Praaline: An Open-Source System for Managing, Annotating, Visualising and Analysing Speech Corpora

George Christodoulides
Metrology and Language Sciences Unit, University of Mons
Place du Parc 18, B-7000 Mons, Belgium
george@mycontent.gr

Abstract

In this system demonstration we present the latest developments of Praaline, an open-source software system for constituting and managing, manually and automatically annotating, visualising and analysing spoken language and multimodal corpora. We review the system's functionality and design architecture, present current use cases and directions for future development.

1 Introduction

In this system demonstration, we present *Praaline*. *Praaline* is an open-source application for creating and managing speech and multi-modal corpora, annotating them manually or automatically, visualising the speech signal and annotations, querying the data and creating concordances, and performing statistical analyses. The system was first presented in (Christodoulides, 2014). *Praaline* is a cross-platform standalone application that runs on Windows, Mac OSX and Linux. It is written in C++ using the Qt framework. The application is available for download at `https://www.praaline.org` and the source code is available on GitHub (`https://github.com/praaline`), under the GPL version 3 license.

Several tools are being used currently in spoken language research. In the phonetics and prosody domain, the most widely used tool is *Praat* (Boersma and Weenink, 2018); in the pragmatics and discourse studies community Exmaralda (Schmidt and Wörner, 2009) is appreciated for its concordancing functionality; for multimodal and signed language corpora, *ELAN* (Brugman and Russel, 2004) is often preferred; and large corpora have been transcribed using *TranscriberAG* (Barras et al., 1998).

We have focused on creating a tool that will cover the needs of researchers working simultaneously on speech (phonetics, phonology, prosody) and discourse (syntax, semantics, pragmatics). *Praaline* should help an individual researcher organise their data using the best practices in corpus design and data management, and allow them to work independently. At the same time, our system aims to facilitate depositing the corpus data to a central repository at the end of a research project, using a structure that will facilitate long-term storage and reuse. For these reasons, we have decided to use an SQL relational database for data storage: an SQLite database for local projects, or a MySQL or PostgreSQL database for using *Praaline* in a network environment. Furthermore the system provides extensive support for importing and exporting data between *Praaline* and other annotation tools (*Praat*, *Exmaralda*, *ELAN*, *TranscriberAG*, other formats), as well as support for importing and exporting annotation data in the XML and JSON formats.

In the following, we will present the main functionality of *Praaline*, describe its design architecture and conclude with our plans for future development.

2 Features

2.1 Corpus Management

A collection of recordings (audio and video) constitutes a Corpus, which can be stored in a Corpus Repository. A repository provides a common structure for metadata and annotations (see next section). A Corpus contains a collection of Communications and Speakers. Each Communication consists of one or more Recordings and corresponding Annotation documents. Speakers are linked to the Communications in which they are involved though Participation objects. The six basic objects can all be described using a set of metadata. *Praaline* does not impose any schema for metadata: the user can define their own metadata schema per Repository. The Corpus Explorer (see Figure 1) is used to constitute the corpus and to edit its metadata: corpus objects can be browsed in a tree (possibly grouped based on metadata information); the user can also edit metadata using a spreadsheet-like display.

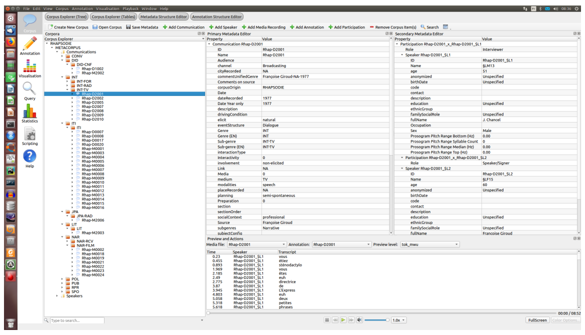

Figure 1: Corpus Explorer

2.2 Metadata and Annotation Structure

Each Corpus Repository will have a defined data structure for its metadata and annotations. Templates are provided to help individual researchers; it is also possible to standardise these structures when multiple users are accessing the same database. *Praaline* includes an editor for metadata and annotation structures: the editors perform data definition language (DDL) queries to change the SQL database structure. The metadata structure editor allows the user to define the fields that will be stored for each of the six main objects described in the previous section. Annotations are organised in Levels and Attributes: a Level describes a unit of linguistic analysis (e.g. a syllable, a token, an utterance etc) and contains a label and an arbitrary number of Attributes. Each Annotation Level corresponds to a table in the SQL database and Attributes correspond to columns. The data structure editor for annotations is shown in Figure 2).

The metadata and annotation structure editors allow the user to define fields having any of the standard SQL data types (integers, real numbers, strings, boolean values etc.). Furthermore, a system for defining and importing/exporting name-value lists is available. A name-value list (NVL) can be associated with any number of metadata fields or annotation attributes. The NVL subsystem can improve data quality by enforcing ref-

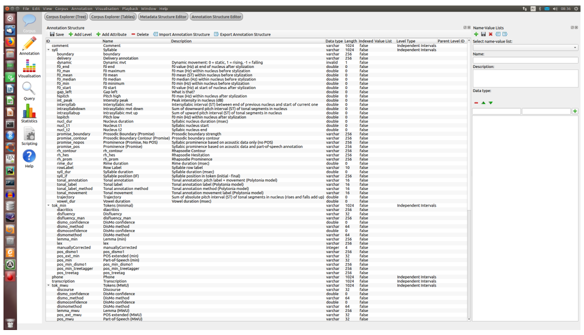

Figure 2: Annotation Structure and Vocabularies

erential integrity rules and help users in annotating data based on a closed vocabulary (e.g. a defined set of part-of-speech tags, dependency relations, discourse relations, phoneme labels etc.). The metadata and annotation structure of a corpus repository can be exported as an XML or JSON file; when these files are imported on another repository, *Praaline* recreates the corresponding database structures and NVL data.

2.3 Annotation

Annotations can be added to the corpus using one of the Manual Annotation editors: a spreadsheet-like editor that can combine multiple levels of annotation; a transcriber interface; an editor for sequences or for relations. The tabular editor is show in Figure 3.

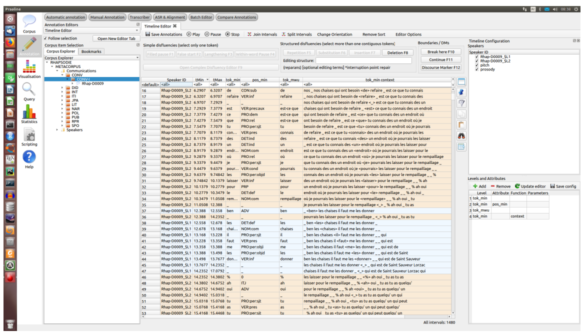

Figure 3: Manual Annotation Editor

The annotation tools offered attempt to cover the entire workflow of constructing a speech corpus: collecting recordings, transcribing them or re-using existing transcriptions, speech-text alignment, enrichment of the annotations with additional levels and feature extraction (before proceeding with analysing the data). The Transcriber module provides a user interface for quick manual transcription, especially in the presence of multiple speakers. The Long Sound Aligner module allows the user to reuse existing, possibly imperfect, transcriptions of the corpus materials. The Long Sound Aligner uses the output of a speech recognition engine and the transcription text to produce iteratively refined alignments of transcription utterances to the speech signal (similar to other long sound alignment tools, e.g. Katsamanis et al. (2011)). The Forced Alignment module allows the user to produce alignments at the phone, syllable and token level, based on a pronunciation dictionary or other system for phonetisation, and an ASR engine. The currently supported ASR engines in *Praaline* are HTK(Young et al., 2006), PocketSphinx (Walker et al., 2004) and Kaldi (Povey et al., 2011).

The annotation framework in *Praaline* is language independent: annotations are stored in Unicode format and no assumptions are made about language. However, several tasks require language-specific resources: tokenisation rules, pronunciation dictionaries, acoustic models and language models for the ASR engine. A collection of open resources is available, and the development of language resources for the back-end tools in the context of other open source projects can be harnessed.

Much of the functionality in *Praaline* comes from its automatic annotation plug-ins. The user can run a cascade of automatic annotation plugins, after setting the relevant parameters (each plugin defines its own set) on the entire corpus or on a subset of corpus items. The user interface for these operations is shown in Figure 4.

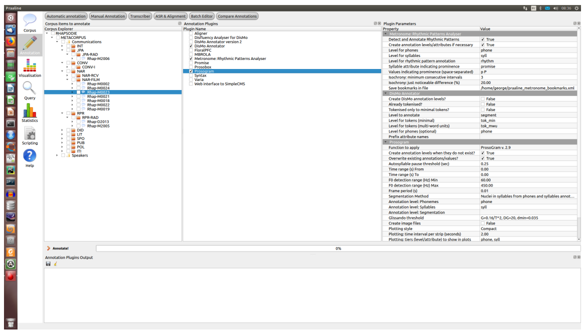

Figure 4: Automatic Annotation Plugins

Currently, plugins are available for the following tasks:

- Part-of-speech tagging and syntactical analysis of spoken language, in several languages, using the *DisMo* plug-in (Christodoulides et al., 2014). Statistical models for many languages are provided, based on the work by the Universal Dependencies project (Nivre et al., 2016).

- Prosodic feature extraction and pitch stylisation, using either the Prosogram (Mertens, 2004) system, or the INTSINT/MoMel system (Hirst, 2007) for intonation annotation.

- Automatic detection and annotation of prosodic events, with language-specific statistical models extracted from manually annotated corpora.

2.4 Visualisation

The Visualisation module of *Praaline* reuses and extends the code of *Sonic Visualiser* (Cannam et al., 2010), which is also an open-source (GPL) project written in C++/Qt and used in the field of musicology. The user can create the visualisation that best suits their needs by combining panes and layers, containing: annotation tiers, points, curves, histograms, colour-coded regions, spectrograms etc. Extensions have been added for visualising intonation information, for studies in prosody. Visualisation templates can be saved in XMl and JSON formats; a collection of available visualisation templates is presented to the user. An example of a visualisation is show in Figure 5.

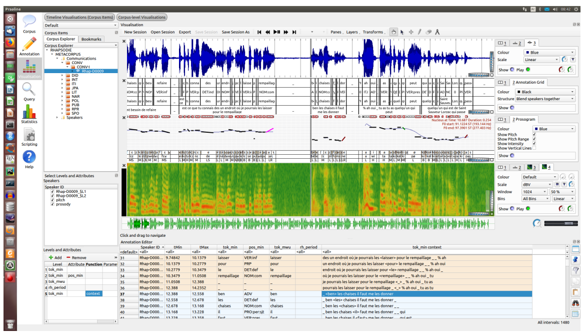

Figure 5: Visualisation

An annotation editor can be combined with the visualisation user interface, to facilitate use cases where the user codes a linguistic phenomenon with the aid of a visual representation. Visualisations can be exported in image formats for use in presentations and publications. A system of Bookmarks allows the user to save points of interest in the corpus: the identification data of the Communication, Recording and Annotation along with a time-code constitute a Bookmark that can be stored in collections of bookmarks for easy access. The results of a concordance search (see next section) can also be exported as bookmarks.

2.5 Queries and Concordances

The user can perform queries on any of the annotation levels or a combination of these levels. The results are displayed in keyword-in-context (KWIC) concordances and can be exported for further analysis. It is possible to search for sequences and patterns. The Concordancer results can be exported as Bookmarks for immediate access in the Annotation editors and in the Visualiser. The Concordancer is show in Figure 6.

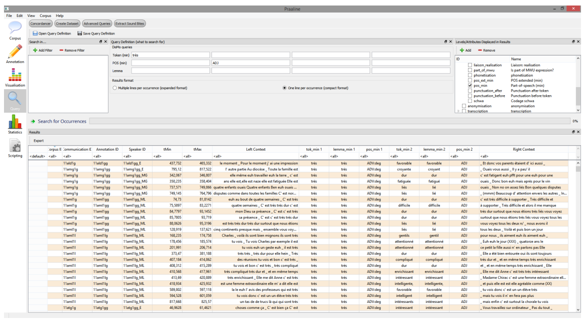

Figure 6: Concordancer

Furthermore, an interface is provided for creating datasets. Using a dataset, information from multiple annotation levels can be combined, a set of basic statistic operations can be applied and the results can be saved for further analysis. As an example, in a corpus annotated on four levels (phones, syllables, tokens, discourse units), the user may export a dataset containing: all the syllables in the corpus, their acoustic features (extracted from upstream plug-ins), the corresponding tokens and associated attributes (e.g. POS tags), the corresponding discourse units and associated attributes, and some metadata attributes. A dataset has a minimum unit (in the previous example, the syllable) and lower-level (grouped) or higher-level (associated) data.

2.6 Statistics

Praaline includes a set of statistical analysis plugins, covering a range of common analyses in speech research: basic counts; temporal analysis (pauses, speech rate, dialogue dynamics); prosodic and disfluency analysis; clustering corpus items using principal component analysis. The results of the analysis can be exported, and the basic graphics can be immediately accessed from within *Praaline*. For more advanced statistical analysis of corpus data, users can use R scripts (R Core Team, 2018) that take advantage of the fact that the corpus data and annotations are stored in SQL format. An example of a PCA analysis is shown in Figure 7.

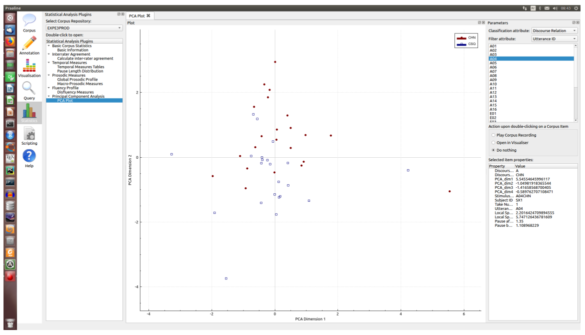

Figure 7: Statistics (PCA plot)

3 System Architecture and Extensibility

The system is following a modular design. A core library (Praaline Core) contains all the basic functionality needed to access and manage *Praaline* corpus metadata and annotations. The library can be reused in other software (see also next section). At the next level, a set of modules (Praaline Libraries) contain the functionality needed for automatic speech recognition, the application of machine learning algorithms, feature extraction etc. These libraries are often wrappers of existing and well-established open-source NLP tools, such as Sphinx, OpenSmile, HTK, CRF++ etc. An additional module contains the user interface elements (e.g. widgets for visualisation). All these modules are brought together in the main application.

An API for the Core library and for interfacing with the main application allows for the creation of plugins. Plugins add functionality to *Praaline* and include implementations of automatic annotation algorithms, statistical analysis plugins or new visualisation elements.

Examples of plug-in development are provided on the project's website: a skeleton C++ project is given, and Python scripts using the Praaline Core library, or other modules of the Praaline Libraries illustrate the use of these resources for common NLP tasks. The Praaline Core library can be used as a tool for input-output and interfacing with annotation information without using *Praaline*'s user interface.

4 Future Development

Praaline is under active development. The main priorities for adding features are as follows:

- A signal annotation editor strongly resembling Praat. We have found that researchers in some communities are strongly attached to this user interface, and this addition will facilitate the acceptance of a new system on their part.
- Stabilise the API of the Praaline Core library and provide Python and R bindings.
- Finalise the development of *PraalineWeb*, a system that allows the publication of corpora on the web, using Django.
- Add functionality for editing dependency relations.

The development of *Praaline* has been mainly driven by the expectations and needs of its users. It is hoped that this system demonstration will provide additional feedback.

References

Claude Barras, Edouard Geoffrois, Zhibiao Wu, and Mark Liberman. 1998. Transcriber: a free tool for segmenting, labeling and transcribing speech. In *LREC 1998 – 1st International Conference on Language Resources and Evaluation, 28–30 May, Granada, Spain Proceedings*, pages 1373–1376.

Paul Boersma and David Weenink. 2018. *Praat: doing phonetics by computer, ver. 6.0.37*.

Hennie Brugman and Albert Russel. 2004. Annotating multimedia and multi-modal resources with elan. In *LREC 2004 – 4th International Conference on Language Resources and Evaluation, May 26–28, Paris, France, Proceedings*, pages 2065–2068.

Chris Cannam, Christian Landone, and Sandler Mark. 2010. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the ACM Multimedia 2010 International Conference*, pages 1467–1468.

George Christodoulides. 2014. Praaline: Integrating tools for speech corpus research. In *LREC 2014 – 9th International Conference on Language Resources and Evaluation, May 26–31, Reykjavik, Iceland, Proceedings*, pages 31–34.

George Christodoulides, Mathieu Avanzi, and Jean Philippe Goldman. 2014. DisMo: A morphosyntactic, disfluency and multi-word unit annotator. In *LREC 2014 – 9th International Conference on Language Resources and Evaluation, May 26–31, Reykjavik, Iceland, Proceedings*, pages 3902–3907.

Daniel Hirst. 2007. A praat plugin for momel and intsint with improved algorithms for modelling and coding intonation.

Athanasios Katsamanis, Matthew P. Black, Panayiotis G. Georgiou, Louis Goldstein, and Shrikanth S. Narayanan. 2011. SailAlign: Robust long speech-text alignment. In *Proceedings of the Workshop on New Tools and Methods for Very-Large Scale Phonetics Research.*

Piet Mertens. 2004. The Prosogram: Semi-automatic transcription of prosody based on a tonal perception model. In *Proc. of Speech Prosody 2004, March 23–26, Nara, Japan*, pages 549–552.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, and et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016).*

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

R Core Team. 2018. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.

Thomas Schmidt and Kai Wörner. 2009. EXMARaLDA – creating, analysing and sharing spoken language corpora for pragmatic research. *Pragmatics*, 19(4):565–582.

Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. 2004. Sphinx-4: A flexible open source framework for speech recognition. Technical report, Mountain View, CA, USA.

Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. 2006. *The HTK Book Version 3.4*. Cambridge University Press.