Natural Language Processing for Rapid Information

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 in the domain of touristy attractions in Romania.

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

1 Introduction
In some of the most important Romanian cities you can find touristic informers as a kiosk placed on sidewalk near mayor, train station or airport. The target users are tourists who can seek specific information about attractions in the area. In this case, the time spent by a user to achieve an information is critical and determine, with a low value, the success of the search. With natural language, the interaction between computers and humans become easier and more accessible. The user can make use of its own methods to access computers resources. We discuss in this paper the case of written natural language (use of words in order to communicate with the computer), which we use to build question so that the computer may provide an answer (Question Answering Systems). In order to successfully achieve this NLP task, we enforce two restrictions: one related with the words membership (which are parts of a closed-domain: touristic attractions in Romania) and the second requiring the user to choose words from a list and not to write them freely from keyboard. As a building method we use a deep approach based on syntactic and semantic processing.

2 Problem Formulation
We analyze QA systems syntactic and semantic issues in order to identify problem solution. The problem formulation is of two points of views such as: syntactic and semantic.

a. Syntactic
When building QA systems we first need a corpus (an initially set of questions). We use one corpus (translated) for each language. For an entry from corpus (a question) is always required to contain an object (OB), but optional an attribute [AT] and an 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. (see 2.b) For each of these we implement specific behavior when 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 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><in><oras>.
(7)<where_can_i><vb_a_avea><a_obiect_cultura_singular><in><oras>.

Using this procedure, we build classes of equivalent patterns. So, if in (1) we replace Brasov with other city name (eg. Constanta), then the pattern representation would be the same (5). Although for a translation of (1) in other language, the order of tags in pattern (5) may be different, the same answering procedures are used. E.g. the pattern for Romanian translation of (5) is:
(9)<are_there><locatie_cultura_oras_plural><atrib_arhit><in><oras>.
Using pattern representation and other related stored properties (see 3.3 and fig. 13), QA system determine the appropriate search procedure (depending on (OB)(AT)(LO) structure) and the specific way to formulate the answer using human natural language.

b. Semantic
Then we establish a custom hierarchy, suitable for chosen domain and system capabilities. The object “churches” from (1) belongs to a subcategory named “church” of category (class of objects) named “t_a” (tourist attraction). Others members of “t_a” category are: “volcano”, “mosque”, “opera”, “museum”, etc. Examples of other categories:
- startq (first word into a question): Are there, Where Can I, Is there,
- winter_sp (winter sports): ski-flight slope, ice-ring
- attrib_archit (architectural attribute): baroque, gothic.
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. Any tourist attraction has localization such as a village or a city (we name them L0 localization) which are included into a county (L1). A county is included into a region (L2) and all the regions form the country (L3). We consider the historical regions. Then we consider three steps in solving the QA task:
- question analysis (transform natural language questions into queries appropriate for the next step);
- acquisitions of items (search in the database relevant items to the issues raised by the user, based on data received from the previous step);
- building response (with items from the previous step build the answer using human natural language).

3 Problem Solution
We use 2.a specification when implement the NLI in the process of building the question and 2.2 section for the process of answering. We build the QA system with Visual FoxPro 9.0 (VFP 9.0) and we store information into a database inside the 16 tables and interact with users through 25 forms. The process of searching and retrieving information is modeled through queries to database. To store temporarily results we use cursors, arrays and tables. When started, the QA System shows the screen from fig.1 (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.

Fig. 1 The first screen of QA system
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, and the third to show and hide a right panel with utilities for updating existing tables (fig. 2).

Fig. 2 Building a question and the user choose words from drop-down combo box
When the process of building a question is completed, the user may not access the combo list, but instead the Ask button appears (fig. 3).

Fig. 3 The QA system establishes the end of construction process and the Ask button in visible
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 (fig. 4).
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.1). 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, the user press RO button, then the QA system is reseated as the image of the fig.7. Also, all other labels on this form are displaying text in Romanian.

The Option button contains, as records in combo boxes, utilities to view and to manage records in any table from database. The windows launched from this section are modals, so no actions are allowed to the main form unless the option window is closed. The main form can be extended using Option button (fig. 8) in order to access utilities for tables from database. Using these utilities the records from tables may be updated or deleted. Also is possible to add new records to tables.

We use standard forms for browsing tables, customized from defaults forms provided by VFP 9 and also, forms build to satisfy user needs to control and have an overview on database (fig. 9, fig. 10).
3.1 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 build questions, are stored in table “cuvant” (Romanian for word). Each of them has a tag and belongs to a category or subcategory. It also important to define one word in all languages, otherwise it can not be used. The representation of words “gothic” and “churches” in table Cuvant is presented in fig. 11.

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. 12).

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 the end of any possible question building has been reached, the option list is hidden and the Ask button is enabled. In 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 an 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 atribut, 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. 13).

The translation of the question (1) with pattern (5) stored as in fig.13 (table Formeaza) into another language (e.g. Romanian) implies another pattern (9) and another five...
records in table “Formeaza” with “lang” using value “ro”. We mention that even if strings (5) and (9) are identical (no word switching) it also needs to register the new pattern in table Formeaza.

3.2 Item acquisitions

The search procedure starts when pressing the Ask button. The Ask button is active only when a valid question (VQ) has been built. It finds (F0) objects with an attribute (if any) from a location (if specified) and pass the results to the next step. If (F0) successful then we name this situation most favorable case (fig. 5, fig. 16). The situation of a worst case (fig. 17) happens when (F0) search is unsuccessful, and we deal with (L0) specified localization. In this situation, QA system is designed to search as follows (F behavior):

F1: first search (OB)[AT] in county level
F2: if not found, search (OB)[AT] in neighboring counties
F3: if not found search (OB)[AT] in region level
F4: if not found search in country level
F5: if found pass result to the next step (3.3 Building response).

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

If 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 acquirement job, the database contains the following important tables: Subcategorie (subcategory table), Obiect (objects table: “Black Church”, “Merriott 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).

Also, for realize F behavior in case of search failure, we implement with tables, an hierarchy regarding Romania territorial division. 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.15. Table Judet contains 42 records, corresponding to the number of counties. They are outlined in fig.15, for all regions. We record neighboring counties into a new table, named “Judete_venice”, 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.15) in which, for any input A is neighbor with B we took care in updating also the symmetrical relationship: B is neighbor with A.

![Fig. 15 Utility tool: record neighboring counties](image)

At this moment the hierarchy regarding Romania territorial division is recorded at all levels except level L0

3.3 Building response

When building a response, we must consider factors like word order in sentence (which may be different for any language) or issues regarding words concordance. QA system builds a response using words referred by indices, selected from VQ or stored in table Conversatie. In fig. 13 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 labeled “The answer for your question” (fig. 16), a list of matching objects, and each
object found is detailed by a photo, an internet link and localization (County and City).

Fig. 16 The answer to the question (most favorable case)
In fig. 16 above we illustrate the most favorable case (when item is found where asked). The value for M is an AA answer “Yes, there are gothic churches in Brasov (see list below):” But, if not found at city level, the value for M is classified as NA and has de following form “No, there are no gothic churches [in Carei]0, or in other [neighboring cities]1, or in [neighboring counties]2 (…), or in [region Muntenia]3, but found [in country]4 (Romania)” (fig. 17).

Fig. 17 The answer to the question (worst case)
In fig. 17 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. When building an answer the QA system uses indices which are not required for all records. This table 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.

4 Conclusion
This method provides rapid information by one-step access to the all information stored in database. With NLI the programmer uses more time to develop an application then working with classing GUI. The benefit of this situation belong entirely to the user, which spend less time when use 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. Further work may implies extension to other languages and migration to web technology

References: