Documentation and analysis of an endangered language: aspects of the grammar of Griko

Database and Website manual

Antonis Anastasopoulos — NTUA
Marika Lekakou — UOI

December 12, 2013

Contents

Introduction

This manual includes all necessary information regarding the implementation of the database and of the website for the project "Documentation and analysis of an endangered language: aspects of the grammar of Griko".

The flow of the manual follows the flow of the project. First the preprocessing of the transcribed data is presented. The relational database is described in the next section and the website user interface in the final chapter.

Data Preprocessing

The programme used for the transcription of the data is Praat. Using Praat, the information for each audio segment (which corresponds to a sentence) is stored in tiers.

Following the transcription protocol, the tiers used were:

- transcription
- tagging
- gloss
- lemma
- metadata

The metadata tier included information on the speaker, only in the cases of the locations where there were multiple speakers in the interviews.

The preprocessing of the data included two steps:

1. Parse the .TextGrid files into a more suitable format, such as plain text format. This was done with the `ParseTextGrid.py` python script, which is available in griko.project.uoi.gr/pythoscripts/ParseTextGrid.py.
2. Check the tags for possible inconsistencies in the tags, according to the tagging protocol. This was done with the `CheckData.py` python script, which is available in griko.project.uoi.gr/pythoscripts/CheckData.py.

Database

The processed data are then stored in a relational SQL database, using the InnoDB database engine for SQL. The database is automatically filled from a python script, which reads the preprocessed data and stores them into the appropriate tables.

The character set of the database is set to '`utf8_general_ci`' in order to accommodate for all the characters present in the Griko, Italian and Greek alphabet.

SQL Tables

The main table of the database is table '`sentences`'. It stores the whole trascription, tagging, gloss, lemma and question id. In addition, it stores information on the location and whether the segment is retrieved from other sources. It also provides the name of the .wav file that corresponds to this segment, if it is available.

The table '`questions`' stores the questionnaire that was used during the interviews, including the test sentence, its Italian translation, and a list of the syntactic phenomena that this question tries to examine. The table '`location`' stores a list of the locations of the interviews and the table '`keyword`' stores a list of all the syntactic variables/keywords, which were examined with the various questions.

Moreover, the table '`questionkeywords`' provides the M-to-N (multiple in both ways) match of each question to each syntactic keyword. This is needed in order to efficiently retrieve the question ids and the sentences when performing queries based on the syntactic keywords.

In addition, the tables '`tokens`', '`itgloss`', '`lemma`' and '`tags`' store each individual transcription word, Italian gloss word, lemma and tag for all sentences, also storing (incrementally) its position in the sentence, as they would result from any *split()* function on the sentences. Although it may seem redundant, these tables enable a faster individual search on their contents and also ensure the matching of the transcription tokens to their relevant tag, gloss and lemma, through the variable of the `position`.

Finally, the rest of the tables include the information of the tags. For each part of speech, there exists its corresponding table, which stores the information of its particular features. This is needed because the different part-of-speech tags employ different features (for more information, see 'Tagging protocol' -link!!).

Tags' features

The features for each tag are always stored as an integer (or boolean) value, in order to ensure coherency, avoid issues with string comparisons and also increase the speed of the queries' execution.

The features are stored in the following way:

- If the feature has only two values, then it can be represented with a boolean variable ("0" or "1"). Example of such feature is the feature `Italian` which denotes whether the word is Italian or not.
- If the feature can have multiple values, then each value is matched with an integer (incrementally, starting from "0"). The matching of these values to the integers is presented in the relevant tables in the appendix.

Note: It is important to note that the feature "case" -for nouns, adjectives, pronouns, determiners- is referred as `casse` *in the database, because the term* `CASE` *is a bound word, as it is part of the SQL syntax.*

Website

The website is created by simple html pages, using javascript and php for specific functionalities.

Javascript is used for creating the adjusting forms for querying the database. This allows for the form to be user friendly and also ensure that no meaningless queries are executed. For example, for each selected tag to search by, only the corresponding features are shown.

Javascript is also used for the functionalities of the images. The lightbox functionality uses the `lightbox-2.6.min.js` package.

PHP is used for the connecting, querying the database and displaying the query results.

Database Search UI

The database server uses the `mysqli` php extention (its main advantage being that it supports Unicode character encoding), so the interface for querying the database also uses `php` and `mysqli`.

Search Form

The form is organised into tabs, enabling queries with the various parameters.

The tabs *Word, Lemma, Gloss (Italian)* receive as input from the user a term and search in the appropriate tables for it. Note that in the current version the input term has to be spelled consistently with the orthographic conventions adopted for Griko. As a result, a Griko term also has to include the appropriate accent diacritic (eg. "tròo" instead of "troo"), otherwise it will not return any results. This means that an Italian (or other Unicode) keyboard is required. The next version will hopefully enable search without the accent.

The *Test Sentence* and *Keyword* tabs receive input only from the checkboxes and return the corresponding sentences. The form is constructed by retrieving the test sentences and the keywords from the `questions` and `keyword` tables of the database. The form in the *Location* tab is constructed in the same way, only in this case the input is requested in the form of a drop-down list.

Special attention is given in the *Tag* tab, which enables search by PoS tags. Since each part of speech has different features, with which the user must be able to search the database, the form is constructed dynamically.

The user first selects the PoS tag and the needed features are then presented for selection. This is accomplished with the *Pane()* and *init()* javascript functions.

The function *Pane()* determines the properties of each "pane" (every form is defined as a different pane) and implements the functionality that allows for forms to only appear when they are needed. The syntax of the function can be interpreted as:

$$Pane(X, Y, Z) \equiv Show\ pane\ Y,\ if\ X\ takes\ value\ Z.$$

The function *Init()* defines the various dependencies of the form-panes on the user input, initializing the page.

The values for each option of the features correspond to the values that are stored in the database and can be found in the appendix.

The current version does not support queries with more than one PoS tags, but a new version with this feature enabled is already planned.

Queries

The queries in general are the result of SQL `JOIN` queries between the tables `sentences, location` and a 3^{rd} table, which depends on user input.

The third table is decided based on the search tab that the user has selected (for example, if the user is querying through the *Lemma* tab, then the third table is the `lemma` table) and, in the case of PoS tags, the type of PoS (eg, if the selected PoS tag is *Verb*, then the third table is the `verbtags` table).

The rest of the input of the user is used for constructing the conditions of the `JOIN`. The `searchtags.php` script constructs and executes the query, by iterating over the possible user inputs.

It is important to note that input for different features is used in an additive way to construct the conditional query. For example, when querying the database for all *Adjective* tags and the *Masculine* and *Singular* options are selected (for the *Gender* and *Number* features), then the result will contain all adjectives that are both masculine AND in singular number.

However, the additional input over the same feature is used in a different way, constructing the conditional query using "OR" for the conditions of this feature. For example, when querying the database for all *Adjective* tags and the *Masculine* and *Feminine* options are selected (for the *Gender* feature), then the result will contain all adjectives that are either masculine OR feminine.

Of course, the above can also be combined, in order to construct more complicated queries.

If no option is selected for some feature, then this feature does not form part of the query. As a result, if no option is selected in general, the result consists of all the sentences in the database.

Results

The results of the query are presented in a table, which is also constructed by the `searchtags.php` script. As a default, the question id, transcription, location and audio file (if available) of the sentence are included in the results.

According to the options selected by the user, also italian gloss, tags and lemmas may be shown.

Where audio data are available, the appropriate image is shown, also opening, when clicked, a new browser tab or window (depending on user preferences) with a simple player for the wav file. The audio files are accessed through a simple interface, which automatically selects the player that each browser supports, in order to avoid compatibility issues.

Conclusion

For any enquiries, suggestions or more information on the implementation and the technical details of the database, the website or the whole project, please contact:

- Antonis Anastasopoulos (for technical information or for help on using the search engine) at anastasopoulos.ant@gmail.com
- Marika Lekakou (for enquiries about the project or the corpus) at mlekakou@cc.uoi.gr.

Appendix

The following tables present the 1-1 match of the features to the values used in the database and the online form.

Locations	
1	Calimera
2	Corigliano
3	Martano
4	Sternatia
5	other

Gender	
0	Masculine
1	Feminine
2	Neutral
3	Unknown

Number	
0	Singular
1	Plural
2	Unknown

Case	
0	Nominative
1	Genitive
2	Accusative
3	Vocative
4	Unknown
5	Undefined

Degree	
0	Positive
1	Comparative
2	Superlative

Position	
0	Preposed
1	Postposed

Verb Finiteness	
0	Non Finite
1	Finite

Verb Type	
0	Main
1	Auxiliary

Auxiliary Verb Type	
1	Modal
2	Perfect
3	Passive
4	Aspectual

Voice	
0	Active
1	Non Active

Tense	
1	Past
2	Non Past

Aspect	
1	Perfective
2	Imperfective

Mood	
1	Indicative
2	Imperative
3	Subjunctive

Person	
0	First
1	Second
2	Third

Non Finite Verb Subtype	
0	Not applicable (finite verb)
1	Infinitive
2	Participle

Adverb Type	
0	Temporal
1	Locative
2	Interrogative
3	Aspectual
4	Epistemic
5	Quantitative
6	Quantitative (Negative)
7	Other
8	Manner

Subordinating Complementizer Subtype	
1	Temporal
2	Default
3	Declarative
4	Interrogative
5	Relative
6	Conditional
7	Subjunctive
8	Causal

Pronoun Type	
0	Personal
3	Demonstrative
5	Interrogative
6	Possessive
7	Quantificational

Pronoun Strength	
0	Weak
1	Strong
2	Non applicable

Complementizer Type	
0	Subordinating
1	Coordinating

Determiner Type	
0	Definite
1	Indefinite

Particle Type	
0	Negative
1	Other

Particle SubType	
0	Indicative
1	Non Indicative
2	Sentential
3	Unknown

The rest of the features:

- Proper (nouns)
- Nominalised (adjectives)
- Fused (adpositions)
- Enclisis (pronouns)
- Participation in Clitic Doubling (pronouns)
- Italian (all words)

are modelled with boolean values. Their default value is 0 (false) and if the specification applies to the particular word/tag, then the value is 1 (true).