Developing Question Answering (QA) Systems using the Patterns

MARIA MOISE
CIPRIAN GHEORGHE
Romanian American University
1B, Expozitiei Avenue, 012101 code, Bucharest
ROMANIA
maria.moise@rau.ro 7ciprian@gmail.com
http://www.rau.ro
MARILENA ZINGALE
UN, USA
zingale@un.org

Abstract: This paper describes a way to use Natural Language Processing (NLP) techniques in order to gain faster access to information from a closed domain. Using traditional graphical user interfaces built as a tree structure, such as menus, determine users to browse irrelevant information. To avoid this, we use one of the major tasks in NLI, Question Answering (QA) and build with Visual FoxPro 9.0 a demonstrating application. We impose restrictions related to the domain of use that of tourist attractions in Romania, and restrictions related to natural language input method. We use tags to represent words or groups of words and patterns to symbolize questions. We use this representation in order to provide to user a non-traditional human-computer interaction. The QA system rely on a database where we store specific information on chosen domain and related data in order to accomplish the task of providing an answer. The answering mechanism uses a knowledge engineering with syntactic and semantic approach. At the end of this paper we discuss the QA System Evaluation.

Key-Words: Tag, Pattern, Natural Language Processing, Question Answering.

1 Introduction

In some of the most important Romanian cities you can find tourist informators (fig. 1) placed on sidewalk near mayor, train station or airport. The target users are tourists who can seek information (accommodation, things to see, historical places, monuments, zoos, museums and art galleries, maps ,...) in the area.

Fig. 1 Tourist Informator in Busteni, PH, Romania

Time spent by the tourists at these tourist informators must be minimum and maximum information must be achieved. At this moment the informators use tree structure menus organized in categories of tourist attractions. We propose an alternative to traditional GUI by using natural language, so that, the interaction between computers and humans become easier and more accessible. In this new approach the user can make use of its own native methods to access computers resources. In this paper we discuss the case of written natural language interfaces (NLI) ("use of words in order to communicate with the computer"), used as an alternative to traditional human-machine interaction.

We design this NLI as a tool which one may use to address questions to the computer in order to receive a fast, accurate and comprehensive answer. In order to successfully achieve this natural language processing task, we enforce two restrictions:

- R1: first related with the knowledge domain: application uses a closed-domain, that of touristic attractions from Romania. So, any

questions addressed to system may refer only to this domain;

- R2: the second concerning technical methods of implementing human-machine interaction. So, we do not allow users to write words freely from keyboard, but we require that they choose theirs words from a dropdown list (fig. 2).

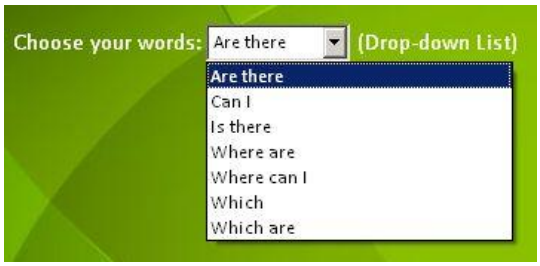

Fig 2. Second restriction: user chooses words from dropdown list

2 Problem Formulation

The fastest way to access the computer resources is when the user can make use of its own native methods. To make it possible, we use natural language interfaces (NLI) and we build a Question Answering System (QA). The problem formulation consists in two points of views: syntactic and semantic. These aspects are discussed in the following two paragraphs. We also establish (the fourth paragraph) the steps to follow in solving the task of building the QA system. These steps will be followed in building problem solution and they are highlight as paragraphs in the chapter three.

2.1 Syntactic

In this section we analyze aspects of form and structure of questions that can be processed by QA system. These issues are related to the type of sentences and the language of communication used in human-machine dialogue.

In order to identify these aspects we use an initially set of questions (a given corpus) and its translations for each language. After analyzing the corpus we concluded that for an entry from corpus (a question) is always required for the question to contain an object (OB), but optional an attribute [AT] and a localization [LO]. So, the maximum structure accepted is (OB)[AT][LO], e.g.:

(1) Are there [gothic]$_{AT}$ (churches)$_{OB}$ in [Brasov]$_{LO}$?

and the minimum (OB), e.g.:

(2) Where Can I find (a zoo)$_{OB}$?

Also valid inputs are (OB)[AT], e.g.:

(3) Where Can I [buy]$_{AT}$ (a violin)$_{OB}$?

and (OB)[LO], e.g.:

(4) Is there (a mosque)$_{OB}$ in [Constanta]$_{LO}$?

By localization we understand any area, village, city, county, region or city. For each of these we implement specific behavior when initial search fails (see 3.2).

Then for each question in corpus we use tags (markers bounded with "<" at the begin and ">" at he end) to symbolize its structure (words or its group of words). As a result we obtain a string of tags named pattern. The corresponding patterns for examples (1), (2), (3), (4) are:

(5)<are_there><atrib_arhit><locatie_cultura_oras_plural><in><oras>

(6)<where_can_i><vb_gasi><a_locatie_calatorii_oras_singular>

(7)<where_can_i><vb_a_avea><a_obiect_cultura_singular>

(8)<is_there><a_locatie_cultura_oras_singular><in><oras>

Using this procedure, we build patterns which are classes of structures. So, if we consider the question:

(9) Are there [baroque]$_{AT}$ (castles)$_{OB}$ in [Constanta]$_{LO}$?

the corresponding pattern is idem to (5).

Also, the representation with patterns is independent as structure (not as order) of the communication language used between computer and humans.

The translation of (1) in Romanian is:

(10) Exista (biserici)$_{OB}$ [gotice]$_{AT}$ in [Brasov]$_{LO}$?

and the associated pattern is:

(11)<are_there><locatie_cultura_oras_plural><atrib_arhit><in><oras>.

Special attention should be given to questions such:

(12) *Can [I rent]$_{AT}$ (a surf)$_{OB}$ in [Constanta]$_{LO}$?*

In this situation, action "*to rent*" is assimilated as an attribute of the object "*surf*". This type of question has a maximal structure (OB)[AT][LO] and its pattern representation is:

(13)<can_i><vb_a_impr><a_obiect_sport_singular><in><oras>

Another similar action is "*to buy*"and an example of a question is:

(14) *Where Can I [buy]$_{AT}$ [mountain maps]$_{OB}$?*

Structure of this question is (OB)[AT], similar with (3) and using patterns:

(15)
<where_can_i><vb_a_avea><obiect_travel_plural>

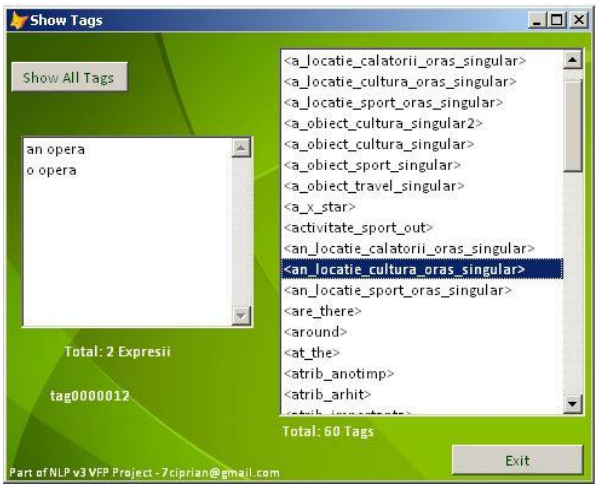

Fig. 3 Utility tool: list of all tags used in QA system

A list of all tags from QA system can be obtained using an utility from Options section. (fig. 3)
The pattern representation is mainly used to predict words when user builds a Question (see 3.1). Also, using information about question structure by pattern representation, QA system determine the appropriate search procedure (depending on (OB)[AT][LO]) and the specific way to formulate the answer using human natural language.

2.2 Semantic

The semantic point of view analyzes the meaning of the question and also the meaning of its constituent words/ group of words.
Regarding the words or group of words used in this QA system, we establish a custom framing in categories and subcategories, suitable for the application domain. A category has at least one subcategory and a subcategory belongs only to a category. The object "churches" from (1) belongs to a subcategory named "church" of category (class of objects) named "t_a" (tourist attraction). Others members (subcategories) of "t_a" category are: "volcano", "mosque", "opera", "museum", etc. Examples of other categories:

- startq (class assigned to words that can begin a question). Its subcategories are named after the words: Are there, Where Can I, Is there, …
- winter_sp (class of winter sports): ski-flight slope, ice-ring,…
- atrib_archit (class of architectural attributes): baroque, gothic,…

Query

Cod_cuv	Expresie	Lang	Cod_tag	Cod_cat	Cod_sub
cuv0000003	churches	en	tag0000003	cat0002	sub00001
cuv0000081	biserici	ro	tag0000003	cat0002	sub00001
cuv0000002	gothic	en	tag0000002	cat0003	sub00002
cuv0000082	gotice	ro	tag0000002	cat0003	sub00002
cuv0000005	Constanta	en	tag0000005	cat0004	sub00003
cuv0000006	Brasov	en	tag0000005	cat0004	sub00003
cuv0000008	Carei	en	tag0000005	cat0004	sub00003
cuv0000010	Timisoara	en	tag0000005	cat0004	sub00003
cuv0000012	Lugoj	en	tag0000005	cat0004	sub00003
cuv0000013	Curtici	en	tag0000005	cat0004	sub00003
cuv0000018	Babadag	en	tag0000005	cat0004	sub00003
cuv0000026	Bucuresti	en	tag0000005	cat0004	sub00003

Fig. 4 The categories and the subcategories are independent of the language of the QA system

The categories and the subcategories are independent of the language of the QA system. For example the word "*gothic*" and its translation into Romanian ("*gotice*") have the same category ("*atrib_archit*") and subcategory ("*gothic*") (in fig. 4 you can see the same ids: "*cat0003*" and "*sub00002*"). These two words have also the same tag: <atrib_arhit> (in fig. 4 you can see tag id: "*tag0000003*"), but the tag position in the pattern can vary depending on the language of communication in the QA system.
After analyzing the corpus we establish that we need 24 categories and 60 subcategories.
A set o general info of the words/group of words used in the QA system can be generated using a utility tool from Options section. (fig. 5)

Fig. 5 Utility tool: general info on each word used in QA system

Regarding the meaning of the question differences between ways of expression are performed using attributes. The keywords that specify the exact meaning or make the difference among sentence close meanings are marked as attributes of

requested object. We consider the following questions:
(16) *Which are (possible short trips)$_{OB}$ [around]$_{AT}$ (Bucharest)$_{LO}$?*
(17) *Which are (possible short trips)$_{OB}$ [in]$_{AT}$ [Bucharest]$_{LO}$?*
Their translations in Romanian are:
(18) *Care sunt (posibilele excursii scurte)$_{OB}$ [in jurul]$_{AT}$ [Bucurestiului]$_{LO}$?*
(19) *Care sunt (posibilele excursii scurte)$_{OB}$ [in]$_{AT}$ [Bucuresti]$_{LO}$?*
The structure of these questions is (OB)[AT][LO] and their representations with patterns are:
for (16) and (18):
(20)
<which_are><travel_oras_plural><around><oras>
For (17) and (19):
(21) <which_are><travel_oras_plural><in><oras>
Although is not a rule, this is the case when questions representation with patterns are the same in Romanian and English.
Relevance in determining the meaning of the question are words "around" (in Romanian "in jurul") and "in" (in Romanian "in") which have the same category ("detailed_localization" with id *cat0018*) but different subclases ("around" with id *sub00039* and "in" with id *sub00040*).

Query

Cod_cuv	Expresie	Lang	Cod_tag	Cod_cat	Cod_sub
cuv0000056	around	en	tag0000041	cat0018	sub00039
cuv0000167	in jurul	ro	tag0000041	cat0018	sub00039
cuv0000004	in	en	tag0000004	cat0018	sub00040
cuv0000083	in	ro	tag0000004	cat0018	sub00040

Fig. 6 Relevant words for question meaning: they have the same tag representation (tag id), the same category (category tag), but different subcategories (subcategories ids)

We also have to consider a way to store Romania's map in order to establish the geographic inclusion of area used in this QA system. For this we define "localization taxonomy" (fig. 7). Any tourist attraction is placed into a village or a city, which are geographically, the lowest framed level. We name this level L0 localization. Each L0 component is included into a county (L1). We consider here an intermediate level consists of neighboring counties, for any county (L2 localization level). A county is included into a region (L3) - we consider Romanian historical provinces. The root of this taxonomy is the country (L4).

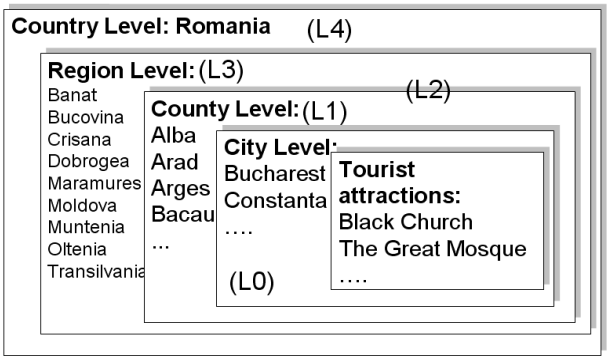

Fig. 7 "Localization taxonomy": city (the lower level), county, neighboring counties, region (Romanian historical provinces) and country (the root of taxonomy)

The L2 localization level is used when searching fails to locate the requested object in L1 (county level) and searching in neighboring counties (L2) may lead to more suitable results (lower distance to requested object) then using L3 level.

2.3 Knowledge Base

Knowledge Base (KB) should be regarded as a set of links between tourist attractions and application custom established categories and subcategories. Using a descriptive way, we can say that in KB there are statements like "Black Church is a Church" and "Black Church is Gothic" or "National Art Museum is a Museum" (fig. 8). So, if one search a museum then the QA system may returns as answer "National Art Museum". Also, if the search uses a multiple key, such as "Church" and "Gothic", then the QA system is able to answer "Black Church" (in case the above example).

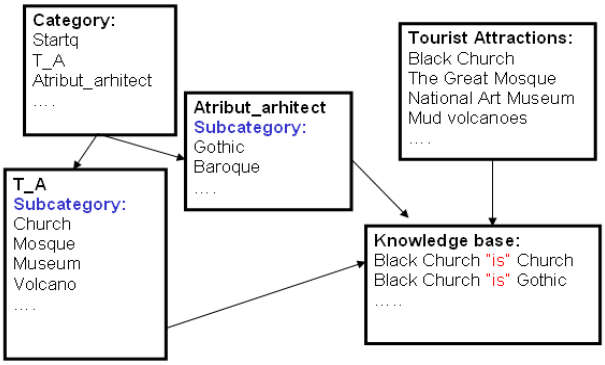

Fig. 8 Knowledge Base as a set of links between Tourist Attractions and Subcategories

The Knowledge Base contains 110 records and technical implementation is shown in fig. 9.

Contine

Cod_con	Cod_sub	Cod_ob
con000000095	sub00011	ob000000057
con000000096	sub00051	ob000000065
con000000097	sub00052	ob000000064
con000000098	sub00052	ob000000065
con000000099	sub00053	ob000000066
con000000100	sub00053	ob000000067
con000000101	sub00051	ob000000066
con000000102	sub00051	ob000000067
con000000103	sub00054	ob000000067
con000000104	sub00009	ob000000068
con000000105	sub00015	ob000000069
con000000106	sub00005	ob000000069
con000000107	sub00056	ob000000070
con000000108	sub00051	ob000000070
con000000109	sub00012	ob000000071
con000000110	sub00016	ob000000069

Fig. 9 Table *Contine*: the Knowledge Base technical implementation

In fig. 9 tourist attractions are marked by their ids (for example: ob000000057-Brukenthal National Museum) and Subcategories also by ids (for example: sub00011 - museum). This table contains no other columns. Table *Contine* is formed as an associative table between tables Object and Subcategory and implements a relation of cardinality many-to-many.

2.4 Steps in solving the task

We consider the next three steps in solving the task of building a QA system:

- step 1: question analysis (transform natural language questions into queries appropriate to be processed in the next step);
- step 2: acquisitions of items (search in the database relevant items to the request of the user, based on data received from the previous step);
- step 3: building response: with items from the previous step build the answer using human natural language.

3 Problem Solution

We use 2.1 specifications in the process of building the question and 2.2 section for the process of answering. The process of answering respects steps from 2.3 section. In order to accomplish this task, we use a knowledge engineering approach based on a deep analysis of the input. For the input question we use an elaborated analysis, based on syntactic and semantic processing.

3.1 General description

We build the QA system with Visual FoxPro 9.0 (VFP 9.0) and we store information into a database inside of 16 tables and interact with users through 25 forms.

Fig. 10 The first screen of QA system: contains user tips and map of Romania

The process of searching and retrieving information is modeled through queries to the database. To store temporarily results we use cursors, arrays and tables. When started, the QA System shows the screen from fig.10 (which is, in fact, a VFP form), and for better user orientation contains a map of Romania. The user is directed, using text labels, to start the process of build a question by pressing buttons. At this first screen, and also available at any moment in run-time, the QA system shows three more buttons: two of them to change de current QA system language (labeled "*RO*" and "*EN*"), and the third ("*Options*") to show and hide a right panel with utilities for updating existing tables (fig. 17).

To start the process of building a question, the user press "Start" button and a combo box appears (fig.11).

Fig. 11 Building a question: user choose words from drop-down combo box (restriction 2)

The combo box contains only suitable words/ group of words so that, any selection from list may form a valid question (see 3.2). A build question is valid only if it can be processed by the QA system, or in other words, if the associated pattern is among those recognized by the application.
When the user builds a question, the QA system uses a function to predict words using the structure of stored patterns.

Fig. 12 The QA system establishes the end of construction process and the Ask button in visible

When the process of building a question is completed, the user may not access the combo list (it will hide), but instead the Ask button appears (fig. 12). Built question is visible and will maintain this state until the user initiate a new question (by pressing the Reset button) or user change the QA system language of communication.
If user press ask button, the QA system shows an answer to this question.
Note that the Reset button is always enabled, so in any moment the user can reset the action of building a question. In this situation the QA system state will return to first screen (fig. 10)

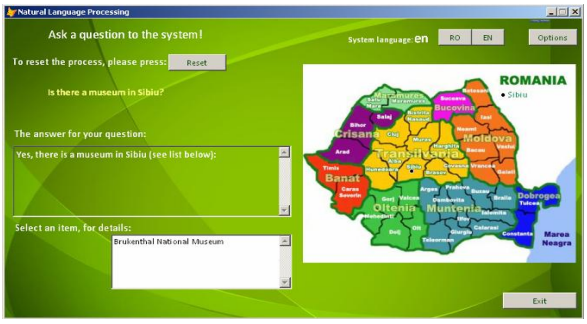

Fig. 13 After press the Ask button the QA system performes a search in the knowledge base and a list with suitable tourist attractions is displayed

For asked question ”*Is there a museum in Sibiu*”, the QA system returns the answer “*Yes, you can find a museum in Sibiu (see list bellow)*”. So, the user is guided to select a item form bellow list in order to receive more details (fig. 13) for selected tourist attraction.

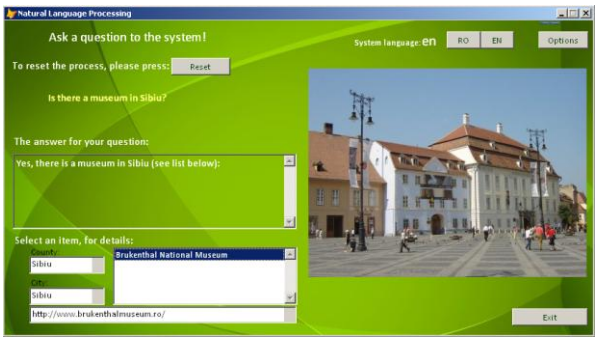

Fig. 14 Details for an entry from result list

For a selection, the user receive, as a part of the answer, a relevant photo of the tourist attraction and the possibility to access a web link which offers detailed information (fig. 14).

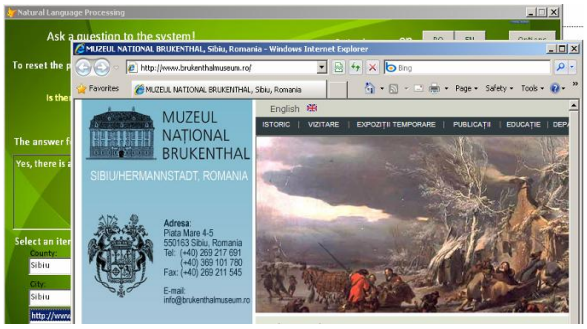

Fig. 15 With a click on provided link, the user can access a web page for more detailed information

With a click over the link, the default browser opens the web page. (fig. 15)
Information about tourist attractions is stored into the knowledge base which contains, at this point, about 100 records about objectives from the whole country. The process of building the knowledge base requires documentation work in advance and data entry procedures with respect to certain conditions. For example, the picture associated with a tourist attractions must have 400 x 300 pixels and the link must lead to a relevant website (preferably link to the official site) and not to a document (.pdf, .doc,…), a search engine list or a personal webpage. When using the provided web link, the Internet connection is required.

Fig. 16 QA system "first screen" when changing user interface language to Romanian.

After each question answered, the user can start build another by press the Reset button, from the top of form. The result is that QA system regains the state from the first screen (fig. 10). If user change QA system language, then regardless of the state of application (build question, read answer), it resets at first screen, but with labels in new selected language. For example, after reading answer about Brukenthal National Museum, if the user press RO button, then the QA system is reset as in the fig. 16. Also, all other labels on this form are displaying text in Romanian.

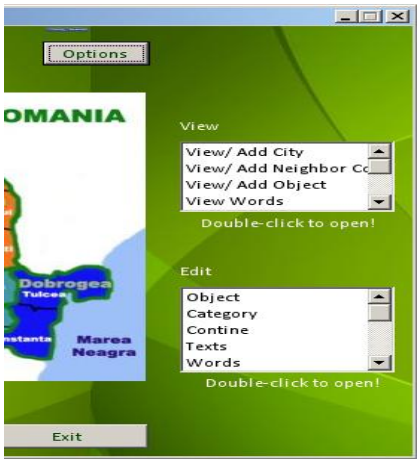

Fig. 17 The main form can be extended using Option button in order to access utilities for tables from database

The button "Options" (fig. 17) hides a portion of the graphical user interface which contains utilities to view and to manage records in any table from database. Lists of available utilities are grouped in combo boxes.

Any of the windows launched from this section are modals, so no actions are allowed to the main form unless the option window is closed.. By using these utilities, the records from tables may be updated or deleted. Also is possible to add new records to tables. After closing a utility window, if the user returns to main screen to build another question or to change the QA system current language, then the interface resets (options are hidden again) to first screen (fig. 10).

We use two types of forms: a) standard forms for browsing tables, customized from defaults forms provided by VFP 9 (fig. 16) and b) forms build to satisfy user needs to control and have an overview on database (fig. 3, fig. 5, fig. 17).

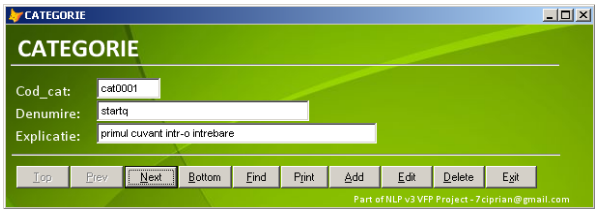

Fig. 18 Standard form, comes from a VFP template which was customized with QA system appearance

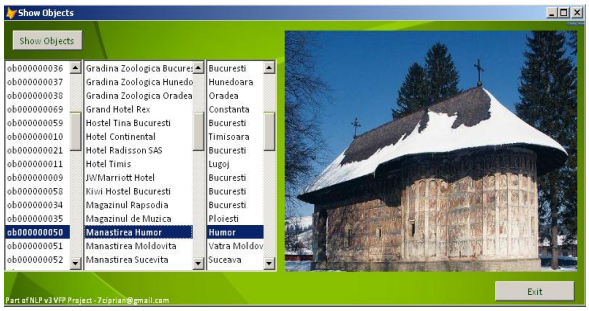

Fig. 19 Form build to browse all objects (TA) and their localization from QA system. Uses tables Object and Oras (L0 localization, see below)

3.2 Question Analysis

As a consequence of the second restriction, the first step is accomplished when a question is completed. All the words which may be used in this application, for building questions, are stored in the table "*Cuvant"* (Romanian for word). Each of them has a tag and belongs to a category or subcategory. In order to be used in the QA System, any word, regardless of communication language, necessary must be defined in the table "*Cuvant*". The representation of words "gothic" and "churches" in the table "*Cuvant*" is presented in fig. 20.

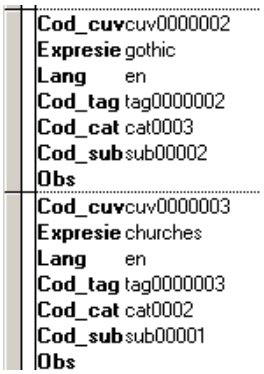

Fig. 20 Representation of words "gothic" and "churches" in table *Cuvant* (the Romanian for Word)

The process of building questions is based on patterns. A question first-word appears into selection list only if it belongs to "startq" category. For a selection from the available options in the list, the system uses a process of predict the next available inputs (fig. 21).

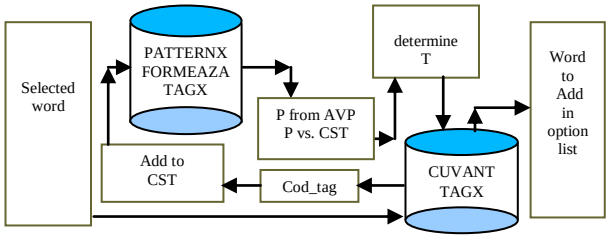

Fig.21 Building selection list (predicting words). Tables involved: *Cuvant, Formeaza, Tagx, Patterns*

So, for all selection (from the last "startq"), the system builds the corresponding string of tags (CST) and compare it (from the first tag) to the available patterns (AVP). For any inclusion (CST in P, where P from AVP) determine the first tag (T) from P which is not in CST and append de list of available option for next selection with those words which has as tag T.

If tag T is null, the end of any possible question building has been reached, the option list is hidden and the Ask button is enabled. At this point the user has built a valid question (VQ) which may be processed by QA system. During this process the QA system uses variables to store information about categories and subcategories for each selection.

To store information on patterns, tags and words we use tables (as components of a database). To avoid redundant records we use a table to store information and parameters for pattern (primary key *cod_patt*, name, number of tags, position of city, object and attribute, id of starting words when answering this type of question) and another table to store tags (primary key *cod_tag*). Between those tables we built associative tables (as the relation is many-to-many) named *"Formeaza"* and by linking tag ID (*cod_tag)* with pattern ID (*cod_patt*) we define patterns (fig. 22).

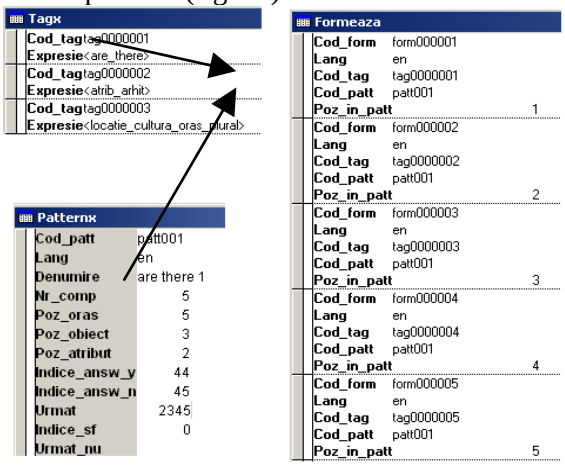

Fig. 22 Sample from tables *Tagx* (tags definition), Patterns (patterns definition) and *Formeaza* (define members (tags) of patterns)

When predicting words the QA system uses a representation of questions with id-tags. For example, table *Formeaza* represents a combination of ids (codes) (of tags - *tag0000000*and of patterns – *patt000*). Each of these ids are unique to the corresponding table. If we consider the questions (1), (2), (3), (4), then the representation with id-tags are as follows:

(22)<tag0000001><tag0000002><tag0000003><tag0000004><tag0000005>

(23)<tag0000024><tag0000029><tag0000030>

(24)<tag0000024><tag0000027><tag0000028>

(25)<tag0000009><tag0000010><tag0000004><tag0000005>

The translation of the question (1) with pattern (5) stored as in fig.22 (table *Formeaza)* into another language (e.g. Romanian) implies another pattern (10) and another five records in table "*Formeaza*" with "lang" set to value "ro". We mention that even if strings (5) and (10) were identical (no word switching) it also was needed to register the new pattern in table *Formeaza*. The id-tags representation for (5) is

(26)<tag0000001><tag0000003><tag0000002><tag0000004><tag0000005>

which is similar to (22)

3.3 Item acquisitions

The Ask button is active only when a valid question (VQ) has been built. The search procedure starts when pressing the Ask button. It finds (F0 - initial search) objects with an attribute (if any) from a location (if specified) and pass the results to the next step (see 2.4). If (F0) successful then we name this situation most favorable case (fig. 13, fig. 25). The situation of a worst case (fig. 26) happens when (F0) search is unsuccessful. If we deal with (L0) specified localization, then in this situation, QA system is designed to search as follows (F behavior):

F1: first search (OB)[AT] in county level (in all objects that belongs to the same county as [LO])

F2: if not found, then search (OB)[AT] in neighboring counties (in all objects that belongs to the neighboring counties of the county which contains [LO])

F3: if not found, then search (OB)[AT] in region level (in all objects that belongs to the counties that are in the same region as the county which contains [LO])

F4: if not found, then search in country level (in all objects that belongs to any county from Romania).

As a consequence of how we implement the QA system, any question (registered in the system) has at least one positive response, in the sense that, there is at least one object that satisfy that question. So, at least after F4 a result is passed to the next step (see 2.4).

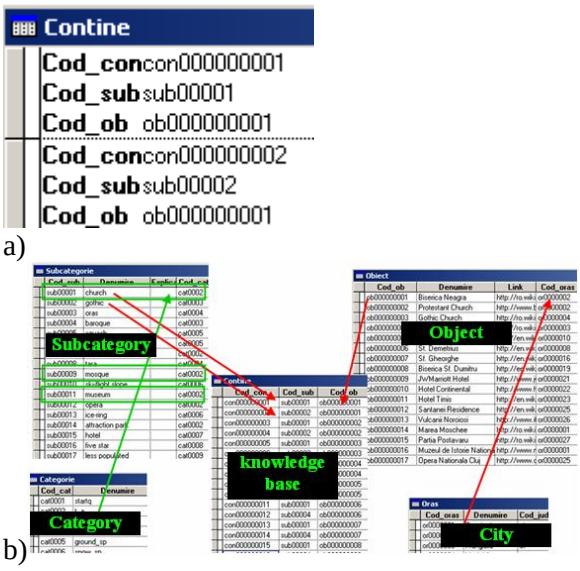

Fig 23 (a, b) The mechanism (knowledge engineering) used in QA System to establish that "Black Church" from Brasov (which is a gothic church) (ob000000001) is a church (sub00001) with gothic style (sub000022)

We define the (county/ region/ country) level as all the objects (tourist attraction) which are located geographically there.

If there is no localization specified in VQ, then the search procedure browse the country level (F0 is successfully). If in VQ the localization level is L1 or L2 and F0 is unsuccessful, then the searches (F behavior) start with F2 or F3. Because the QA system is designed for tourists (which may travel) we prefer to search both object and attribute in other location and not to search only object or only attribute. In order to accomplish the acquisition job, the database contains the following important tables: *Subcategorie* (subcategory table), *Obiect* (objects table: "Black Church", "Meriott Hotel", "Mud Volcanos"), *Oras* (table containing all the localizations of database objects). Every object has the id of the area/village/ city it belongs.

A subcategory is the granular description of a word. We use an associative table, named *Contine*, between table *Subcategorie* and table *Obiect* in order to declare in the QA system the properties of an object. If the Id from Object table for "Black Church" is "ob000000001", then in the *Contine* table two records are written (con000000001) and (con000000002).

The F behavior (in case of initial search failure), is based on an implementation with tables, of an hierarchy regarding Romania territorial inclusion. (a way to store Romania's map) So, we define tables for each level of localization: *Oras* (which is Romanian word for city, for L0), *Judet* (Romanian word for county - L1), *Regiune* (Romanian for region - L2), *Tara* (Romanian for country - L3). The top of this structure is represented by table *Tara* (which contains only one record: Romania). Table *Regiune* has nine records corresponding to the Romanian historical regions, represented with different colors in fig. 24. Table *Judet* contains 42 records, corresponding to the number of counties. They are outlined in fig. 24, for all regions.

We record neighboring counties into a new table, named "*Judete_vecine*", which has about 200 symmetrical records. We kept the symmetrical form of construction from reason of ease of use. When fill out this table we build a utility tool (fig. 24. a) in which, for any input A is neighbor with B we took care in updating also the symmetrical relationship: B is neighbor with A.

At this moment the hierarchy regarding Romania territorial division is recorded into QA system, at all levels.

a)

Judete_vecine

	Cod_i_vec	Cod_jud	Cod_vecin
	192	mm	sj
	193	sj	mm
	194	mm	cj
	195	cj	mm
	196	mm	bn
	197	bn	mm
	198	mm	sv
	199	sv	mm
	200	sv	bn
	201	bn	sv
	202	sv	ms

b)

Fig. 24 a) Utility tool: Registration and visualization of neighboring counties; b) table "*Judete_vecine*" where we record neighboring counties

3.4 Building answer

When building an answer, we must consider factors like word order in sentence (which may be different for any language) or issues regarding words concordance. The QA system builds a response using words referred by indices, selected from VQ or stored in table *Conversatie.* In fig. 22 fields "*Indice_answ_y*" and "*Urmat*" set the pattern to establish the affirmative answer (AA) for any VQ which is represented in QA system by pattern "patt001". The AA starts with the word referred by index "*Indice_answ_y*" and continues with words corresponding to tag position from VQ pattern as mentioned in string "*Urmat*".

When a negative answer (NA) is build the QA system typically uses "*Indice_answ_n"* and "*Urmat_nu*" fields. If value of "*Urmat_nu*" is null then QA system uses "Urmat" value. We mark an answer as NA when it is a result of F behavior. By difference, the answer obtained with F0 is an AA. After pressing the Ask button, the user receive a message M under the label "*The answer for your question"* (fig. 25), a list of matching objects, and each object found is detailed by a photo, an internet link and localization (County and City).

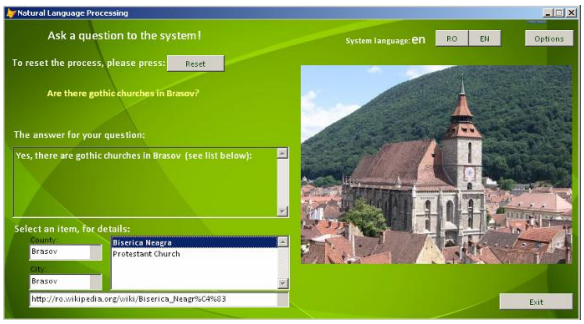

Fig. 25 The answer to the question (most favorable case)

In fig. 25 we illustrate the most favorable case (when item is found where asked). The message for user is an AA answer "*Yes, there are gothic churches in Brasov* (see list below):" But, if not found at city level, the value for the message for user is classified as NA and has de following form "*No, there are no gothic churches* [in Carei]$_{F0}$, or in other [neighboring cities]$_{F1}$, or in [neighboring counties]$_{F2}$ (…), or in [region Muntenia]$_{F3}$, but found [in country]$_{F4}$ (Romania)!" (fig. 26).

In fig. 26 the message M marks all the failed steps (F behavior) with answer in F4. The list of objects shows all the records (country level) for (OB) ="Church" and [AT]="gothic" available in database.

Fig. 26 The answer to the question (worst case)

When building an answer the QA system uses indices which are not required for all records. The table for indices (*Conversatie)* has a primary key (*cod_conv*) and columns to specify language (*lang*), expression (*expresie*) and index. A translation of an expression has the same index number. When changing the language, QA system building response procedures work in the same manner, only changing the language of the displayed text (using the corresponding index and QA system language). Also, all the texts from NLI are contained in table *Conversatie*, so when a user changes QA system language, the messages are

shown in the new language. This feature is also based on indexes and current QA system language. The extension to other languages (as far as it concerned the NLI) becomes easy to implement by adding to the table *Conversatie* the corresponding translation, with respect of indexes.

3.5 QA System Evaluation

Regarding the evaluation of the QA system we can state the following:

a) The **success rate** is 100%.We define success rate, as a percentage of the number of questions for which the application provides an answer of all those addressed in a session. As a consequence of the application programming, the QA system provide an answer for any question which could be built with words from dropdown lists. For any tourist attraction, selectable from dropdown lists, exists at least one record in knowledge base.

b) **Speed** of the QA system is good. We use Select statements over a database to search items specified in user request. Also, the process of building the answer (in natural language) uses select statements in order to identify appropriate words to be assembled in the form of a sentence.

c) **Availability and reliability** are based on very good characteristics of Visual FoxPro 9.0 DBMS regarding the stability of its programmed applications under Windows environments.

d) **Completeness** of answers achieves a good level especially because of the F behavior in case of initial search failure. Application does not offer short/ positive/ negative answers like: "Yes!", "No!" or "You cannot find a museum.". The QA system shows a complete response in natural language ("Yes, you can find a gothic church in Brasov") and furthermore related information on found objects (localization, photo, link to a web page). If initial search fails (search for gothic church in Bucharest), the user receives a list with similar objects ("gothic church") located in neighborhoods (Brasov)

e) **Accuracy** of the answer is guaranteed in the most favorable cases (when QA system has information on requested object). In other cases (when execute F behavior) the accuracy is related only to the requested object and not to the requested location.

f) **Relevance** of the answer is suitable to an application whose target users are tourists. If, for requested location, the search with multiple key (object with attribute) fails, then the QA system will not omit the attribute or the object to make a partial search (after one key only). Because tourists are willing to travel, the QA system provides answers for the multiple key formulated although this can determine that they cross a certain distance.

g) In terms of **utility**, the QA system can have a good implementation on Tourist Informators. Although the current system uses a traditional GUI, transition to a NLI can be easily done, both applications using queries to a database. As system requirements, the QA system may run on machines with medium resources in terms of computing power. To increase the utility factor the Tourist Informators can be equipped with touch screens so that the user may benefit also from hardware innovative technology (and not only from software new trends)

4 Conclusions

As the volume of information is growing the more difficult is to find what interests us. When we deal with large amounts of information is becoming increasingly difficult to organize and to provide data quick access. A new method, more intuitive and closer to human nature is NLI. This method provides rapid information by one-step access to the all information stored in database. The benefit of this situation belongs entirely to the user, which spends less time when using it. QA systems makes the use of a computer simplest and does not require special knowledge. More, if the requests from the question cannot be satisfied, then with F behavior QA system offers results from neighborhood (geographical) areas.

With NLI the programmer uses more time to develop an application then working with classing GUI. Mechanisms or restrictions must be defined in order to to simulate a computer human behavior. Implementing human ways of communication are more complicated than implementing a menu interface. When building a meniu is enough to establishes categories and subcategories and populate them with components. If building natural language interface, then we must add the effort to establish the connection between this components and human words/ group of words. Human communication involves nuances and features that can be identified at NLI level by syntactic and semantic analysis. Not all aspects of human communication can be successfully captured. Therefore successful NLI impose restrictions related to area of applicability and processed forms/ size of communication.

Further work may implies extension to other languages and migration to web technology

References:

[1] Moise M., *Sisteme Informatice cu Baze de Date*, ProUniversitaria Publishing House, 2008.
[2] Popescu I., *Modelarea Bazelor de Date*, Tehnica Publishing House, 2001.
[3] MacCartney B., Natural language inference - Ph.D. dissertation, Stanford University, June 2009.
[4] McNamee P., Snow R., Schone P., Mayfield J., Learning Named Entity Hyponyms for Question Answering, Proceedings of IJCNLP 2008.
[5] Collins M.C., Interactive Visualizations of natural language, University of Toronto, 2010.
[6] Winograd T, *Language as a Cognitive Process*, Addison-Wesley Publishing Company, 2004.
[7] Jackson P., Moulinier I., Natural Language Processing for Online Applications - Volume 5, John Benjamins Publishing Company, Amsterdam, 2002.
[8] Jurafsky D., Martin J.H., - Speech and Language Processing, Prentice Hall, 2000.
[9] Gavriel M., Capturing Negation in Question Answering Systems, Master dissertation, University of Edinburgh, 2005.
[10] Hristea F., "Introducere in procesarea limbajului natural cu aplicatii in Prolog", University of Bucuresti Publishing Company, 2000.
[11] Diekema A.R.,Yilmazel O.,Liddy E.D.,Evaluation of Restricted Domain Question-Answering Systems, Syracuse University, 2004.
[12] *** http://www.profox.ro/