DOI: 10.17469/O2104AISV000017

PAOLO BRAVI

Prosit: a Praat plug-in for the search and inspection of corpora of annotated audio files

The paper presents the Prosit software, a plug-in for Praat (Boersma, Weenink, 1992-2018), one of the most renowned programs for carrying out phonetic research. The Prosit plug-in is designed to help researchers in (i) making a flexible search on corpora of sound files with Praat TextGrid annotations; (ii) building their own corpus by editing and managing sound files and TextGrids; (iii) listening, visualizing, inspecting and analyzing sounds that match the search criteria. The rationale behind the project is the observed use of different schemes in sound annotation carried out via Praat, an instrument whose potential benefits allow searches to be carried out on sound corpora built up with different approaches. The Prosit plug-in also makes it possible to apply a number of either batch or single operations on the search outcomes.

Keywords: Praat plug-in, Praat search engine, TextGrid, sound annotations, sound visualization.

1. *Introduction and motivation*

The Praat TextGrid is a widely diffused format for storing information related to audio files. In many cases audio file analysis needs previous manual or automatic annotation of the sound files, so that the corpus on which the analysis is carried out eventually comes to consist of a number of pairs of sounds and TextGrids.

That said, a typical analytical batch procedure consists of iterating some kind of procedure on all the files comprising the corpus, which are usually stored in one directory and have a consistent annotation. But what happens if the "corpus" – or a pre-version of it – is not (yet) built up using a consistent annotation scheme or if we simply want to explore what could be large and non-homogeneous groups of annotated sound files as a possible preliminary step towards the creation of a well-organized corpus?[1]

Prosit[2] is a Praat plug-in designed to face this kind of need: on the one hand, it is a search-engine (with no external dependencies) that allows step-wise flexible research within possibly large and non-homogeneous groups of TextGrids,

[1] "Standards are like toothbrushes, everyone agrees that they're a good idea but nobody wants to use anyone else's" is a brilliant statement attributed to Murtha Baca that well synthesizes the difficulties relevant to metadata definition and management (see Pomerantz, 2015: 65).

[2] The name is an acronym for the *Pr*aat *O*bject *S*earch and *I*nspection *T*ool. At the same time, not without a hint of irony, it expresses the author's wish to allow users to carry out a fruitful search and successful work with TextGrids and audio files.

while, on the other, it allows the user to either listen to, edit, visualize or analyze the audio files that match the TextGrid interval(s) requested and save the relevant output.[3] The lack of a proper internal database management system and a corresponding search engine has been noted in numerous cases.[4] This is linked both to the fact that Praat allows its users the freedom to develop their own system of file management by means of its built-in scripting language (BIM: scripting [1][5]), and also because complex research on documents and relevant metadata and annotations can be managed properly using a software specifically designed for database management and information retrieval.[6]

In many cases, the workflow of researchers studying phonetics involves the use of multiple software. A quite common procedure among researchers is to carry out a first step of sound analysis and annotation using Praat, and then to apply statistics and drawing relevant graphics with R (RDCT, 2011). Some specifically designed software has been developed so as to automatize and improve interoperability between the two programs (Albin, 2014; Boril, Skarnitzl, 2016). Other programs designed for media file annotation which provide utilities for corpus management and analysis are presented and described in Durand, Gut & Kristoffersen, (2014, part III).[7] The decision to avoid any dependency in the Prosit plug-in clearly has its costs in terms of efficiency and entails a number of limitations with regard to database systems explicitly designed for

[3] To date, while pending comprehensive and exhaustive testing and the writing of documentation with a detailed description of the available functions, the Prosit plug-in can be requested directly from the author at the address paolobravi.gm@gmail.com.

[4] The Praat command "Create Corpus..." described in Boersma, (2014) is conceived in view of providing important functions for corpus management and to overcome a previously observed limitation of the program regarding the fact that "Praat does not include a proper database system as such, so searching a speech corpus with Praat must be implemented through Praat scripts (which can become painfully slow)" (Lennes, 2005: 15). The scope and functions of the Prosit plug-in here described only partially overlap with those of the Praat in-built command.

[5] The Praat built-in manual (BIM) will be constantly cited throughout the paper. For convenience, the following method will be adopted to point the reader to the appropriate manual page (note: the Praat version used is 6.0.39). In the Praat *Objects window*, under the *Help* section, use the command *Search Praat manual* and then write the word given (in this case, the word "scripting") in the form field. Then, from the list of outputted links, choose the entry indicated by the number within square brackets (in this case, the entry is [1]). Hence, from now on, these references to the Praat manual will have the following format: BIM: query [number].

[6] Generally speaking, it has to be observed that DMS software based on relational databases, wherein metadata are stored in a principled way, is far more flexible (though harder to design and maintain) than what are known as flat file databases, which contain less strictly organized and more redundant information with respect to the former (Srivastava, 2014; Elmasri, Navathe, 2016).

[7] A comparison between functions provided in Prosit and those implemented in other software designed for similar aims is beyond the scope of this paper. However, the interested reader should be aware that among the programs that can read/import annotations in the TextGrid format and that provide functions specifically designed for corpus analysis, a basic list of the most notable ones comprehends at least ELAN (Wittenburg, Brugman, Russel, Klassmann & Sloetjes, 2006), EMU-SDMS (Winkelmann, Harrington & Jänsch, 2017), EXMERaLDA – in particular the EXAKT tool – (Schmidt, 2002).

information storage and searches, but it does allow a Praat user to drive – so to say – 'his/her' own car, managing data that are structured according to the TextGrid format (BIM: TextGrid [2]), with which he/she is most likely to be well acquainted.

2. *User control and interfaces*

Prosit is based entirely on the tools (types of objects, commands, etc.) and interfaces (editors, forms, windows) provided in Praat. Excluding external dependencies has its pros and cons: the most notable advantages are ease of installation and use,[8] since any Praat user with basic practice can use the program by utilizing its standard apparatus. Moreover, a Praat user with some knowledge and experience with Praat scripting can change or add parts to the plug-in according to his/her own needs.[9] In terms of disadvantages, Prosit, being based on a linear search mechanism, has not the efficiency necessary for the inspection of large corpora. Moreover, there are some practical limitations related to the characteristics of the interfaces and editors which, as part of the Praat GUIs, cannot be overcome as of yet.

User control of the parameters involved in any operation provided by Prosit is carried out on the typical Praat forms. In many cases they appear in sequence, based on the user's choices, and are implemented by means of the "beginPause /endPause" mechanism for user control (BIM: user [1]). Instead, search report and relevant Prosit commands are managed via "ManPages" that are dynamically created whenever a new search starts and at every step of the search (BIM: manpages [1]).

3. *Search: the metaphor of a shopping trip*

Searching intervals in Prosit come about in three phases. For ease of conceptualization, these phases are identified using the metaphor (and the relevant names) of a shopping trip. In phase 1, the user sets what is called the *district*, viz a directory (as a "simple" or "parent" one), and a *store type*, viz a file type capable of storing information relevant to an audio file. At the moment, the only searchable file types are Praat TextGrids, but searches within other file types can be envisaged in a future version of the plug-in. In phase 2, the user sets the *shelf* (or the shelves) that Prosit has to explore. When the store type is "TextGrid",

[8] The plug-in can be installed following the simple instructions contained in the relevant page of the Praat built-in manual (BIM: plug-ins [1]).

[9] Praat has a built-in introduction to the program (Help > Praat intro) and a detailed explanation of its scripting language (Help > Scripting language). Furthermore, several tutorials on the use of the Praat program are available, written in various languages and with a variety of reader proficiency and topic focus in mind. Among those written in English, a certainly non-exhaustive list comprises van Lieshout, 2017; Styler, 2017; Weenink, 2018; Wood, 1994-2018; Cangemi, Auris, submitted.

this means identifying the tier(s) that will be searched through. A spectrum of possibility is provided to establish which tiers are to be searched in. In phase 3, the user eventually sets the *item*(s) he/she is interested in.

Search results, together with a description of the research carried out, is made available to the user by means of Praat "ManPages" that allow the user to perform a number of different operations. These range from listening to the specified part of the sound, visualizing it through specific animation devices (also playing it at a speed lower than the original) and editing and manipulating it. It is also possible to perform a number of batch operations on all items.

Searches can be carried out in different steps. Users can add new items via a new search, refine the preceding search, retain or exclude selected items through successive steps. Every search step and relevant parameters and results are easily available at every moment, so that the overall search can also take different paths according to the researcher's needs.

3.1 Search phases

The paragraphs in this section describe each of the three search phases.

3.1.1 Phase 1: "District"

After the plug-in installation, a button named "Prosit" will appear in the Praat Object window under the "Praat" menu. Clicking on this button gets the search underway. The first form that appears regards the *district* where a defined *store type* is searched for – at the moment, as mentioned above, the only manageable information "stores" are of the TextGrid type. Two main options are available in this search phase. The first one allows the user to specify one of the three ways of stating the directory to be searched for: [a] choosing by browsing; [b] choosing among the last directories searched[10]; or [c] stating the directory address explicitly, either writing it manually on a form or choosing from a pre-set list of directories that the user can easily create or modify in advance.[11]

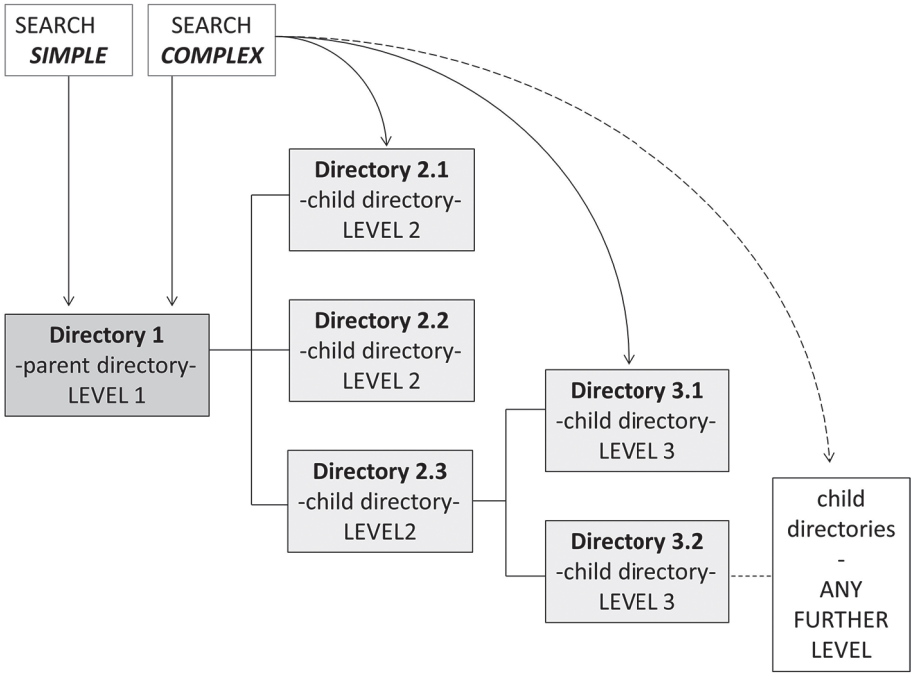
The second parameter allows the user to state the 'complexity' of the search, i.e. the number of levels of 'child directories' to be searched for beyond the main ('parent') directory. By default, this parameter is set to "complex" (i.e. the search will involve not only the directory chosen, but the 'child' directories contained in it) and "unlimited" (i.e. the search will proceed until no further 'child' directory is found). Users are allowed to either choose a "simple" search, i.e. a search limited to the directory stated, or to specify, when a "complex" search is chosen, a number corresponding to the levels of 'child' directories to be searched for (see Figure 1). This

[10] The number of directory addresses held in the memory is set to ten by default. This number may be modified according to the user's preferences.

[11] The list of pre-set directory addresses is stored in the file plugin_PROSIT/lists/Others/SearchDirOM_Input.txt. Each user will set his/her list of preferred addresses according to his/her own needs.

flexibility allows the search to be restricted or widened according to specific needs and to manage every possible different organization of file storage.

Figure 1 – *Sketch of the first phase of the search:two basic options ("Simple" and "Complex") are available and, in the latter case,a definite number of "child" directory levels can be set by the user*



3.1.2 Phase 2: "Shelf"

The second phase of the search is aimed at deciding which *shelves*, i.e. what type of annotations, are to be searched for. Dealing with 'stores' of annotations like TextGrids, 'shelves' are – outside the shopping metaphor – tier names. There are three possible ways of searching through them. The first method is to look up TextGrid tiers whose name matches a given string according to a specific chosen string-matching criterion.[12] The second method is to choose a tier name from a list comprising all tier names available in the TextGrids found in the first phase of the search. The third method envisages carrying out a search in all the tiers of the TextGrids. It goes without saying that the search may take longer with this third option, and that it could possibly yield a vast number of non-consistent results. In all cases the user can decide about the case-sensitiveness of the string search.

[12] The following series of options are available: "is equal to", "is not equal to", "contains", "does not contain", "starts with", "does not start with", "ends with", "does not end with", "matches (regex)". The last option, of course, permits much wider search flexibility, but requires a knowledge of how Praat manages regular expressions (see BIM: regular [1]).

If what is looked for is to carry out a search through two or more, but not all tiers, two options are available. The first way is to use the "match (regex)" option, which allows the search to be made in a single step. The second way is to carry the analysis through different steps, using the ADD method. For example, if one needs to find all non-empty intervals in tiers whose names contain either the text "Profilo" or the text "Strut", the user can either: (i) perform the search in one step, using the "match (regex)" option and writing "Profilo|Strut" in the search text field; or (ii) perform the search in two steps, using the "contains" option and writing "Profilo" in the search text field, and then repeating the option, after selecting the ADD method, searching for the text "Strut".

3.1.3 Phase 3: "Items"

The third phase of the search is aimed at deciding which *item*, i.e., beyond the metaphor, what label is to be searched for. Two out of the three methods described in paragraph 3.1.2 for searching through tier names are also available for labels. This means that it is possible (i) to set a string which matches the annotations according to one of the established matching patterns (see note 12) by writing it manually in the form, or (ii) to choose the string to be searched for from a predefined list. A third method is available for searching through labels, that is (iii) choosing a label from a list of all the ones actually present on the "shelves" – i.e. in the tiers – retrieved in phase 2.

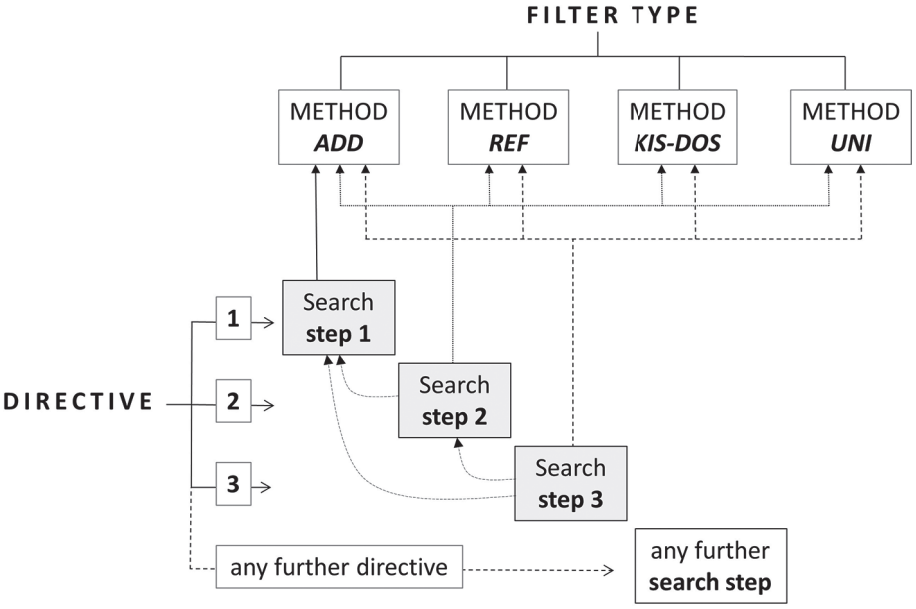
One of Prosit's crucial and potentially beneficial aspects is the possibility of choosing different ways to define the strings to be searched for in the TextGrid annotations, along with the flexibility offered by the chance to choose between various matching criteria, which also comprises the opportunity of using regular expressions. In particular, the method (iii), based on a preliminary examination of the annotation present in the selected tiers, may be useful in a preliminary phase of inspection of the corpus of TextGrids.

3.2 Search steps

Searches can be carried out in more steps, i.e. new directives may be given to modify the outcome of previous search steps. Each step (with the exception of the first one) can operate on the results of the previous steps in four modes. The first mode is addition (symbol: "ADD"). In this mode, new items are added which match the new directive criteria. This method is by rule adopted in the first search step. The second mode is refinement (symbol: "REF"). In this mode, items found in previous steps of the research are held if the outcomes also match the new search directive. The third mode regards selections (symbols: either "KIS" or "DOS"). In this mode, items selected by the user are either excluded (symbol: DOS – drop off selected items) or served, with exclusion of the non-selected one (symbol: KIS – keep in selected items). The fourth mode regards item reduplication (symbols: UNI). In this mode, two (or more) items

that refer to the same sound part outcoming from different directives are synthesized as a single item.

Figure 2 – *Prosit multiple search step method allows the user to conduct an inspection of the annotated sounds using different types of filter at each step*



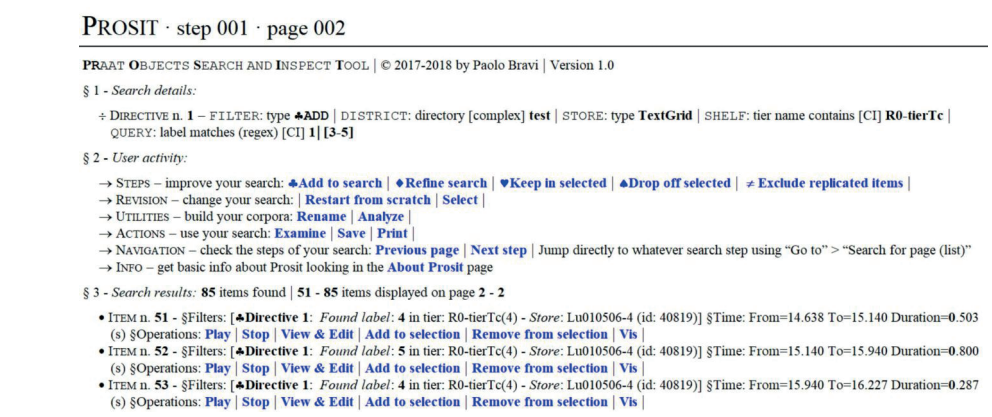
The mechanism has been conceived with the aim of allowing the user great search flexibility, making it possible to rethink and carry out multiple tries during the search process. The Manpage interface makes it easy to pass from one search step to another and to account for the procedures involved in each step.

4. *Inspection, editing, actions*

Search results are reported at every step by means of Praat ManPages which are organized in three sections (see Figure 3a-b). The first section ("Search detail") gives a summary of the parameters used in each research step. In particular, it reports the filter type used (see Paragraph 3.2), the "district" where the search was carried out, the "store" type searched for (as of yet, the fixed type is "TextGrid"), the "shelf" (as of yet, string values referring to TextGrid tier names) and the query, as expressed in the three search phases outlined above. The second section ("User activity") contains a series of subsections that allow the user to improve or change his/her search and to 'navigate' in the various steps of his/her search and may also help him/her in building a corpus and using the search for specific aims (see below, par. 4.1). The third section ("Search results") lists the outcomes of the search. Each item is accompanied by three sub-

sections. The first ("Filters") reports a short summary of the search matching. The second ("Time") gives time details. The third one ("Operations") provides a list of words (or sentences) linked to the plug-in scripts that allow the user to listen, view and edit the sound through the Praat Sound & TextGrid editor, to select or deselect the item and to operate with various types of editor providing animation for sound visualization and manipulation (see par. 4.2). From here on, these links that appear in bold blue characters on the ManPages (see figs. 3a-b) will be referred to as "WL".

Figure 3a – *Example of a two-step search - step 1: the search directive 1 looks up into a parent directory (in this case, the built-in "test" one) for TextGrids comprising tier names that contain the string "R0-tierTc" and looks in these tiers searching for labels matching the regular expression "1|[3-5]" (85 items are found)*


PROSIT · step 001 · page 002

PRAAT OBJECTS SEARCH AND INSPECT TOOL | © 2017-2018 by Paolo Bravi | Version 1.0

§ 1 - *Search details:*

÷ DIRECTIVE n. **1** – FILTER: type **♣ADD** | DISTRICT: directory [complex] **test** | STORE: type **TextGrid** | SHELF: tier name contains [CI] **R0-tierTc** | QUERY: label matches (regex) [CI] **1|[3-5]**

§ 2 - *User activity:*

→ STEPS – improve your search: **♣Add to search** | **♦Refine search** | **♥Keep in selected** | **♠Drop off selected** | **≠ Exclude replicated items** |
→ REVISION – change your search: | **Restart from scratch** | **Select** |
→ UTILITIES – build your corpora: **Rename** | **Analyze** |
→ ACTIONS – use your search: **Examine** | **Save** | **Print** |
→ NAVIGATION – check the steps of your search: **Previous page** | **Next step** | Jump directly to whatever search step using "Go to" > "Search for page (list)"
→ INFO – get basic info about Prosit looking in the **About Prosit** page

§ 3 - *Search results:* **85** items found | **51 - 85** items displayed on page **2 - 2**

• ITEM n. **51** - §Filters: [**♣Directive 1**: *Found label*: **4** in tier: R0-tierTc(4) - *Store*: Lu010506-4 (id: 40819)] §Time: From=14.638 To=15.140 Duration=**0**.503 (s) §Operations: **Play** | **Stop** | **View & Edit** | **Add to selection** | **Remove from selection** | **Vis** |
• ITEM n. **52** - §Filters: [**♣Directive 1**: *Found label*: **5** in tier: R0-tierTc(4) - *Store*: Lu010506-4 (id: 40819)] §Time: From=15.140 To=15.940 Duration=**0**.800 (s) §Operations: **Play** | **Stop** | **View & Edit** | **Add to selection** | **Remove from selection** | **Vis** |
• ITEM n. **53** - §Filters: [**♣Directive 1**: *Found label*: **4** in tier: R0-tierTc(4) - *Store*: Lu010506-4 (id: 40819)] §Time: From=15.940 To=16.227 Duration=**0**.287 (s) §Operations: **Play** | **Stop** | **View & Edit** | **Add to selection** | **Remove from selection** | **Vis** |

Figure 3b – *Example of a two-step search - step 2: the directive 2 refines the retrieving adding a sec-ond condition: labels that do not contain the string "a" or "A" are searched for in tiers whose name is equal to "R2-LIN-VV-PT" (100 items are found)*


PROSIT · step 002 · page 001

PRAAT OBJECTS SEARCH AND INSPECT TOOL | © 2017-2018 by Paolo Bravi | Version 1.0

§ 1 - *Search details:*

÷ DIRECTIVE n. **1** – FILTER: type **♣ADD** | DISTRICT: directory [complex] **test** | STORE: type **TextGrid** | SHELF: tier name contains [CI] **R0-tierTc** | QUERY: label matches (regex) [CI] **1|[3-5]**
÷ DIRECTIVE n. **2** – FILTER: type **♦REF** | REFINE: search results in Step **1** | STORE: type **TextGrid** | SHELF: tier name is equal to [CI] **R2-LIN-VV-PT** | QUERY: label does not contain [CI] **a**

§ 2 - *User activity:*

→ STEPS – improve your search: **♣Add to search** | **♦Refine search** | **♥Keep in selected** | **♠Drop off selected** | **≠ Exclude replicated items** |
→ REVISION – change your search: | **Restart from scratch** | **Select** |
→ UTILITIES – build your corpora: **Rename** | **Analyze** |
→ ACTIONS – use your search: **Examine** | **Save** | **Print** |
→ NAVIGATION – check the steps of your search: **Previous page** | **Previous step** | **Next page** | **Next step** | Jump directly to whatever search step using "Go to" > "Search for page (list)"
→ INFO – get basic info about Prosit looking in the **About Prosit** page

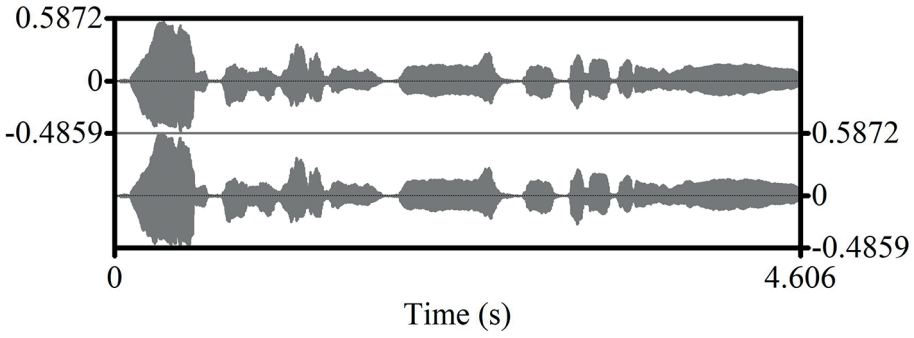
§ 3 - *Search results:* **100** items found | **1 - 50** items displayed on page **1 - 2**

• ITEM n. **1** - §Filters: [**♣Directive 1**: *Found label*: **5** in tier: R0-tierTc(4) - *Store*: Br170163-8 (id: 40816)][**♦Directive 2**: *Found label*: **es** in tier: R2-LIN-VV-PT(1) - *Store*: Br170163-8 (id: 40816)] §Time: From=0.766 To=1.319 Duration=**0**.553 (s) §Operations: **Play** | **Stop** | **View & Edit** | **Add to selection** | **Remove from selection** | **Vis** |

4.1 Batch operations

Prosit allows operations to be carried out in batch mode on the items resulting from retrieving. In the "user activity" paragraph, a number of operations are available that can be of use when dealing with (or building up) a corpus.[13] Among these, two sub-sections are available in the "Utilities" section. The first is accessible via the WL "Rename", which opens a pop-up form that allows the establishment of some parameters relevant for batch tier rename in all the files matching the search criteria.[14] The second may be reached through the WL "Analyse", which allows the resulting pitch in the part of the sound output from the search on the annotation to be extracted (and saved). In the "Actions" section, the user can "Examine" in a synthetic way the output of his/her search: by virtue of the Praat function "Concatenate recoverably" (see BIM: concatenate [2]), the user can listen to all the sounds in sequence, aligned in a single file with an accompanying TextGrid where each label indicates the origin of each portion of sound present in the chain. Other batch operations are accessible under the WL "Save", particularly as far as TextGrid and Sound file are concerned. Either TextGrid or Sound parts relevant to the retrieved intervals can be saved, with different options relevant to amplitude peaks and audio format. The WL "Print" offers three possibilities: waveform, pitch contour and pitch histogram of retrieved items can be printed, with a number of parameters relevant to each type of graph (see figs. 4a-c).

Figure 4a - *Batch operation on outcomes from search: printing of waveform excerpt*



[13] In this paper, only a brief summary of the operations that may be carried out on the retrieved items is provided. A detailed description of these functions will be given in a future contribution.

[14] This may be of help when TextGrid files are not stored in just one directory or when they have different tier names since they are part of different collections of files, whereby tiers are annotated according to non-homogeneous rules and/or in the light of different aims. Obviously, the user is asked to decide whether to overwrite the original TextGrid files or to save the new ones in a separate place.

Figure 4b - *Batch operation on outcomes from search: printing of a pitch contour excerpt*

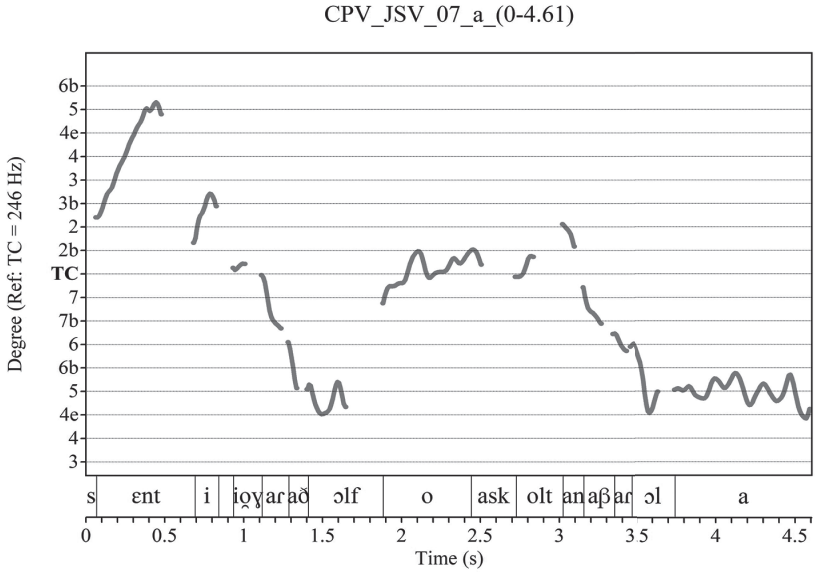
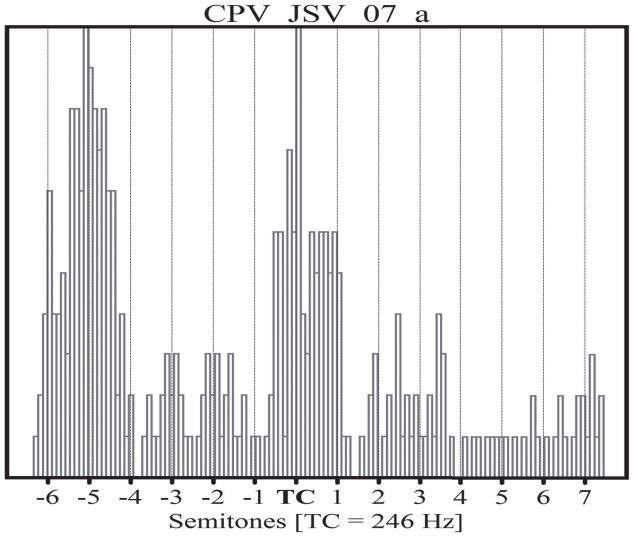


Figure 4c - *Batch operation on outcomes from search: printing of a histogram of pitch values (bins = 10 cents) found in the contour excerpt*

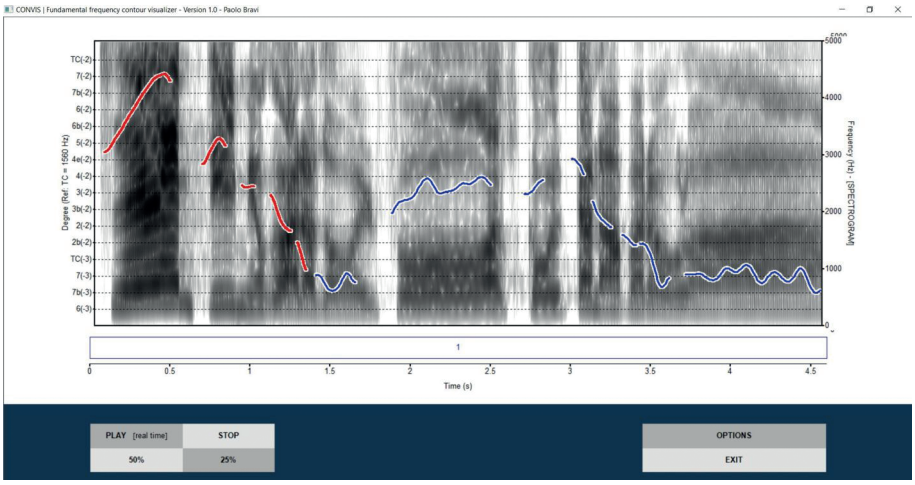


4.2 Single item operations

Items resulting from user retrieving are described in terms of their match to the search directive(s) and their time features (see above, Paragraph 4 and Figures 3a-b). A series of operation may be carried out on each item. Firstly, each sound segment can be quickly listened to via a pair of WLs named "Play" and "Stop". Secondly, the "View & Edit" WL allows access to the Praat TextGrid Editor with the time

boundaries relevant to the item, thereby accessing all commands available in this editor. This WL makes it possible to visualize just the tier searched for or all the tiers actually present in the TextGrid, and also to observe in which context the retrieved annotation is placed. Thirdly, another pair of WLs ("Add to selection" and "Remove from selection") allow the list of selected objects to be managed in the light of further search steps or batch operations. Fourthly, the WL "Vis" allows the user to visualize and/or manipulate the sound relevant to each item in either of the following two ways: either via (i) the "ConVis" panel or (ii) the Praat Manipulation Editor. The (i) ConVis panel is an application designed within Prosit on the Praat Demo Window with the objective of visualizing, through an animation, the evolution of the pitch contour (either at normal speed or delayed up to 4 times the original length) on top of the spectrogram and above a TextGrid tier relevant to the item indicated by the user (see Figures 5).

Figure 5 – *Single item operation: dynamic visualization of the pitch contour excerpt*



'Visualization' via (ii) the Praat Manipulation Editor is actually much more than a means to visualize sounds. In fact, this kind of editor permits each sound to be manipulated through pitch stylization and/or modification and duration changes (detailed information on the use and functionalities of this editor are in BIM, manipulation [1]).

5. *Conclusions*

The plug-in Prosit is at its first steps. This is obvious to the potential user who will find some "one-option menus" that, at first sight, do not seem to have any meaning. In most cases, this single-choice (i.e. pseudo-) menu is part of the lists of commands that are under elaboration or which have yet to be properly tested. However, even at this early stage, Prosit contains a basic infrastructure that allows future develop-

ments along lines that are the same (or analogous) to the ones that have already been set and are in use.

Future developments may regard different parts of the plug-in. In particular, three aspects could be improved or developed to better suit prospective user needs. The first one is to widen the range of possibilities of the search engine, which might look in "stores" other than TextGrids, in particular allowing searches on different metadata sets conceived according to standard procedures.[15] The second one is to enlarge the number of available batch operations, either aimed at helping in the constitution of a specific corpus or at managing and analyzing data and sounds resulting from the search. The third one is to provide other ways of fostering inspection capacity, of editing single items and of analyzing and visualizing the acoustical features of the relevant sound files.

Acknowledgements

I wish to thank Ignazio Macchiarella, Francesco Capuzzi and Francesco Cangemi for their support and advice.

Bibliography

Albin, A. (2014). PraatR: An Architecture for Controlling the Phonetics Software "Praat" with the R Programming Language. In *Journal of the Acoustical Society of America*, 135(4), 2198.

Boersma, P. (2014). The Use of Praat in Corpus Research. In Durand, J., Gut, U. & G. Kristoffersen (Eds.), *The Oxford Handbook of Corpus Phonology*. Oxford: Oxford University Press, 342-360.

Boersma, P., Weenink, D. (1992-2018). Praat: Doing Phonetics by Computer. http://www.fon.hum.uva.nl/praat / Accessed 15.04.18.

Boril, T., Skarnitzl, R. (2016). Tools rPraat and mPraat. In Sojka, P., Horák, A., Kopeček, I. & Pala, K. (Eds.), *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, Brno, Czech Republic, 12-16 September 2016, 367-374.

Cangemi, F., Auris, B. (submitted). *Praat Handbook*. Berlin: Language Science Press.

Durand, J., Gut, U. & Kristoffersen, G. (Eds.) (2014). *The Oxford Handbook of Corpus Phonology*. Oxford: Oxford University Press.

DCMI Usage Board (2012). DCMI Metadata Terms. http://dublincore.org/documents/dcmi-terms / Accessed 15.04.18.

[15] Neither a unique standard or framework, nor a general agreement exists on how to describe the contents and characteristics of digital audio files. Based on the current diffusion of metadata sets and on the main area of research activity of the Prosit developer, the possibility of searching for Dublin Core (DCMI Usage Board, 2012) and BDI (ICCD, 2007) metadata can be envisaged in a version- to- come of the plug-in. For a general survey on the conception and use of metadata in linguistic corpora, see Durand, Gut & Kristoffersen, 2014 (part I).

Elmasri, R., Navathe, S.B. (2016). *Fundamentals of Database Systems* (7th edition). Boston: Pearson.

ICCD (2007). Scheda BDI. Beni demoetnoantropologici immateriali versione 3.01 - Struttura dei dati e normativa di compilazione unificata e integrata - gennaio 2007. http://www.iccd.beniculturali.it/getFile.php?id=284 / Accessed 15.04.18.

Lennes, M. (2005). Hands-on Tutorial: Using Praat for Analysing a Speech Corpus. http://www.helsinki.fi/~lennes/vispp/lennes_palmse05.pdf / Accessed 15.04.18.

Pomerantz, J. (2015). *Metadata.* Cambridge MA – London: The MIT Press.

RDCT (2011). R: A Language and Environment for Statistical Computing. http://www.R-project.org / Accessed 15.04.18.

Schmidt, T. (2002). EXMARaLDA – ein System zur Diskurstranskription auf dem Computer. In *Arbeiten zur Mehrsprachigkeit*, B(34), 1-23.

Srivastava, R. (2014). *Relational Database Management System.* New Delhi: New Age International Private Limited.

Styler, W. (2017). Using Praat for Linguistic Research. http://savethevowels.org/praat / Accessed 15.04.18.

van Lieshout, P. (2017). PRAAT Short Tutorial – An introduction. https://www.researchgate.net/publication/270819326_PRAAT_--_Short_Tutorial_--_An_introduction / Accessed 15.04.18.

Weenink, D. (2018). Speech Signal Processing with Praat. http://www.fon.hum.uva.nl/david/sspbook/sspbook.pdf / Accessed 15.04.18.

Winkelmann, R., Harrington, J. & Jänsch, K. (2017). EMU-SDMS: Advanced Speech Database Management and Analysis in R. In *Computer Speech & Language*, 45, 392-410.

Wittenburg, P., Brugman, H., Russel, A., Klassmann, A. & Sloetjes, H. (2006). ELAN: a Professional Framework for Multimodality Research. In *Proceedings of LREC 2006, Fifth International Conference on Language Resources and Evaluation*, Genoa, IT, 22-28 May 2006, 1556-1559

Wood, S. (1994-2018). *Introduction to Praat.* https://swphonetics.com/praat/introduction / Accessed 15.04.18.