Нашата галактика



<http://trekto.info/semantic-bot>

Тема: Измиване на Зъби

Тема: Измиване на Зъби



<http://trekto.info/semantic-bot>