The JSafran platform for semi-automatic speech processing

Christophe Cerisara, Claire Gardent
LORIA-CNRS UMR-7503 Vandoeuvre-les-Nancy, FRANCE
{cerisara, gardent}@loria.fr

Abstract

JSafran is an open-source Java platform for editing, annotating and transforming speech corpora both manually and automatically at many levels: transcription, alignment, morphosyntactic tagging, syntactic parsing and semantic roles labelling. It integrates preconfigured state-of-the-art libraries for this purpose, including the Sphinx4, TreeTagger, OpenNLP, MaltParser and MATE applications, as well as the companion JTrans software for text-to-speech alignment and transcription. Despite the complexity of such speech processing tasks, JSafran has been designed to maximize simplicity both for the end-user, thanks to an easy-to-use GUI that controls all of these automatic and manual annotation functionalities, and for the developer, thanks to well-defined interfaces and to the multi-level stand-off annotation paradigm. JSafran has been used so far for several tasks, including the creation of a new French treebank on top of the broadcast news ESTER corpus. 

Index Terms: JSafran, dependency parsing.

1. Introduction

Many tools have been developed to process speech signals, such as automatically transcribing speech, aligning speech to existing transcriptions, computing prosodic parameters, parsing speech transcriptions into syntax or semantic frames, etc. Yet, most of these tools can hardly be used apart from specialists of the domain, because they usually require careful configuration and scripting. On the other hand, many professional end-users, and especially linguist researchers, may benefit from exploiting the power of such semi-automatic tools when manipulating large corpora for validation or analysis purposes. But designing such software that both (i) provide ready-to-use advanced semi-automatic speech processing functionalities, and (ii) give a total control to the end-user, who often wants to manually check at least part of the corpus to control the quality of the result, is very challenging. This work hence follows a number of successful efforts that have already been made in this direction, as briefly reviewed in section 2. The JSafran platform is described in section 3, while section 4 presents some of its most typical use cases.

2. Related works

2.1. Existing platforms

Many large scale efforts have already been deployed to design software platforms for the annotation of speech corpora. Hence, the EXMARaLDA software is especially rich with regard to interacting between the audio, video and text views of a corpus, as well as with a rich support for standardized I/O formats.

Similarly, Transcriber is now a widely used platform for manual transcription and annotation of speech corpus. Another well-known annotation platform is the Praat software, which has become a de facto standard for prosodic and phonological work. It can also be used to some extent for text and speech alignment. A related tool that includes semi-automatic algorithms for aligning speech signals is WinPitch. Other well-known applications dedicated to manual speech transcription are ANVIL and CLAN. Some of these applications may also be used to annotate corpus syntactically, but they are not specifically designed for this purpose. Specialised platforms for syntax include Synpathy, ANNIS, which provides a very powerful search language and feature, or Vakyarta and EasyRef designed to support collaborative annotations. The platform described in [1] supports several annotation layers from speech segments up to semantics. The GATE framework also integrates many text processing functionalities into a unified platform. This list is naturally limited and far from exhaustive.

2.2. Why yet another speech processing application?

The first reason to use so many different tools doing, at first sight, similar tasks is the multiplicity of research domains in linguistic and speech processing. Any single tool, even though it has been carefully designed to be as generic as possible, will never do exactly what you want. This multiplicity of usages justifies to regularly develop new applications. JSafran is just the same: it is likely to be usable by a focused community, but not adequate for others. Hence, long-term reusability strongly depends on application-independent factors, and in particular on exchangeability of meta-data between applications as well as on software components adaptability, as discussed in section 3.4.

3. Description of the JSafran platform

3.1. Rationale

The JSafran software is the result of a joint initiative between linguists and computer science researchers, with the objective of developing advanced tools for processing speech corpora semi-automatically. It now constitutes an integrated yet flexible solution for processing speech at many levels, from speech transcription and alignment up to semantic role labelling.

JSafran proposes two dual corpus manipulation approaches: automatic processing and manual editing. It is de-
signed so that the end-user may easily switch from one approach to the other, hence benefiting from the power of automatic algorithms to speed-up the annotation process, while still preserving a complete control of the end-user on the produced annotations. Both approaches are detailed next.

3.2. Automatic processing

3.2.1. Automatic transcription and text-to-speech alignment

Although JSafran works well with written text corpus, it has been primarily designed to handle speech transcriptions. An important feature for manipulating speech corpora is the possibility to easily access and listen to the audio segment that corresponds to any transcribed utterance. This may be used for instance to help disambiguating sentence boundaries, part-of-speech tags or dependencies thanks to prosodic cues. We previously developed the JTrans software [2], which proposes semi-automatic algorithms to facilitate and speed-up manual alignment of speech transcriptions to the corresponding audio segments. It supports very long audio files with imperfect transcriptions, thanks to an automatic incremental alignment background process. Once aligned, the words colour changes, which gives some feedback to the user about the current state of the alignment process. The end-user can then navigate in the audio file by simply clicking on such a word and visually check for the correctness of the alignment with a kind of “karaoke” playback mode. More precise temporal limits between words or phones can be displayed and edited on a spectrogram panel. JTrans has been recently upgraded to further propose automatic transcription algorithms accessible with a single mouse click, which can be used when only audio files are available. These functionalities rely on the external Sphinx4 libraries. A default set of triphone acoustic and 4-gram language models for French are provided.

JTrans is now integrated within the JSafran platform, which gives the end-user the possibility to listen to and align the utterance that he is currently annotating in JSafran. Concretely, JTrans remains independent from the other JSafran modules, but communicates with them via simple meta-data files and common references to immutable data sources, as described in section 3.4.

3.2.2. Automatic tagging and parsing

Once the speech signal has been transcribed and aligned, the next steps proposed in JSafran consist in lemmatizing and annotating the words with morpho-syntactic (part-of-speech or POS) tags, such as “noun”, “infinitive verb”, etc. This step is for now handled fully automatically in JSafran. Three POS-taggers are integrated in JSafran: the TreeTagger [3] and its Java wrapper 11, the OpenNLP tagger 12 and the MATE tagger [4]. The TreeTagger is configured and used by default for French. Pre-trained OpenNLP tagging models that have been trained on the French Treebank [5] are also available.

Syntactic parsing can now be realized on the tagged words sequences. This step is crucial because it exhibits a tree-like structure, which relates words that may be far apart one from the other, and because this structure is the support for computing detailed semantic annotations. We have chosen in JSafran the dependency theory, in contrast to the more traditional constituency-based hierarchy, because dependency structures tend to be more commonly used nowadays, as in the CoNLL campaigns. A syntactic dependency parse builds on top of the words of the utterance a tree-like structure, where each word is a node and any labelled oriented dependency links one head word to one of its dependent words. Two of the best state-of-the-art parsers are integrated within JSafran: the MaltParser [6] and the MATE parser [4]. A set of default pre-trained parsing models for French are included. They have been trained on the Ester Treebank, which is described in section 4.1. The version of the MaltParser that is integrated within JSafran has been slightly modified in order to support:

- Sentence-level probability estimates: this is made possible by first computing local posterior probabilities from the SVM model that is used in the modified Shift-Reduce algorithm. These local probabilities are derived based on the approach proposed in [7], and are then averaged over the whole sentence.
- Constrained parsing, i.e., preserving all dependencies that have previously been set and filling in the missing dependencies. This may be used for instance to complete a previous partial annotation, or to adjust the whole parsing after manual correction of a single dependency.

3.2.3. Semantic Role Labelling

JSafran actually supports viewing and editing multi-level oriented graphs, and not only trees, where nodes correspond to words. We have used this feature to edit other types of graphs, in addition to syntactic dependencies, and in particular Semantic Role Labels. It is further possible in JSafran to associate several layers with a single utterance, for instance one for syntactic and another one for semantic relations. We have also used this possibility to create a third layer for coreference relations when studying left dislocations. Although links are constrained to always be attached to two words, it is also possible to group together several consecutive words, each group being visualized by an under-line that spans all the words of the group.

The automatic Semantic Role Labelling of an utterance is currently based on predefined rules and associated French lexicons, as described in [8]. It can only be applied for now to known verbs and subcategorisation frames of the French Treebank, because of resources constraints.

3.2.4. Search & transformation language

Two important features when working with linguistic corpora are (i) to be able to navigate in the corpus by searching for specific structures, and (ii) to be able to automatically transform some structures with rules, in order to account for modifications in the annotation guide, as well as for normalising and merging several corpora annotated with different annotation schemes. Both features are supported through a single “search & transformation dependency language” within JSafran.

This language is based on the matching of variables to word forms, lemmas, POS-tags or dependency labels, which are combined with constraints. Typical constraints are, for instance, to search for words \( (w_1, w_2) \) for which the head of \( w_1 \) is also a dependent of \( w_2 \), or the head of \( w_2 \) precedes the head of \( w_2 \) by two words, or the root of the subtree containing \( w_1 \) is the verb “to eat”, etc. We will not describe further this language, which supports a wide range of constraints and rewriting rules, as shown with examples on JSafran’s web page15. This language has been used for three purposes so far:

1. Normalisation and merging of successive versions of the Ester Treebank that were based on different versions of the annotation guide;
2. Extraction and automatic annotation of left dislocations;
3. Extraction and annotation of different types of negative structures in French.

Note that the development of such search and transformation rule-based languages is a very challenging topic on its own. In the near future, we plan to reuse existing dedicated libraries, such as AGG [14], but in the meantime, JSafran’s search & transformation language is a temporary but useful and powerful solution to manipulate dependency graphs.

3.2.5. Statistical analysis of the corpus

Working on a linguistic corpus requires to browse it, edit erroneous links, attach comments to difficult examples, or search for and transform specific structures. JSafran provides all these functionalities and more, such as various statistical analysis of a corpus, which are important to detect errors and clean them. The following statistics are provided in the current version, with a direct access to the utterances that match these distributions:

- Distribution of the number of roots per utterance;
- Distribution of the dependency labels per POS-tag of the dependent word;
- Distribution of the POS-tags per dependency labels;

These statistical tools can easily be extended programmatically.

3.3. Manual editing

The JSafran GUI has been designed to allow for fast manual editing of dependency graphs. Keyboard controls are intensively used for this purpose: this makes it slightly more difficult to learn than with the mouse and menus, but after a bit of training, it is much faster. Every utterance in a corpus is numbered and displayed on its own line in JSafran. Navigation in the corpus is intuitively realized with arrow keys, and control-arrow keys to jump to a specific utterance. Left and right keys as well as mouse clicks allow to select a given word in the utterance, and the ‘d’ key toggles between view and dependency editing mode. In editing mode, arrow keys move the head of the current dependency and pop up a menu to choose from potential dependency labels.

This simple approach seems to be the fastest and easiest solution to edit dependency arcs amongst all pieces of software we have tried so far. A part of the main JSafran window is shown in figure 1.

3.4. Software design and architecture

Although JSafran does not implement a real component-oriented architecture, like OSGI or WebServices, it has been carefully designed to minimize dependencies between well-defined focused Java modules and programs. The main idea behind JSafran is to exploit several independent focused tools that may be easily made accessible from within the main GUI. Data sharing hence becomes a major requirement, which has been met thanks to the following guidelines:

- Immutable data source: there are two basic types of objects that are manipulated: text and audio. We assume the original corpora are immutable, and we keep references to the original objects and positions in further processing to ease data sharing.
- Stand-off annotations: we serialize several focused simple meta-data files per application rather than storing different types of meta-data within the same file. These meta-data files are in de facto standard formats, which depend on the type of meta-data considered, but shall always be as simple as possible. Every module is required to support such formats, which is usually not a problem given their simplicity. This approach obviously limits the expressive power of each serialized file, but the overall complexity is handled by the dependency between these different files.

One of the objective that has motivated the development of the JSafran platform is to maintain usage simplicity at all levels: for the end-user, thanks to an easy-to-use GUI, and for the developer, thanks to simple data formats and minimal integration efforts.

4. Instantiations and use cases

4.1. Creation of the first French broadcast news treebank

JSafran has been used in an iterative process to create a French speech treebank, called the “Ester TreeBank” (ETB) because it is based on the French broadcast news ESTER corpus used in national speech recognition evaluation campaigns. Six linguistic students were hired to manually annotate this corpus with dependencies. A first set of MaltParser models have been initially trained on a small bootstrap corpus, and the students were then asked to correct the automatic parsing. Better parsing models have then been iteratively trained and used during the corpus creation process. As of march 2011, the ETB is composed of 50,000 words, 60% of which correspond to journalistic “prepared speech”, while the remaining 40% correspond to more spontaneous “guest speaker” speech coming from interviews [9]. The Labelled Attachment Score, i.e. the ratio of the words that have been automatically assigned the correct dependency label and the correct head word, of the best parsing models trained and tested on this corpus is 76.2%. This result is still low when compared to the one obtained on other corpora, such as the French treebank (FTB), which is greater than 83% when computed on similar conditions, i.e., without punctuation and with automatic POS-tags. But it is also encouraging, because the ETB is six times smaller than the FTB, and increasing the size of the ETB shall help reduce this difference.
4.2. Study of specific speech phenomena

In addition to dependency arcs, JSafran also proposes utterance-level annotation tags. This feature has been used to annotate one fifth of all ETB utterances with the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Annotation Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>This example is totally corrupted and should be removed</td>
</tr>
<tr>
<td>DUNNO</td>
<td>This utterance is difficult to annotate</td>
</tr>
<tr>
<td>DISFLUENCY</td>
<td>This utterance contains disfluencies</td>
</tr>
<tr>
<td>ELIPSIS</td>
<td>This utterance contains ellipsis</td>
</tr>
<tr>
<td>DISLOC</td>
<td>This utterance contains dislocations</td>
</tr>
<tr>
<td>CLEFT-RC</td>
<td>This utterance contains clefts</td>
</tr>
<tr>
<td>GUEST-SPEECH</td>
<td>It comes from a guest-speaker, as opposed to a journalist</td>
</tr>
<tr>
<td>HEADLINE</td>
<td>It is a special kind of “broadcast news” utterance used to structure the news, announce the next speaker.</td>
</tr>
<tr>
<td>EXAMPLE</td>
<td>Remarkable example that deserves to be cited as an example in the annotation guide</td>
</tr>
</tbody>
</table>

These utterance-level tags have been used especially to study the impact of different speaking styles onto parsing accuracy, as reported in [9].

5. Conclusions

5.1. Summary

Creating, browsing, annotating and transforming speech corpora is a fundamental requirement of many research areas in Natural Language Processing and linguistics. But it also often constitutes a major bottleneck in such researches, because of its very high costs in terms of human involvement. Advanced state-of-the-art automatic processing algorithms exist and might help to reduce these costs, but they are often very complex and thus only usable by specialists. JSafran participates in recent efforts to bring together such tools and make them easily accessible and usable within a single GUI. The main advantages of such an effort are twofold: (i) to reduce the cost for creating and annotating speech corpora; the resulting corpora may be used either for linguistic analysis and validation purposes or training new machine learning algorithms; (ii) to open new perspectives in terms of semi-supervision, or how to exploit the feedback loop between the automatic algorithms and the end-user for improving these algorithms.

5.2. JSafran added value

The main strength of JSafran is the integration, within an easy-to-use GUI, of very complex and powerful automatic speech processing algorithms, and in particular: speech transcription, speech and text alignment, POS-tagging, dependency parsing and semantic role labelling. All these algorithms are accessible in a few clicks from JSafran’s menus and are already pre-configured, so that they can run "out of the box". They are paired with a nice visualization GUI that allows for fast and efficient manual corpus editing, as well as with a rule-based language designed to automate many corpus transformation tasks. Despite the integration of such highly-complex algorithms, JSafran has been developed to remain simple both for the end-user and for the developer, thanks to a design principle that relies on interfacing with many independent applications and to the stand-off annotation paradigm. Finally, JSafran is naturally portable to many platforms thanks to Java, and is open-source.

As discussed in section 2, we do not claim that JSafran shall replace other software, which might be better suited for other particular research tasks. Trying to design a unique NLP software solution for most researches is at the opposite of the design principles behind JSafran, which rather promotes the sharing of software components and the continuous evolution and development of new focused applications and their interfacing via common simple data formats.

5.3. Extension to new languages

JSafran only supports for now the French language. However, it may be relatively easily extended to another western language, such as English or German. This may be done as follows:

- **Transcription and speech-to-text alignment**: JTrans needs language-specific acoustic and n-gram models respectively in the CMU Sphinx and ARPA formats, as well as a phonetic lexicon and a WEKA model for proper names phonetisation.
- **POStags**: The TreeTagger supports several languages; adapting JSafran to another one of these languages that is supported by TreeTagger simply requires to reconfigure the TreeTagger for this language. For other languages, the alternative POS-tagger based on OpenNLP can be used. The last option consists in computing the POS-tags with an external tool and loading them "as they are" in JSafran from CoNLL files for instance.
- **Syntactic dependency models**: Language specific parsing models have to be trained; this can be done from within JSafran for the MaltParser and the MATE parser.
- **Semantic role labelling**: Because it is based on French rules and resources, this feature cannot be adapted to another language for now. We plan to integrate the MATE SRL library in JSafran in the next release though.

6. Acknowledgements

The authors would like to thank the CPER MISN TALC and the Région Lorraine for their support, as well as several linguistic researchers of the ATILF laboratory for their help in the design and testing of JSafran & JTrans.

7. References